

POLITECNICO DI TORINO

III Facoltà di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

**Sistema automatico di
classificazione semantica e
clustering di messaggi di posta
elettronica residenti su fonti
IMAP distribuite**



Relatore:

prof. Angelo Raffaele Meo

Correlatore:

prof. Federico Di Gregorio

Laureandi:

Biagio MEIRONE
Giuseppe RIZZO

ANNO ACCADEMICO 2007-2008



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

*Alle persone a noi care
che ci hanno sostenuto
in questa esperienza
meravigliosa, sempre
ricca di sorprese.*

Sommario

Nel mondo dell’informatica e delle nuove forme di comunicazione l’utilizzo della posta elettronica si è diffuso a macchia d’olio, tanto da rappresentare l’applicazione più conosciuta ed utilizzata attualmente. Il lavoro svolto nasce dall’esigenza di classificare i messaggi secondo metodi che sfruttassero una fase di learning automatica e supervisionata. La mente umana è solita organizzare le categorie di problemi o insiemi di eventi in modo correlato, oppure in modo esclusivo. La categorizzazione delle mail può seguire questa strada, utilizzando domini esclusivi, contenitori di argomenti totalmente scorrelati con il resto, e insiemi con inferenze tra loro. Pertanto si è cercato di utilizzare delle etichette, chiamate anche “tag”, molto flessibili che meglio rappresentano le relazioni tra gli elementi contenuti nel dominio di conoscenza.

Per classificazione di messaggi si intende un processo automatico di assegnazione di una mail ad una certa classe o ad una categoria. In questo ambito, un aspetto interessante riguarda la creazione di nuovi modelli per rappresentare un messaggio. Il lavoro presentato in questa sede introduce un nuovo modello bayesiano probabilistico che estende il classificatore Naive Bayes per la classificazione di mail in “spam and ham” in multiclassi ed un modello di Support Vector Machine.

L’idea su cui si fondano entrambi i metodi consiste in una rappresentazione gerarchica dei messaggi, garantito dalla struttura intrinseca della mail. Questo ci permette di modellare ogni singolo messaggio come un insieme di sezioni. Ogni sezione può essere vista come un sottoargomento ed è rappresentata da una variabile nascosta. La classificazione della mail avviene in base ai suoi campi: si introduce così un livello di categorizzazione intermedio non presente nel modello del Naive Bayes “spam and ham” e si permette di dare maggiore importanza ad alcuni campi, introducendo così una notazione pesata.

L’utilizzo di un doppio classificatore permette di creare una struttura di etichette che forniscono statistiche e indicazioni sommarie sui metodi di classificazione. L’approccio sviluppato analizza i vantaggi ed i limiti di questo nuovo motore di classificazione rispetto ai modelli già presenti.

Indice

Sommario	vi
1 Introduzione	1
1.1 Intelligenza artificiale	2
1.2 Apprendimento probabilistico	3
1.3 Apprendimento supervisionato	4
1.4 Addestramento e conoscenza del dominio	6
1.5 Strutturazione della tesi	8
2 Stato dell'arte	11
2.1 Scenario applicativo	11
2.1.1 Interazione Client e Server	13
2.1.2 Modello WebService	14
2.2 Client di posta elettronica	17
2.2.1 Mozilla Thunderbird	17
2.2.2 Apple Mail	18
2.2.3 Novell Evolution	19
2.2.4 Microsoft Outlook	19
2.2.5 Eudora	20
2.3 Web o Web Service Mail	20
2.3.1 Gmail	21
2.4 Protocolli	22
2.4.1 POP - Post Office Protocol	22
2.4.2 IMAP	23
2.4.3 SMTP	24
2.5 Regole di classificazione delle mail	26
2.6 AntiSpam	27
3 Il teorema di Bayes	29
3.1 Apprendimento Bayesiano	30
3.2 Rete di credenze	31

3.2.1	Inferenza nelle reti di credenze	33
3.2.2	Apprendimento delle reti bayesiane	34
3.3	Naive Bayes	35
3.3.1	Utilizzo del Naive Bayes per input discreti	36
3.3.2	Il classificatore Naive Bayes	38
3.4	Inferenze del Naive Bayes e catene di Markov	41
4	Support Vector Machine	45
4.1	La teoria dei kernel	46
4.2	Teoria della generalizzazione	48
4.3	Teoria di Vapnik-Chervonenkis	49
4.4	Duale di Wolfe	54
4.5	Programmazione quadratica	56
4.6	Programmazione quadratica per SVM lineari	57
4.7	Il caso elementi non separabili linearmente	59
4.8	SVM per problemi di regressione	61
5	OWL per la descrizione di ontologie	63
5.1	Resource Description Framework	64
5.1.1	Principi base di RDF	65
5.2	Web Ontology Language	66
6	Architettura di sistema	69
6.1	Protocolli di comunicazione	69
6.2	Classificazione del testo	70
6.3	Apprendimento	72
6.4	Filtri di classificazione	73
6.4.1	Bayes	74
6.4.2	N-grams Bayes	78
6.4.3	Support Vector Machine	80
6.5	Gestione dei messaggi	82
6.5.1	Corpo del messaggio	82
6.5.2	Riconoscimento lingua e le stopwords	82
6.5.3	Stemming	86
6.5.4	Firme	87
6.6	Rubrica	88
6.6.1	Levenshtein: distanza tra due espressioni	89
6.7	Base dati	90

7 Sviluppo dell'applicativo	93
7.1 Progettazione del software	94
7.2 Accesso alle risorse	96
7.3 Apprendimento	98
7.4 Classificazione	103
7.5 Strategie di controllo	105
7.6 Flussi di discussione	107
7.7 Base Dati	108
8 Risultati	111
8.1 Scenario dei test	111
8.2 Risultati acquisiti	114
8.3 Confronto tra le simulazioni	114
9 Conclusioni	133
9.1 Sviluppi futuri	134
A Libsvm	137
B Tabelle	139
C Grafici delle simulazioni	147
D Schemi relazionali	153
D.1 Icetricky	153
D.2 Icetricky-Configuration	154
E Diagrammi delle classi	159
F Schema dell'ontologia	163
Bibliografia	165

Capitolo 1

Introduzione

L'avvento di internet e la sua rapida evoluzione ha di molto modificato il modo di comunicare tra le varie persone, sia in ambito lavorativo sia nel tempo libero.

Siete appena tornati da una rilassante vacanza di due settimane. Quindici giorni senza cellulare e senza leggere la propria posta elettronica. Vi collegate alla vostra inbox e ci sono circa trecento messaggi di posta elettronica. Come gestire la lettura di tutti quei messaggi? Probabilmente occorre un giorno intero per ordinare tutti i messaggi, per rileggerli e successivamente occorrerebbe un'altra vacanza!

In base a statistiche effettuate, in media un utente “business” riceve circa cinquanta messaggi di posta elettronica al giorno sia per motivi di lavoro che per motivi personali. Per questo molti utenti spendono parte del loro tempo di lavoro per analizzare tutte le mail ricevute. A questo, nel corso del tempo, si è aggiunta la necessità di bloccare lo spamming¹.

Per soddisfare le esigenze sempre più sofisticate degli utenti, è necessario sviluppare delle tecniche di apprendimento automatico che elaborino i dati secondo dei criteri intelligenti. La teoria della probabilità fornisce gli strumenti necessari per il ragionamento in presenza di incertezza, mentre le reti bayesiane e il modello SVM permettono la creazione di classificatori di documenti molto efficaci.

I metodi tradizionali per il trattamento e la ricerca dell'informazione non sono più adatti a soddisfare le esigenze dell'utente. La programmazione classica è basata sulla conoscenza che il progettista ha del dominio in esame. Spesso, però, il sistema da modellare è molto complesso, in costante cambiamento ed il progettista può avere solo una conoscenza incompleta dell'ambiente che rappresenta. In un contesto così eterogeneo e articolato, serve un qualche criterio intelligente in grado di emulare delle forme di ragionamento e di apprendere dall'esperienza fornita dal

¹ *spamming*: invio di grandi quantità di messaggi indesiderati generalmente pubblicitari. Può essere messo in atto attraverso qualunque media, ma il più usato è Internet, attraverso l'email.

dominio dell'applicazione.

1.1 Intelligenza artificiale

Imparare dalla conoscenza del dominio, modificare il proprio agire in funzione di argomenti, dati immagazzinati nel corso delle esperienze di vita. Sulla base di queste considerazioni, nel corso degli anni si è sviluppata l'idea di Intelligenza Artificiale. Nel corso degli anni numerose definizioni sono state proposte, riportiamo quella di Kurzweil².

“È l’arte di creare macchine che eseguono funzioni che richiedono intelligenza se vengono eseguite da persone”

Sebbene gli studi condotti sull'intelligenza artificiale si sono ispirati all'imitazione dell'intelligenza umana, i risultati ottenuti non potranno mai presentare tutti i gradi di libertà tipici della sfera umana. Infatti, numerose scuole di psicologia considerano l'intelligenza come distinta da tratti della personalità come il carattere, la creatività e la saggezza. A tal fine, nel corso degli anni, la ricerca è stata mirata su macchine in grado di apprendere, infatti, da quando sono stati inventati i calcolatori, si è sempre cercato di capire se fosse stato possibile realizzare delle macchine in grado di imparare. Si può quindi affermare che l'intelligenza artificiale è rivolta alla costruzione di sistemi intelligenti, che nel nostro caso prendono la forma di Machine Learning³. Questo settore cerca di adattare a nuove circostanze la macchina, raffinando la sua conoscenza e studiando i meccanismi mediante i quali il comportamento di un agente o esperto del dominio risulta migliorato sulla base di esperienze o casi precedentemente trattati.

L'apprendimento automatico è un diretto discendente di una disciplina più vecchia, l'adattamento di modelli statistici ai dati osservati e come quest'ultima ha lo scopo di estrarre informazioni utili da un insieme di dati, costruendo dei buoni modelli probabilistici. La novità principale introdotta è il sistema automatizzato del processo fino al raggiungimento dell'obiettivo, spesso con l'utilizzo di un vasto numero di parametri. L'apprendimento automatico viene utilizzato soprattutto in situazioni in cui non è semplice trovare una soluzione utilizzando degli algoritmi

²Raymond Kurzweil: è un inventore ed un futurista. È stato un pionere nei sistemi di “Optical character recognition”, “text-to-speech synthesis”, “speech recognition technology”. È uno degli artefici di “One Laptop Per Child Project”.

³Machine Learning: programmi o automi che migliorano automaticamente le loro prestazioni con l'esperienza.

tradizionali. Questo è dovuto tipicamente alla presenza di alcuni fattori che possono essere elencati come segue:

- *Difficoltà di formalizzazione*: supponiamo, ad esempio, che un nostro amico ci invii due e-mail nelle quali in una ci chiede a che ora possiamo incontrarci la sera per andare ad un concerto e nell'altra ci invia i dettagli di un corso che segue con noi all'università. Mentalmente ci basta leggere il contenuto delle mail per poterle catalogare facilmente nelle varie cartelle di posta che ci siamo creati sul nostro client e-mail, ma difficilmente riusciamo a descrivere una sequenza di passi computazionali che classifichi queste e-mail che ci sono arrivate e tutte quelle che arriveranno o che sono già arrivate.
- *Elevato numero di variabili in gioco*.
- *Mancanza di una teoria*: ad esempio non esistono leggi matematiche su cui basare l'andamento dei titoli in borsa.
- *Necessità di personalizzazione*: ad esempio la classificazione delle e-mail dipende esclusivamente dall'utente visto che normalmente ognuno di noi suddividerà i propri messaggi di posta elettronica in base alle sue esigenze e alle attività che svolge.

In questo settore si è affermata la disciplina del Data Mining, ossia l'estrazione della conoscenza dai dati a disposizione.

Cercando di definire un programma in grado di apprendere dall'esperienza, possiamo affermare:

Un programma è in grado di apprendere dall'esperienza E rispetto ad un certo compito C e ad una certa misura di prestazioni P, se le sue prestazioni nell'eseguire il compito C, misurate rispetto a P, migliorano con l'esperienza E.

1.2 Apprendimento probabilistico

“Il mondo sembra casuale perché non lo abbiamo descritto a livello giusto?”

A questa domanda hanno cercato di rispondere i sostenitori dei sistemi a logica formale⁴. Il concetto sul quale si basano questi sistemi è la consapevolezza di avere

⁴*Logica formale*: scuola di pensiero che mira a rifiutare il carattere contraddittorio della realtà e affermano l'identità, negano lo sviluppo e il cambiamento dei concetti e delle cose e sostengono la rigidità e l'immobilismo.

delle informazioni le quali generano delle conclusioni che possono essere false o vere. In molte situazioni però, non si può avere quasi mai accesso all’intera verità del dominio che si sta esaminando e vi saranno domande a cui non si potrà dare una risposta certa. I sistemi generalmente non sono statici, ma nella loro evoluzione nascono nuovi concetti che possono essere contraddittori tra loro, introducendo grandi di incertezza. Il sistema deve quindi agire in presenza di incertezza. La teoria della probabilità fornisce le basi per il trattamento dei sistemi che ragionano con incertezza. La conoscenza che il nostro sistema riesce a derivare, può fornire al massimo un grado di credenza sulla soluzione di un certo problema. La probabilità fornisce un modo per riassumere l’incertezza che deriva dalla non completa conoscenza del dominio. Sono stati proposti diversi tipi di ragionamento (Cfr. [24]):

1. Logica non monotona: permette di cancellare e aggiungere enunciati alla base dati. Rappresenta quasi un ossimoro, poiché logica definisce una deduzione, mentre non monotono significa che le conclusioni sono rivedibili, correggibili. Si contrappone fortemente alla logica formale.
2. Ragionamento probabilistico: permette di rappresentare associazioni probabili ma non certe.
3. Logica sfumata: è una logica polivalente e pertanto un’estensione della logica booleana in cui si può attribuire a ciascuna proposizione un grado di verità compreso tra 0 e 1.

Poiché i dati disponibili sono affetti da rumore ed essendo costretti a ragionare in un ambiente incerto, l’approccio probabilistico fornisce una teoria robusta che unifica differenti tecniche.

1.3 Apprendimento supervisionato

L’apprendimento supervisionato è una particolare tecnica di apprendimento automatico il cui scopo principale è istruire un sistema informatico a svolgere dei compiti, in particolar modo si pone l’obiettivo di risolvere dei problemi in automatico. Lo schema di questo tipo di tecnica si può riassumere in tre punti principali:

1. Si definiscono i dati in ingresso come un insieme I , tipicamente dei vettori.
2. Si definisce in uscita l’insieme O ; gli output possono essere valori continui in caso di regressione oppure delle etichette numeriche.
3. Si definisce una funzione h che associa ad ogni input in I la risposta corretta in O .

Gli algoritmi di apprendimento supervisionato hanno in comune tutti la stessa ipotesi di partenza: se all'algoritmo forniamo un numero adeguato di esempi, questi sarà in grado di calcolare una funzione h_1 che approssimerà la funzione h . Con una adeguata approssimazione, se si propongono dei dati non ancora classificati in ingresso ad h_1 , questa dovrebbe essere in grado di fornire risposte in uscita simili a quelle di h e quindi corrette o quasi; dovrebbe, quindi, approssimare in modo buono la funzione ottimale. Questi algoritmi principalmente lavorano in modo lineare, ossia assumono che per ingressi simili corrispondano uscite simili. Questo generalmente non è vero, specialmente nei casi in cui nel problema si inseriscano dinamiche caotiche dovute al tempo ma, in molti casi, questa approssimazione può dare ugualmente buoni risultati. In questo caso è facile intuire che la “bontà” dell'algoritmo è dovuta in particolar modo ai dati in ingresso: pochi dati in input non potrebbero essere sufficienti per dare una buona “istruzione” all'algoritmo, mentre troppi potrebbero renderlo troppo lento visto che la funzione h_1 potrebbe risultare molto complicata. Questi sistemi sono anche molto sensibili al rumore, quindi alcuni esempi errati potrebbero portare ad un errore in uscita. I principali algoritmi ad oggi usati sono:

- Albero di decisione⁵
- Regole di decisione⁶
- Sistemi esperti⁷

Oggi giorno la ricerca si sta concentrando su due particolari categorie di algoritmi che sono:

- Metodi generativi
- Metodi discriminativi

I primi sono basati sulla creazione di un modello dei dati che, attraverso il quale, permettono di predire le risposte desiderate. Un particolare esempio di questa famiglia di tecniche sono le reti bayesiane. I metodi discriminativi, al contrario cercano di

⁵ *Albero di decisione*: nel machine learning è un modello predittivo in cui ogni nodo rappresenta una variabile mentre gli archi sono i possibili valori per quella proprietà. Le foglie rappresentano il valore predetto per la variabile obiettivo a partire dai valori delle altre proprietà.

⁶ *Regole di decisione*: nel ramo dell'intelligenza artificiale sono delle formule logiche che permettono di associare delle scelte o dei fatti con altri.

⁷ *Sistemi esperti*: identificano una categoria di programmi informatici che, dopo una opportuna fase di apprendimento, sono in grado di dedurre nuove informazioni da un insieme di informazioni di partenza.

modellare direttamente una relazione tra dati in input con dati in output in modo di minimizzare una funzione di perdita. L'esempio principale di questo tipo di tecniche sono le Support Vector Machine (in italiano Macchine a vettori di supporto).

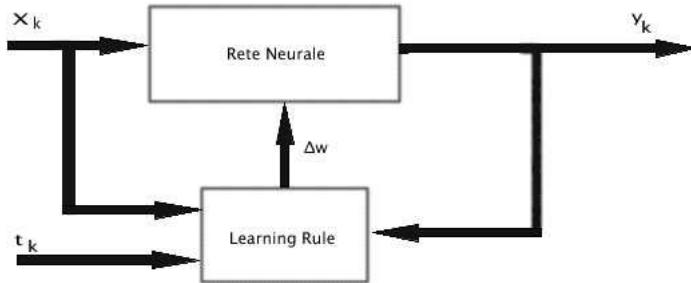


Figura 1.1. Apprendimento supervisionato

La figura 1.1 mostra lo schema base dell'apprendimento supervisionato. Possiamo notare come in questo approccio vengono “addestrati” i pesi cercando di insegnare al modello come associare le coppie (x_k, t_k) che, rispettivamente, stanno ad indicare il k-esimo dato in ingresso e la k-esima uscita corrispondente dell’insieme di dati del training set. Sempre osservando la figura possiamo vedere come l’uscita reale y_k viene confrontata con quella desiderata e i pesi vengono adattati in funzione della legge di apprendimento scelta. L’intero training set viene presentato ciclicamente per intero fino a quando non si ha che $y_k \approx t_k$.

Il ciclo di vita di questo modello lo possiamo dividere in due fasi distinte:

- Addestramento (o fase di learning o training)
- Generalizzazione (anche chiamata recall)

Durante la fase di addestramento si cerca di istruire il modello attraverso le informazioni fino ad allora conosciute, ad esempio possiamo immaginare di catalogare manualmente 100 mail secondo le nostre esigenze e quindi comunicare al sistema la nostra catalogazione. La fase di generalizzazione è invece la fase successiva, ovvero quando vengono dati nuovi ingressi e il sistema li analizzerà per associare con degli output. Riprendendo l’esempio precedente, questa fase si ha quando arrivano nella nostra inbox nuovi messaggi di posta elettronica e il sistema li cataloga automaticamente sul modello creato con la classificazione manuale.

1.4 Addestramento e conoscenza del dominio

“La potenza di un programma intelligente nel risolvere un problema dipende primariamente dalla quantità e qualità di conoscenza che possiede su tale problema”

La definizione di Feigenbaum⁸ rappresenta un dogma per tutti i sistemi esperti o anche chiamati sistemi basati sulla conoscenza. Infatti un sistema basato sulla conoscenza è un sistema in grado di risolvere problemi in un dominio limitato ma con prestazioni simili a quelle di un esperto umano del dominio stesso(Cfr. [24]). L’addestramento in un sistema esperto riveste notevole importanza. Esso infatti determina il “concept” associato alla memoria di lavoro, cioè dove sono contenuti i dati attraverso i quali vengono mantenute le conclusioni raggiunte negli instanti di tempo precedenti dal sistema. Un’errore nell’addestramento può portare a casi di non convergenza degli algoritmi utilizzati. Quindi, ogni tipo di sistema esperto deve riuscire ad esprimere due tipi di conoscenza in modo separato e modulare:

1. conoscenza sul dominio dell’applicazione
2. conoscenza su come utilizzare la storia pregressa sul dominio per risolvere i problemi.

Modellizzando quanto detto otteniamo il grafico 1.2. Gli ingressi, mediante strategie di controllo, vengono elaborati dall’apprendimento del sistema, il quale rappresenta una vera e propria interfaccia alla base di conoscenza e al motore inferenziale. Se necessario l’apprendimento classificherà i dati in gruppi e saranno poi collezionati dentro la memoria di lavoro. La classificazione avviene mediante delle regole probabilistiche: è il caso del modello Bayesiano oppure mediante degli algoritmi supervisionati come le Support Vector Machine (SVM). La memoria di lavoro servirà per la successiva classificazione.

Il progetto di un sistema di apprendimento coinvolge varie fasi. Innanzitutto si tratta di determinare il tipo di esperienza che deve essere usata per l’apprendimento. Poi si deve stabilire quale sia la funzione “target” f che deve essere appresa: infatti è utile ridurre il problema del miglioramento delle prestazioni P in un certo compito, al problema dell’apprendimento di una certa funzione. Determinata la funzione f , bisogna giungere ad una descrizione operativa della funzione target, cioè ad una descrizione che può essere usata dal nostro programma in una quantità limitata di tempo. Spesso può essere molto difficile apprendere una definizione operativa in modo esatto. Il passo successivo è quello di determinare una rappresentazione, detta modello o ipotesi, che approssimi la funzione da apprendere: ad esempio, Bayes oppure SVM.

⁸ *Mitchell Feigenbaum*: Studioso di metodi di risoluzione di quesiti fisici con strumenti matematici, è stato tra i primi a sviluppare la teoria del caos.

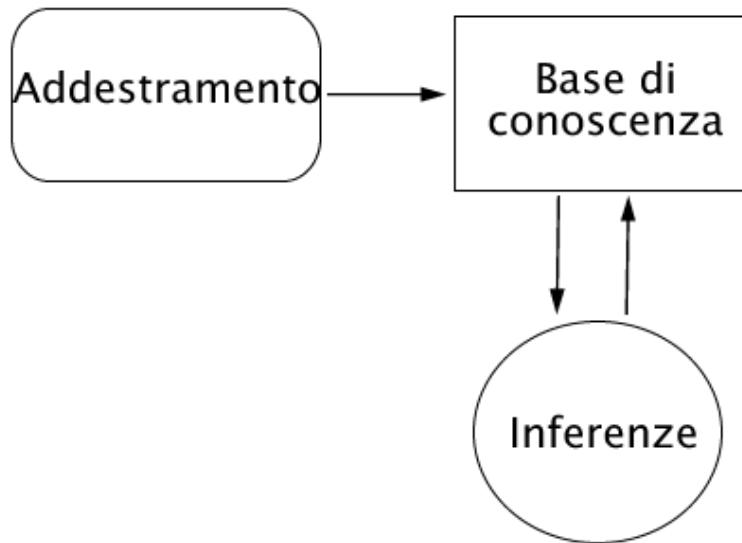


Figura 1.2. Architetture di sistemi basati sulla conoscenza

1.5 Strutturazione della tesi

Questa tesi sarà divisa in due macroblocchi: nella prima parte analizzeremo gli aspetti teorici che stanno alla base del nostro lavoro, mentre nella seconda parte daremo spazio all’analisi dello sviluppo e agli eventuali sviluppi futuri di Icetricky. In dettaglio:

Capitolo 2 : Nel secondo capitolo sarà affrontato lo stato dell’arte delle implementazioni di client di posta elettronica facendo una breve descrizione dei protocolli utilizzati e delle problematiche associate alla classificazione di messaggi di posta.

Capitolo 3 : Nel terzo capitolo sarà descritta la teoria di Bayes e l’apprendimento a reti di credenze sino a giungere alla descrizione dell’implementazione “naive” bayesiana in particolare all’applicazione di classificatori e alle catene di Markov.

Capitolo 4 : In questo capitolo saranno affrontati i principali aspetti che portano alla creazione di una Support Vector Machine. In particolar modo i metodi matematici e non, utili per una buona comprensione di questa tecnica.

Capitolo 5 : Generalizzazione del concetto di Web Ontology Language e l'astrazione del Resource Description Framework.

Capitolo 6 : Proseguendo con la descrizione del progetto e dell'architettura di sistema.

Capitolo 7 : Nel settimo capitolo saranno trattate le maggiori problematiche riscontrate nell'implementazione e nello sviluppo dell'applicativo.

Capitolo 8 : I risultati sperimentali raggiunti.

Capitolo 9 : Possibili sviluppi futuri e conclusioni.

Appendice : Descrizione di LibSvm, Tabelle, Grafici delle simulazioni, schemi relazionali della base dati, diagramma delle classi, schema dell'ontologia.

Capitolo 2

Stato dell'arte

Questo capitolo descrive lo stato attuale dell'arte, nei suoi principali aspetti, per quanto riguarda il mondo della posta elettronica. Più specificatamente nella prima parte si è analizzato lo scenario applicativo e trattato, nella seconda i programmi di posta elettronica che hanno maggiore diffusione tra gli utenti. Nell'ultima parte si è fornita una descrizione dei protocolli ad oggi esistenti, utilizzati per ricevere e spedire i propri messaggi “email”. Inoltre si è fornita una panoramica generale sulla classificazione del testo e sull'apprendimento automatico.

2.1 Scenario applicativo

Il modello Client Server, che sta diventando il modello di riferimento per il “networking” e le applicazioni di data base a qualunque livello, si contrappone come filosofia e come struttura al Mainframe.

Nel modello di computing del Mainframe è il Mainframe stesso che esegue tutte le operazioni sui dati e usa terminali (macchine senza capacità propria di prosecuzione) come unità di visualizzazione e di inserimento dati. Ad esempio ricercare e visualizzare i dati di una generica informazione su un terminale richiede le seguenti operazioni:

1. visualizzazione sul terminale di un modulo su cui l'utente può inserire l'argomento della ricerca e l'ambito ad esempio;
2. accettazione dell'informazione relativa ai dati inseriti;
3. convalida del formato dei dati;
4. verifica della disponibilità delle informazioni richieste;
5. acquisizione delle informazioni dal database;

6. formattazione dei dati per la visualizzazione sul display dell’utente
7. visualizzazione delle informazioni richieste.

Ad eccezione della presentazione dei dati, tutte queste operazioni vengono svolte dal “Mainframe”. Questa soluzione, in uno scenario distribuito di un centro stella intelligente e di tanti satelliti, comporta uno “overhead” di calcolo per il “Mainframe”. Questo richiede soluzioni architettoniche molto spinte per il centro stella. Centralizzando l’intelligenza, una caratteristica del “Mainframe” è quella di offrire garanzie per quanto riguarda la sicurezza dell’informazione e la stabilità globale del sistema. D’altro canto però, quasi nella totalità dei casi, si è legati all’azienda produttrice, in quanto si utilizza una struttura proprietaria sia a livello “hardware” che a livello “software”.

Una soluzione Client Server offre un notevole risparmio economico (il costo dei terminali oramai è simile a quello dei Personal Computer e un Server può essere realizzato su un qualsiasi PC). Inoltre si ha una maggior flessibilità nei confronti delle esigenze del mercato, soprattutto per quanto riguarda i prodotti “software”. L’affermazione di questo modello è legata inoltre ad altri fattori, quali:

1. la disponibilità sul mercato di reti locali a basso costo, permette di sostituire quasi radicalmente la soluzione centro stella con “Mainframe”;
2. l’avvento di internet e della comunicazione “broadband”, che permette di inviare grandissime quantità di dati verso i Client;
3. la capacità dei Personal Computer di essere molto performanti, garantendo una navigazione delle risorse offerte molto fluida.

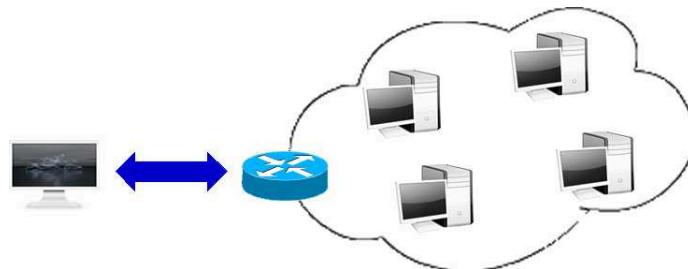


Figura 2.1. Modello multitier di interazione client server

2.1.1 Interazione Client e Server

Lo stato dell’arte delle applicazioni di posta attualmente in commercio riguarda applicazioni client server, generalmente “multi tier” soprattutto per i grandi provider. La presentazione dei dati viene demandata alle applicazioni create ad “hoc” per la visualizzazione delle mail (per esempio programmi di posta) o utilizzando i browser. Quest’ultimo, a causa della sua grande flessibilità nella gestione sia a livello “server-side”, sia in quello “client-side”, rappresenta la soluzione preferita, rispetto a quella dei classici client che risiedono n locale.

Modello distribuito di primo livello

I programmi di posta che risiedono in locale devono essere dotati di grandi capacità di calcolo in quanto si occupano di presentare i “raw data”¹. L’applicazione si deve anche occupare di gestire la comunicazione con il server, implementando così totalmente il protocollo richiesto. Talvolta i dati possono essere organizzati in “cluster”², archiviando informazioni che possono essere così facilmente reperibili (per esempio l’utilizzo del protocollo POP3 nel “download” delle mail in locale). Il server, in questo modo, svolge funzioni di MTA³ e inoltra le risposte al client.

Modello distribuito del secondo livello

Questo è un modello utilizzato per l’esposizione dei servizi mail su web da parte di un “provider”. Tutto il lavoro di gestione dell’account e la presentazione dei dati è demandata al server, che si fa carico di un lavoro aggiuntivo, oltre al classico inoltro delle mail con i vari MTA. Questo paradigma rappresenta quasi un paradosso nell’evoluzione dei servizi client server, in quanto è come tornare dietro al modello “Mainframe”, ma questa soluzione è maggiormente apprezzata dai “provider” per i motivi citati nel cap. 2. Il client generalmente utilizza un visualizzatore HTML, un “browser”.

Modello distribuito del terzo livello

Unisce entrambi i modelli analizzati sopra, cercando di spostare verso il client una parte di controllo sui dati e visualizzazione delle informazioni. Il server, in tal modo, si occupa di gestire l’inoltro delle mail, da e per altri MTA e fornisce una

¹ *raw data*: termine che si usa per indicare i dati grezzi, senza alcuna formattazione.

² *cluster*: generalmente indica in ambito informatico un insieme di dati.

³ *MTA*: Mail Transfer Agent, è una macchina che svolge le funzioni di inoltro e ricezione delle mail.



Figura 2.2. Modello di computing distribuito I livello

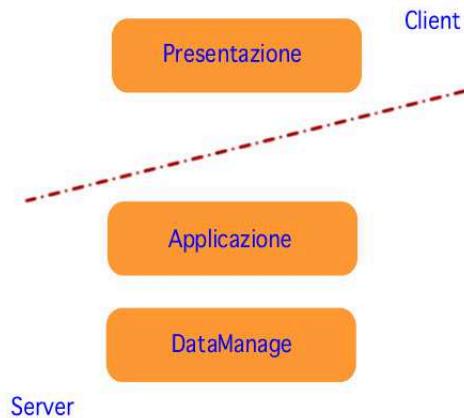


Figura 2.3. Modello di computing distribuito II livello

prima formattazione dei dati. Si alleggerisce il lavoro del centro stella. Questo rappresenta lo scenario più utilizzato nell’esporre la posta su internet.

2.1.2 Modello WebService

I Web Service forniscono uno standard per applicazioni software che lavorano in maniera interoperabile su una grande varietà di piattaforme e infrastrutture. Caratteristica fondamentale di un Web Service è quella di offrire un’interfaccia software

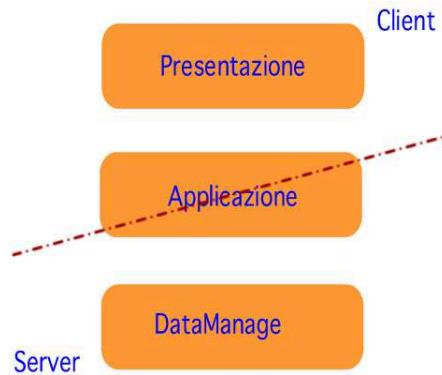


Figura 2.4. Modello di computing distribuito III livello

(descritta in un formato automaticamente elaborabile quale, ad esempio, il WSDL⁴) utilizzando la quale altri sistemi possono interagire con il Web Service stesso attivando le operazioni descritte nell’interfaccia tramite appositi “messaggi” imbustati in una risposta SOAP⁵: tali messaggi sono, solitamente, trasportati tramite il protocollo HTTP (non è comunque una prerogativa l’utilizzo di questo protocollo) e formattati secondo lo standard XML.

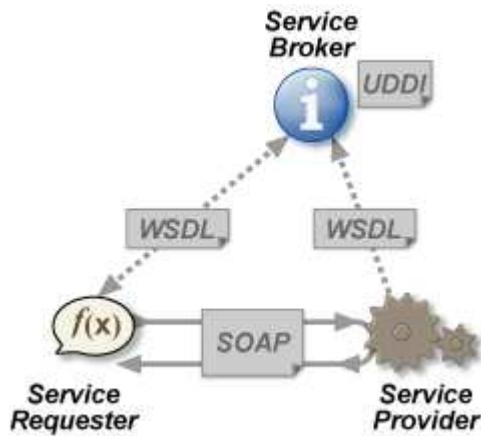


Figura 2.5. Protocolli di comunicazione Client Web Service[32]

⁴ *WSDL*: Web Services Description Language, linguaggio formale in formato XML utilizzato per la creazione di “documenti” per la descrizione di Web Service.

⁵ *SOAP*: Simple Object Access Protocol, è un linguaggio creato per lo scambio di informazioni legato agli oggetti.

A differenza dei modelli basati sul Web, il Web Service non è direttamente dipendente alla UI⁶, questo “disaccoppiamento” rende più flessibile le modifiche ad uno o all’altra parte dell’applicazione, potendo operare in completa trasparenza. A maggior ragione, permette lo sviluppo di progetti modulari, i quali possono “crescere” con il tempo e rende possibile poter riutilizzare applicazioni già esistenti.

Nella fig. 2.1.2 possiamo notare i diversi ruoli che vengono forniti dagli attori di una classica interazione Client - Web Service:

1. Client, generalmente utilizza un browser, link [1] nella figura in questione, ed effettua una richiesta al Server Web, il quale formatta i dati in funzione della base dati e li invia al client. Può anche utilizzare un programma che faccia richieste SOAP direttamente, link [2] nella figura.
2. Server Web, contenitore di informazioni statiche, come ad esempio pagine HTML. Talvolta, il suo ruolo è anche quello di proxy⁷, questo perché i moderni browser tendono ad impedire redirezioni da parte dei server verso altre macchine esterne alla rete. Esso è il contenitore della UI.
3. Web Service, è generalmente il motore dell’applicazione ed espone le chiamate ai servizi offerti.

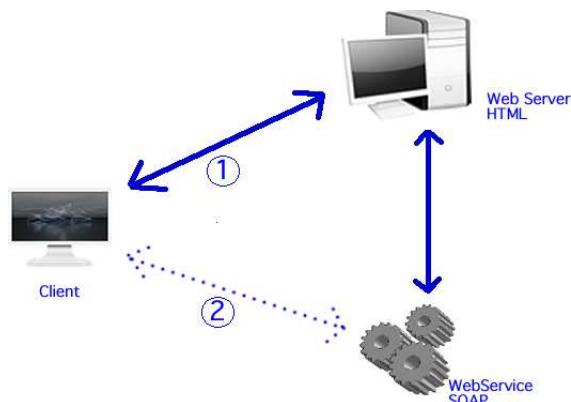


Figura 2.6. Interazione client Web Service

Il protocollo HTTP, “over” TCP sulla porta 80, ha riscosso notevoli consensi da parte dei sistemisti, in quanto non richiede modifiche sulle regole di filtraggio dei firewall.

⁶UI: User Interface.

⁷proxy: Svolge la funzione di Server nel confronto dei client, da cui ricevono le richieste, e di client nei confronti dei server a cui inoltrano le richieste dei client stessi. A causa del loro lavoro spesso rappresentano dei “colli di bottiglia” per i nodi di rete.

Questo è un aspetto che se da un lato è positivo, dall'altro suscita preoccupazioni per ragioni essenzialmente di sicurezza.

2.2 Client di posta elettronica

In questa sezione descriviamo i programmi più utilizzati dagli utenti per consultare la propria posta elettronica sia online, che offline. Tali client email hanno come funzionalità base la possibilità di collegarsi alla casella email dell'utente e prelevare i messaggi di posta dal server (cancellandoli o meno) in modo che l'utente possa in seguito, anche se non connesso in rete, consultare le email che precedentemente ha scaricato sul proprio elaboratore. Ci limiteremo a trattare le caratteristiche di questi client strettamente al caso di classificazione delle mail, mediante tag o cartelle e nell'analisi dei loro filtri per la ricerca di “Spam and Ham”.

2.2.1 Mozilla Thunderbird

Mozilla Thunderbird è un client di posta elettronica e news, in grado di gestire i feed RSS ed i Newsgroup, sviluppato da Mozilla Foundation ed è distribuito come software libero. La versione 1.0 è stata rilasciata il 7 dicembre 2004. Attualmente viene distribuita come stabile la versione 2.0, rilasciata il 18 aprile 2007.

Nativamente, alla versione attuale, la 2.0, permette di effettuare una prima selezione delle mail in entrata, in junk (spam) e non junk. L'apprendimento di posta indesiderata viene guidato dal motore stesso di thunderbird, da esempi negativi o da ricorrenze di parole che sono considerate “pericolose”. Sostanzialmente vi è un'applicazione dei filtri bayesiani. Thunderbird impara rapidamente i gusti: dopo qualche decina di messaggi (questa costituisce la base di conoscenza del motore bayesiano) distingue automaticamente fra Spam e messaggi desiderati con abbastanza precisione (limitata dai livelli del filtro applicato). A quel punto, si può dire di releggere in una cartella o nel cestino i messaggi che identifica come Spam. Nel giro di qualche giorno, la posta indesiderata verrà eliminata. Generalmente lo spam si compone di mail scritte in html, ricche di immagini di ogni tipo. Questo client permette di visualizzare il corpo della mail solo in text plain se desiderato ed impedisce la visualizzazione di immagini.

L'organizzazione delle mail può essere gestita mediante cartelle gerarchiche, mentre ad ogni mail è possibile associare dei Tag, nei quali esprimiamo il tipo di mail ricevuta, ad esempio “Lavoro”, “Importante”. Nonostante questa sia una tecnica utilizzata da anni, l'archiviazione di mail mediante cartelle può portare a problemi del tipo creazione di cartelle “Da Archiviare” o simili, impedendo in tal modo la possibilità di creare insiemi di mail che possono avere intersezioni. Il classico esempio è una mail che viene mandata da un collega di lavoro, ma che si riferisce ad un

appuntamento serale. Dove inseriamo la mail?

In aggiunta thunderbird supporta l’utilizzo di filtri. Nonostanti siano molto comodi da impostare, non risolvono il problema sopra citato. Infatti si basano su analisi degli “headers” (From, To, Cc, etc.) o del body della mail mediante strumenti del tipo:

Contiene una parola o un riferimento al campo selezionato;

Non Contiene l’espressione.

È filtra in base ad una occorrenza ben definita il campo considerato.

Non è il negato di quanto detto sopra.

Inizia con.

Finisce con.

A questi strumenti si aggiungono quelli addizionali, che possono essere utilizzati mediante l’aggiunta di Componenti Aggiuntivi. Tra essi, possano citare il tool *MailClassifier*. Questo strumento utilizza dei filtri bayesiani per classificare le mail, ma la classificazione si basa solo sulla presenza di cartelle. Quest’ultime costituiscono la categoria da associare al motore inferenziale. Nonostante sia un vantaggio poiché estende l’utilizzo del filtro junk e non junk, rappresenta comunque una limitazione in quanto si fa uso di una stretta struttura gerarchica ottenuta dalle cartelle.

2.2.2 Apple Mail

È un client di posta elettronica e news, in grado di gestire i feed RSS ed i News-group, sviluppato da Apple Foundation. Presenta numerose caratteristiche, tra cui facilità d’uso e uno stile curato, che lo rendono uno dei più apprezzati sul mercato. Gestisce, come avviene anche per Thunderbird, lo spam mediante un filtro di tipo bayesiano, in grado di imparare dagli usi dell’utilizzatore.

Importante novità rispetto a Thunderbird è la gestione delle conversazioni. Esse vengono riconosciute basandosi sulla risposta ad un messaggio inviato, analizzando il campo di *InReplyTo*. In aggiunta utilizza anche il *Subject* per la costruzione degli stessi. Infatti, se Mail riceve nella sua inbox, mail che hanno lo stesso *Subject*, crea una conversazione. La creazione della conversazione stessa, però, non è molto flessibile, infatti non è permesso lo spostamento della mail dalla conversazione, se ad esempio il classificatore di Mail ha commesso un errore.

A queste caratteristiche, si aggiunge la possibilità di “taggare” le mail. La classificazione, comunque, viene demandata all’utilizzatore. Presenta, probabilmente per compatibilità con i vecchi client, l’organizzazione gerarchica delle mail per cartelle.

2.2.3 Novell Evolution

Evolution o **Novell Evolution** è il personal information manager e workgroup information management ufficiale per l’ambiente desktop GNOME. Prima dell’acquisizione di Novell era denominato **Ximian Evolution**. Questo software include le funzionalità di e-mail, calendario, rubrica e task list. È stato incluso ufficialmente in GNOME a partire dalla versione 2.8. Le sue funzionalità sono molto simili a *Microsoft Outlook*, ma in più contiene alcune funzionalità che lo contraddistinguono come ad esempio il supporto ad iCalendar, indice di tutte le mail in ricezione, e le cartelle virtuali, ossia delle cartelle speciali create virtualmente sulla base di una ricerca effettuata dall’utente.

Come per Thunderbird, vi è un’implementazione di filtri bayesiani per la ricerca della posta indesiderata. Il livello di precisione del filtro cresce con l’aumentare di mail considerate “Spam” (dalla teoria di Bayes, maggiore è il training set e maggiore sarà la fedeltà del risultato). Si può associare la scelta di relegare direttamente la mail nel cestino o in qualsiasi cartella creata “ad hoc”.

La classificazione delle mail, oltre al caso di posta indesiderata, viene gestita mediante cartelle, quindi soffre dei limiti di questa categorizzazione. Per rendere più automatica l’associazione della mail ad una categoria, sono implementati dei filtri con i quali possiamo suggerire al client dove consegnare i messaggi ricevuti. Estende l’utilizzo del filtro di Thunderbird, con l’utilizzo di etichette “Lavoro”, “Importante” e con:

1. Non comincia con.
2. Non termina con.
3. Suona come.
4. Non suona come.

Queste caratteristiche aggiuntive permettono di rendere più precisa la scelta effettuata. È anche possibile esprimere molteplici regole in cascata.

2.2.4 Microsoft Outlook

Microsoft Outlook è un software della Microsoft Corporation ed è incluso nella suite Microsoft Office. Le sue funzionalità sono molteplici, infatti oltre da fungere da client di posta elettronica contiene un calendario, una agenda per la memorizzazione delle attività, un raccoglitore di note, un diario, una lista contatti.

Anche Outlook espone un motore interno che permette di fare una prima selezione di mail, “spam” e “not spam”. Questo è garantito per mezzo della tecnologia Microsoft

SmartScreen. Questa è basata sull’apprendimento bayesiano di un filtro, calcolando la probabilità che una mail sia “junk” o invece sia “non spam”. SmartScreen apprende come distinguere le mail “buone” da quelle “cattive” imparando dalle decisioni dell’utilizzatore. Vi è anche la possibilità di esprimere l’insieme di mittenti “safe” e quelli che devono essere considerati come poco attendibili. Con la versione 2007, è stata aggiunta la segnalazione di “phishing”.

La gestione di classificazione è demandata all’utente, con la creazione di cartelle gerarchiche le quali hanno i difetti rappresentati sopra.

2.2.5 Eudora

Eudora è uno dei primi programmi di posta elettronica creati e resi pubblici gratuitamente. La prima versione risale al 1988 e fu scritta da Steve Dorner. Fu anche il primo programma di questo tipo ad avere un’interfaccia grafica. Nel 1991 il prodotto fu acquistato da Qualcomm. Per diversi anni Eudora è stato uno tra i programmi di posta elettronica più diffusi sia per Mac OS che per Microsoft Windows. L’11 ottobre 2006, Qualcomm ha annunciato che le future versioni di Eudora saranno basate sulla stessa piattaforma tecnologica di Mozilla Thunderbird e sarà “open source”. Il nome di questo client deriva dal nome di una scrittrice statunitense, la quale scrisse un libro intitolato: “Perché vivo in un ufficio postale”.

Questo client prevede l’utilizzo di filtri addizionali, i quali, come per i client sopra, permettono di identificare le mail da scartare, quelle indesiderate, mediante l’utilizzo di filtri bayesiani. La classificazione rimane comunque legata alle cartelle.

2.3 Web o Web Service Mail

Con l’avvento del mondo di internet e della comunicazione, molti “provider” di posta hanno scelto di utilizzare esclusivamente e non esclusivamente il servizio di posta solo sul web. Ci sono varie ragioni dietro questa scelta, per motivi pubblicitari, ad esempio, perché è più facile inserire degli spot su una portale piuttosto che nel testo di una mail. Comunque va anche citato il problema dell’amministrazione di “firewall” di reti che espongono un servizio di posta. Possiamo anche dire che il mondo del web è diventato conosciuto dal pubblico e esporre un servizio su web facilita l’utilizzo del consumatore, il quale non ha costi di apprendimento per capire come usare un nuovo strumento.

Tra i “provider” maggiori possiamo citare Gmail (la Google Mail) e Hotmail. Quest’ultima, per la gestione dello spam e l’archiviazione delle mail, utilizza gli strumenti citati in Outlook a livello di web service. Un discorso a parte va fatto, invece, per Gmail, la quale allo stato attuale dell’arte rappresenta un’ottima, se non la migliore, soluzione mail a livello mondiale.

2.3.1 Gmail

I vantaggi dell’uso di Gmail rispetto a provider concorrenti sono molteplici. Il team Gmail ha previsto l’utilizzo di strumenti contro lo spam, adottando una soluzione in più livelli:

- White list, o anche detta lista di indirizzi di provider fidati, i quali possono ricevere ed inviare mail al dominio Gmail, comunque vengono elaborate per essere considerate attendibili, mediante l’uso di filtri.
- Black list, o anche detta lista di indirizzi di provider che non possono essere considerati attendibili: tutte le mail inviate da questi sono completamente scartate.
- Le mail inviate da uri di provider che non si trovano nei due insiemi sono soggette all’analisi di filtri bayesiani che valutano il livello di attendibilità.

Una novità sostanziale rispetto a soluzioni passate è l’utilizzo di etichette per la gestione delle mail. Scompare in tal modo il concetto di cartelle. Le etichette funzionano esattamente come le cartelle, ma presentano in aggiunta una caratteristica speciale: permettono di applicare più di un’etichetta a una conversazione. Una volta creata un’etichetta, si può visualizzare tutti i messaggi con quella etichetta facendo una ricerca oppure selezionando il nome della stessa.

Per rendere automattizzato l’associazione di una data mail ad un’etichetta, sono stati creati dei filtri, i quali lavorano in base alla presenza di date parole dentro il corpo della mail o dentro il suo header. Questo rappresenta comunque una forte limitazione poiché si applica solo in circostanze di “Contiene”, “Non contiene” o un’associazione biunivoca con alcuni campi dell’header come “Oggetto”, “Da”, “A”.

Un’altra novità che il team Gmail ha introdotto è il concetto di flusso di discorso. Questo è esteso solo al caso di “Re:” o “reply” oltre che al caso di “mailinglist”. Pertanto tutti i messaggi mandati in quella modalità verranno rappresentati sotto un’unico flusso di discussione. Questo è garantito dall’unione a livello logico dei messaggi inviati da quelli in ingresso. Il web service di gmail fornisce una modalità di riferimento tra i due tipi di dati, creando così una sinergia tra i messaggi mediante una rappresentazione che si avvicina molto alla rappresentazione umana di una discussione.

A queste vanno aggiunti nuovi algoritmi di ricerca delle mail e la possibilità di controllare in tempo reale lo stato di un contatto, utilizzando il protocollo XMPP, comunque non argomento di questa tesi.

2.4 Protocolli

In questa sezione si analizzeranno i protocolli utilizzati fino ad oggi riguardanti la posta elettronica.

2.4.1 POP - Post Office Protocol

Il Post Office Protocol è un protocollo che è stato pensato e realizzato per permettere di accedere mediante autenticazione a un account di posta elettronica, presente su un host remoto, per scaricare le e-mail presenti. Il demone del POP, per quanto riguarda la versione 3, di norma è attivo sulla porta 110 TCP. I messaggi di posta elettronica, per essere letti, devono essere prelevati dal server. Come si vedrà in seguito, questa è una sostanziale differenza rispetto all’IMAP.

Il protocollo POP3 è stato pensato senza nessuna forma di crittografia e quindi tutte le password viaggiano sulla rete in chiaro. Per ovviare a ciò è stata pensata un’estensione che risolve questo problema utilizzando l’algoritmo MD5, il nome del protocollo è quindi APOP.

Le caratteristiche del POP3 permettono a utenti con connessioni intermittenti (come i collegamenti dial-up) di recuperare le e-mail quando sono connessi e possono in seguito consultarle senza la necessità di essere collegati alla rete. I client di posta che utilizzano POP3, normalmente, scaricano i messaggi sulla postazione dell’utente e cancellano i messaggi dal server per default, ma esiste la possibilità di lasciare una copia delle email sul server. Quando si scaricano i messaggi utilizzando l’opzione di lasciare i messaggi sul server, generalmente, si utilizza il comando POP3 UIDL (**Unique IDentification Listing**). La maggior parte dei comandi del protocollo identificano i singoli messaggi attraverso un numero che specifica l’ordine dei messaggi sul mail server. Questo crea un problema per i client che vogliono lasciare i messaggi sul server poiché il numero di un messaggio può cambiare da una connessione a un’altra. Per esempio, se una mailbox contiene cinque messaggi l’ultima volta che ci si è connessi, e se un client diverso cancella il messaggio #3, alla nuova connessione il client troverà gli ultimi due messaggi number decrementati di uno. UIDL è una soluzione per ovviare a questo tipo di problema di numerazione dei messaggi. Il server assegna una stringa di caratteri che sarà un ID permanente e univoco del messaggio. Quando un client POP3-compatibile accede al server, può usare il comando UIDL per recuperare l’attuale mapping tra gli ID dei messaggi e i message-number. Il client può utilizzare questo per determinare quali messaggi sono già stati scaricati e quali invece sono ancora da prelevare.

Nella tabella 2.1 riportiamo un esempio di dialogo tra POP3 client e server.

C:	telnet pop3.example.com 110
S:	+OK <22593.1129980067@example.com>
S:	...
C:	QUIT
S:	+OK

Tabella 2.1. Parte di una comunicazione con un POP3 server, l’esempio completo è riportato in appendice B.1

2.4.2 IMAP

IMAP è un protocollo di comunicazione per la ricezione di email. La sigla fino alla versione 3 del protocollo stava ad indicare *Interactive Mail Access Protocol* però dalla quarta versione è stato modificato in Internet Message Access Protocol. Attualmente siamo alla versione 4 revision 1(Cfr. [2]). Tale protocollo è stato pensato nel 1985 da Mark Crispin⁸. Esso nasce come una alternativa moderna all’utilizzatissimo POP. Entrambi i protocolli permettono di accedere, leggere e cancellare i messaggi di posta elettronica da un server, ma hanno alcune differenze. Nel seguito descriviamo le principali novità introdotte dall’IMAP rispetto al POP.

Con il protocollo IMAP più utenti posso utilizzare la stessa “mailbox”, permette, quindi, connessioni simultanee fornendo dei meccanismi per controllare i cambiamenti apportati da ogni singolo utente. Nel POP invece si assume che una casella di posta sia univocamente di un utente.

Il protocollo fornisce anche supporto all’accesso a singole parti MIME di un messaggio. La maggior parte delle email sono realizzate nel formato MIME, che realizza una struttura ad albero del messaggio in cui ogni ramo rappresenta un contenuto diverso (testo, allegati, immagini, ecc...). L’IMAPv4 permette di scaricare anche solo una singola parte del MIME o addirittura un frammento di una parte, in modo da avere una anteprima della mail prima di scaricarla oppure per prelevarla senza allegati.

Inoltre è presente il supporto per degli attributi dei messaggi conservati sul server. Attraverso l’utilizzo di questi attributi, definiti dal protocollo, ogni client può tenere traccia di ogni messaggio e sapere se è già stato letto oppure se ha già avuto risposta o meno, ad esempio.

⁸Mark Crispin: lavora alla University of Washington, è conosciuto come l’inventore di IMAP. È autore e co-autore di numerose RFCs; è il principale artefice di UW IMAP, una delle implementazioni di riferimento di IMAP4rev1 descritta nella RFC 3501.

L’IMAP permette agli utenti altresì di creare, cancellare e modificare diverse “mailbox”, che corrispondono normalmente a delle cartelle sul server. Inoltre, con questa gestione delle ‘mailbox’, permette di avere cartelle condivise tra utenti diversi.

Grazie all’IMAP è anche possibile fare delle ricerche sui messaggi. Si può quindi chiedere al server quali messaggi soddisfano certi criteri e scaricare quindi solo quelli che interessano in quel momento, evitando quindi di dover recuperare tutti i messaggi presenti nella propria mailbox.

Il protocollo definisce anche la possibilità per la definizione di estensioni da parte del server. Nelle specifiche è descritto come un server può comunicare agli utenti di avere ulteriori funzionalità rispetto a quelle di base.

Infine l’IMAP ha una gestione migliore delle password rispetto al POP. Infatti in quest’ultimo, solitamente, le password vengono inviate in chiaro rendendo quindi molto facile a un malintenzionato il recupero delle “password” che lo interessano. Con l’IMAP è quindi possibile cifrare le “password”, ma server e client devono trovare una politica di sicurezza comune per poter comunicare tra di loro.

L’IMAP è principalmente utilizzato nelle grandi reti, come ad esempio le università o le aziende, luoghi in cui l’utente cambia spesso postazione. Con il POP sarebbe necessario tutte le volte che si cambia PC riscaricare tutti messaggi, mentre con l’IMAP è possibile scaricare solo i nuovi messaggi o quelli di proprio interesse.

La porta standard associata all’IMAP è la 143, nel caso in cui la connessione usa SSL la porta associata è la 993. Nella tabella 2.2 mostriamo un esempio di comunicazione con il server imap.

C:	telnet imap.example.com 143
S:	* OK Dovecot ready.
C:	* 001 login biagio@patrocloo.it password
S:	...
C:	004 logout
S:	* BYE Logging out 004 OK Logout completed. +OK

Tabella 2.2. Parte di una comunicazione con un IMAP server, l’esempio completo è riportato in appendice B.2

2.4.3 SMTP

La sigla SMTP è l’abbreviazione di Simple Mail Transfer Protocol ed indica il protocollo standard per la trasmissione di email attraverso la rete. Formalmente è

descritto nella RFC 821, il protocollo ad oggi utilizzato si chiama ESMTP (Extended SMTP)⁹.

L’SMTP è un protocollo relativamente semplice, testuale, dove vengono specificati uno o più destinatari e il messaggio viene inoltrato a questi ultimi. Normalmente il protocollo, prima della spedizione, verifica l’esistenza degli indirizzi e-mail specificati. È un protocollo che segue il modello client-server, dove un client trasmette un messaggio email al server. Un client email tipicamente conosce il server SMTP di inoltro dalla sua configurazione. Un relaying server normalmente determina a quale server SMTP occorre connettersi per trovare il MX¹⁰ DNS per il dominio¹¹ di ogni destinatario. Alcuni MTA¹² possono anche utilizzare l’SVR record, più generale dell’MX, ma questo non è molto usato (i relaying server dovrebbero essere configurati per l’utilizzo di uno smart host).

La porta di accesso di SMTP è la 25 TCP e si può facilmente verificare e comprendere il modo di funzionamento del protocollo utilizzando il telnet.

SMTP fu sviluppato su Mail Box Protocol¹³, FTP Mail¹⁴ e Mail Protocol¹⁵. Il lavoro di sviluppo continuò negli anni settanta, fino a quando ARPANET lo inglobò nei modem internet intorno agli anni novanta. SMTP iniziò ad essere utilizzato agli inizi degli anni ottanta. In quegli anni era largamente utilizzato UUCP¹⁶ che era più adatto a trasferire messaggi di posta elettronica tra postazioni con una connessione intermittente. Viceversa SMTP si adattava meglio quando i computer disponevano di una connessione permanente alla rete. Entrambi utilizzano una tecnica Store and Forward.

Sendmail fu uno dei primi (se non proprio il primo) ad implementare SMTP. Fino ad oggi sono stati scritti numerosi programmi che implementano il protocollo come client o come server. Tra i più famosi ricordiamo: Exim di Philip Hazel, Postfix di Wietse Venema¹⁷, qmail di D. J. Bernstein, Courier di Sam Varshavchik e Microsoft Exchange Server dell’omonima casa americana.

Poiché SMTP è un protocollo testuale basato sulla codifica ASCII, non è permesso

⁹Definito nella RFC 2821.

¹⁰Mail eXchange

¹¹La parte a destra del simbolo @ nell’indirizzo e-mail.

¹²Mail Transfer Agent

¹³circa 1971.

¹⁴circa 1973. RFC 469

¹⁵RFC 524.

¹⁶Unix to Unix CoPy.

¹⁷<http://www.porcupine.org/wietse/>

trasmettere direttamente testo composto con un diverso gruppo di caratteri e tantomeno file binari. Lo standard MIME permette di estendere il formato dei messaggi mantenendo la compatibilità col software esistente. Per esempio, oggi molti server SMTP supportano l’estensione 8BITMIME, la quale permette un trasferimento di un testo che contiene caratteri accentati (non-ASCII) senza bisogno di trascodificarlo. Altri limiti di SMTP, quale la lunghezza massima di una riga, impediscono la spedizione di file binari senza trascodifica. (Da notare che per i file binari inviati con HTTP si utilizza il formato MIME senza bisogno di una trascodifica).

L’SMTP è un protocollo che permette soltanto di inviare messaggi di posta, ma non di richiederli ad un server: per fare questo il client di posta deve usare altri protocolli, quali il POP3 o l’IMAP, descritti in precedenza.

Nella tabella 2.3 riportiamo un esempio di dialogo con un server SMTP in seguito al comando telnet www.example.com 25:

S:	220 smtp.example.com ESMTP Postfix
C:	HELO mydomain.com
S:	250 Hello mydomain.com
C:	...
S:	2 21 Bye

Tabella 2.3. Parte di una comunicazione con un SMTP server, l’esempio completo è riportato in appendice B.3

2.5 Regole di classificazione delle mail

Come espresso nei paragrafi precedenti, la maggioranza dei sistemi di riconoscimento attuali delle mail si basano su assunzioni di tipo “keyword-spotting” (Cfr. [7]). In un sistema basato su questa assunzione, la condizione primitiva verifica se una parola appare (o non appare) in un dato campo di un messaggio di posta. Un esempio di regola basata su “keyword-spotting” è la seguente:

$$\begin{aligned}
 amici &\leftarrow giuseppe \in from, rizzo \in from \\
 amici &\leftarrow biagio \in from, meirone \in from \\
 amici &\leftarrow calcio \in body, partita \in body
 \end{aligned} \tag{2.1}$$

Tramite questo insieme di regole, i messaggi di posta che presentano gli elementi “giuseppe” o “biagio” nel mittente vengono etichettati come appartenenti alla categoria *amici*, così come per i messaggi che presentano nel “body” i termini “calcio”

o “partita”. Benché questo metodo di classificazione ha il vantaggio di essere di immediata comprensione all’utente e di veloce implementazione, ha il difetto di essere poco flessibile in scenari con una forte presenza di classi e di multiclassificazione. Su questo si basano gli attuali filtri di posta dei maggiori “client”. Il paragrafo successivo è dedicato ai problemi di riconoscimento dello “spam”, che si basano sulla teoria di Bayes o su algoritmi come il “TD-IDF”¹⁸.

2.6 AntiSpam

L’invio di grandi quantità di informazioni non richieste a scopo commerciale ha avuto un notevole incremento con il passare degli anni. Per ovviare a questo problema, gli amministratori hanno adottato le tecniche più svariate, utilizzando sia quelle euristiche, che quelle statistiche. Tra le ultime, quella degna di nota, come visto in precedenza, prevede l’uso di filtri bayesiani. Una prima proposta di filtro che utilizzasse queste funzioni, è stata fatta da Paul Graham, presentando nel 2002, al mondo del web, un articolo con il quale spiegava la sua implementazione. Il suo filtro, scritto in perl, presentava degli ottimi risultati a regime, con una base di conoscenza, ossia di esempi negativi e positivi, di un centinaio di mail.

Fu una scoperta molto importante, infatti tutti i maggiori sistemi che combattono lo spam prevedono l’utilizzo di filtri di Bayes. Degni di nota sono “Spam Assassin” e “Bogo Filter”. Utilizzando una struttura molto simile a quella proposta da Graham, prevedono un funzionamento a regime che praticamente presenta dei falsi positivi e dei positivi negativi che è molto prossimo allo zero, con un “training set” elevato. Entrambi sono soluzioni che possono essere sia lato client, che lato server.

Graham nel corso degli anni ha esteso il progetto di semplice filtro “Spam and Ham”, al caso multi-classe. Questo progetto prende il nome di “Pop File”. Quest’ultimo è un programma, lato client, per la classificazione automatica della posta. Dopo essere stato istruito mediante degli esempi, sarà in grado di controllare tutte le email in arrivo e classificarle. Può essere utilizzato sia per un compito semplice, tipo il separare lo spam dalle email normali, che per compiti più complessi tipo la categorizzazione dell'email in dozzine di cartelle. Lo si può vedere come un assistente personale per la posta in arrivo. Questo strumento, come per il caso del “Mail Classifier” proposto per Thunderbird, soffre delle limitazioni dovute alla classificazione in cartelle.

¹⁸ *TF-IDF*: “term frequency-inverse document frequency”, un algoritmo basato sulla costruzione di un vettore di occorrenze positive ed uno di occorrenze negative, la cui differenza determina le “feature” della categoria.

Capitolo 3

Il teorema di Bayes

La legge di Bayes è un fondamento nella teoria della probabilità, essa mette in relazione distribuzioni di probabilità condizionali e marginali di variabili casuali. Il teorema di Bayes rappresenta una rivoluzione nei confronti delle credenze passate della probabilità, in quanto si contrappone al metodo classico basato sulle prove ripetute.

Prendendo come riferimento due eventi stocastici A e B, il teorema di Bayes definisce:

$$P(A|B) = \frac{P(B|A)*P(A)}{P(B)} \quad (3.1)$$

Con:

- $P(A)$ è la probabilità a priori o la probabilità marginale di A. Definita a priori poiché non è influenzata in nessun modo dall'evento B.
- $P(B)$ è la probabilità a priori dell'evento B, normalizza il rapporto e agisce da costante se $P(B)$ è a densità uniforme.
- $P(B|A)$ è la probabilità condizionata di B dato A. È detta funzione di verosimiglianza, ed è ciò su cui si fonda l'inferenza classica, o frequentista.
- $P(A|B)$ è la probabilità condizionata di A dato B. Essa è definita probabilità a posteriori poiché è influenzata dal valore specifico di B.

Il teorema definisce la probabilità a posteriori come direttamente proporzionale alla probabilità a priori ed una variabile, chiamata speranza, ed inversamente proporzionale alla costante di normalizzazione.

$$P(A|B) \propto L(A|B)*P(A) \quad (3.2)$$

Considerando:

- $L(A|B)$ la verosomiglianza di A dato un valore fissato di B.

3.1 Apprendimento Bayesiano

L’approccio bayesiano alla probabilità può essere esteso al problema dell’apprendimento: da questa unione scaturisce una teoria estremamente potente che fornisce una soluzione generale ai problemi di rumore, sovraddestramento e previsione ottima(Cfr. [24]).

L’apprendimento bayesiano si pone come obiettivo il problema di fare delle previsioni e ritiene il problema della formulazione di ipotesi a partire dai dati, come un suo sottoproblema. L’idea è di utilizzare le ipotesi come intermediario tra dati e previsioni. La probabilità delle ipotesi è calcolata dato i dati. Le previsioni sono date dalle ipotesi, usando la probabilità a posteriori delle ipotesi per calcolare le previsioni.

Si prenda per esempio il problema dello spam, creando due categorie: *spam* categoria A e *ham* B. Piuttosto che scegliere una delle due categorie, darà una misura di probabilità di entrambe. La scelta verrà infatti presa in funzione della conoscenza regressa e della verosomiglianza delle informazioni in ingresso con la sua base di conoscenza.

In modo più formale, sia D un insieme di dati, sia X una quantità ignota su cui vogliamo fare delle previsioni e siano H_1, \dots, H_n delle ipotesi; l’apprendimento bayesiano assume la forma:

$$\begin{aligned} P(X|D) &= P(X|D, H_i) * P(H_i|D) \\ &= P(X|H_i) * P(H_i|D) \end{aligned} \quad (3.3)$$

Si può dimostrare che nessun altro metodo di predizione, che fa uso dello stesso spazio di ipotesi e della stessa conoscenza a priori, ha in media delle prestazioni migliori(Cfr. [21]).

L’apprendimento Bayesiano richiede il calcolo di $P(H_i|D)$ per tutti gli H_i , questo comporta problemi molto complessi ed in generale non è il miglior modo di calcolare le previsioni. Si fanno allora delle ipotesi semplificative tra cui la più comune è quella di considerare solo l’ipotesi più probabile, ovvero di considerare solo l’ipotesi H_i che massimizza $P(H_i|D)$. Questa ipotesi viene chiamata massima a posteriori o MAP (maximum a posteriori) e si indica con H_{MAP} :

$$P(X|D) \approx P(X|H_{MAP}) \quad (3.4)$$

Il problema è definire H_{MAP} . Applicando la regola di Bayes possiamo calcolare $P(H_i|D)$ come segue:

$$P(H_i|D) = \frac{P(D|H_i)*P(H_i)}{P(D)} \quad (3.5)$$

Il termine $P(D|H_i)/P(D)$ rappresenta la verosimiglianza (likelihood) dei dati, assunta l’ipotesi H_i come quella corretta. La $P(H_i)$ è la probabilità a priori (prior)

dell’ipotesi e riflette la nostra conoscenza di fondo sulla probabilità che H_i sia quella corretta prima di aver osservato i dati; se non disponiamo di nessuna conoscenza, si può assegnare un “prior” uniforme a tutte le ipotesi. Il denominatore $P(D)$ rappresenta la probabilità dei dati D senza nessuna conoscenza su quale ipotesi possa valere; essendo una costante di normalizzazione, non influisce nella precedente massimizzazione. Infine il termine $P(H_i|D)$ è la probabilità a posteriori dell’ipotesi che combina insieme la verosimiglianza dei dati e la conoscenza a priori sull’ipotesi e riflette la nostra confidenza che valga l’ipotesi H_i dopo aver osservato i dati. Da notare che la probabilità a posteriori $P(H_i|D)$ riflette l’influenza dei dati, in contrasto con la probabilità a priori $P(H_i)$ che ne è indipendente.

$$H_{MAP} = P(H_i|D) = P(D|H_i)*P(H_i) \quad (3.6)$$

Ci sono state molte discussioni su come scegliere la probabilità a priori di un modello: un metodo è quello di privilegiare le ipotesi più semplici rispetto a quelle più complesse (rasoio di Ockham¹) e di fare in modo di rispettare il vincolo di normalizzazione sull’intero spazio delle ipotesi.

La preferenza per le ipotesi più semplici garantisce una certa immunità al rumore e al sovraddestramento, ma tale preferenza non deve essere troppo spinta per evitare di andare incontro ad un sottoaddestramento, cioè ad una situazione in cui molti dati vengono ignorati. Molto spesso, per mancanza di informazioni o per semplicità di calcolo, si assume una probabilità uniforme sulle ipotesi e si sceglie quindi l’ipotesi H_i che massimizza la $P(D|H_i)$: tale ipotesi viene chiamata massima verosimiglianza o ML (maximum likelihood) e indicata con H_{ML} :

$$H_{ML} = P(D|H_i) \quad (3.7)$$

3.2 Rete di credenze

Una rete di credenze, anche definita rete bayesiana, permette di rappresentare le relazioni tra un insieme molto vasto di variabili aleatorie che codifica in modo molto efficiente la distribuzione congiunta di probabilità, sfruttando le relazioni di indipendenza condizionale tra le variabili. Una rete di credenze fornisce una descrizione completa del dominio(Cfr. [23]). Una rete di Bayes è un grafo con le seguenti proprietà:

¹*Rasoio di Ockham*: è il nome con cui viene contraddistinto un principio metodologico espresso nel XIV secolo dal filosofo e frate francescano inglese William of Ockham. Tale principio, alla base del pensiero scientifico moderno, nella sua forma più semplice suggerisce la ricerca di ciò che è necessario per risolvere un problema o un spiegare un evento.

- Un insieme di variabili casuali costituiscono i nodi della rete.
- Un insieme di archi con un verso connette le coppie di nodi. L’inferenza tra un nodo X ed un nodo Y significa che X ha un’influenza diretta su Y.
- La mancanza di un arco tra due nodi indica l’indipendenza condizionale tra le due variabili casuali.
- Ogni nodo ha una tabella di probabilità condizionate che quantifica gli effetti che i genitori hanno sul nodo. I genitori di un nodo sono tutti quei nodi che hanno un’inferenza su quel nodo.
- Il grafo ottenuto non deve presentare cicli (DAG, Direct Acyclic Graph).

Vi sono due modi d’intendere la semantica di una rete di credenze:

1. si può vedere la rete come una rappresentazione della distribuzione della probabilità congiunta;
2. si può vedere la rete come una codifica di un insieme di enunciati di indipendenza condizionale.

Per poter valutare ogni probabilità di interesse, è sufficiente conoscere la distribuzione congiunta di probabilità dell’insieme di variabili aleatorie $X = X_1, \dots, X_n$ che descrivono il dominio di interesse. Usando la legge di Bayes che definisce $P(X,Y) = P(X|Y)*P(Y)$, si ottiene che:

$$P(X) = P(x_1, \dots, x_n) = P(x_n|x_{n-1}, \dots, x_1)*P(x_{n-1}, \dots, x_1) \quad (3.8)$$

Ripetendo la regola, otteniamo:

$$\begin{aligned} P(x_1, \dots, x_n) &= P(x_n|x_{n-1}, \dots, x_1)*P(x_{n-1}|x_{n-2}, \dots, x_1)*P(x_2|x_1)*P(x_1) \\ &= \prod_{i=1}^n P(x_i|x_{i-1}, \dots, x_1) \end{aligned} \quad (3.9)$$

Considerando l’equazione usata per la rete di credenze, che era

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i|parent(X_i)) \quad (3.10)$$

e si pone

$$parent(X_i) \subseteq x_{i-1}, \dots, x_1 \quad (3.11)$$

Questa relazione è di fondamentale importanza in quanto permette di esprimere la probabilità congiunta di tutte le variabili, come il prodotto delle probabilità condizionali locali limitate alle sole variabili che hanno un’influenza diretta sulla variabile in esame. Si può affermare che la rete di credenze è una rappresentazione

corretta del dominio solo se ogni nodo è condizionalmente indipendente dai suoi predecessori nell’ordinamento dei nodi, dati i suoi genitori. Pertanto, per costruire una rete di credenze con la corretta struttura per il dominio, occorre scegliere i genitori di ogni nodo in maniera che valga la proprietà espressa sopra.

Per costruire una rete bayesiana di un dominio di interesse, si può procedere nel seguente modo:

1. si individua un insieme di variabili aleatorie $X = \{X_1, \dots, X_n\}$ che descrivono il dominio;
2. si sceglie un ordinamento per le variabili in X ;
3. si ripete il procedimento seguente finché ci sono variabili:
 - si prende una variabile X_i e si aggiunge un nodo alla rete;
 - si pone $parent_i$ come un qualche insieme minimo di nodi già presenti nella rete, tale che la proprietà di indipendenza condizionale sia soddisfatta;
 - si definisce la tabella delle probabilità condizionate per X_i .

Possiamo rappresentare il grafo come un albero, alla radice abbiamo le cause, che sono i genitori (parent), mentre alle foglie abbiamo i figli che non possono influenzare le altre variabili casuali.

3.2.1 Inferenza nelle reti di credenze

Il processo che permette il calcolo di una probabilità di interesse dato un modello, viene chiamato inferenza probabilistica. Il compito fondamentale di un sistema di inferenza probabilistica è quello di calcolare la distribuzione di probabilità a posteriori per un insieme di variabili di interrogazione, dati i valori esatti per alcune variabili di prova. L’inferenza bayesiana è un approccio all’inferenza statistica in cui le probabilità non sono interpretate come frequenze, proporzioni o concetti analoghi, ma piuttosto come livelli di fiducia nel verificarsi di un dato evento. Rappresenta una generalizzazione del metodo scientifico. Data una generica ipotesi, non si ha certezza che essa possa verificarsi, ma avendo una storia di dati raccolti allora si può avanzare delle considerazioni di fiducia sul risultato ottenuto. Gli statistici bayesiani sostengono inoltre che l’inferenza bayesiana costituisca la base più logica per discriminare tra ipotesi alternative o in conflitto. Attraverso questo approccio, si usa una stima del grado di confidenza in una data ipotesi prima dell’osservazione dei dati, al fine di associare un valore numerico al grado di confidenza in quella stessa ipotesi successivamente all’osservazione dei dati(Cfr. [1]).

Il teorema di Bayes ci permette di modificare il livello di confidenza in una data

ipotesi, alla luce di una nuova informazione. Definendo H_0 l’ipotesi nulla² e con e il dato empirico o evento verificato, possiamo enunciare il seguente risultato:

$$P(H_0|e) = \frac{P(e|H_0)|P(H_0)}{P(e)} \quad (3.12)$$

L’ipotesi iniziale deve essere formulata in anticipo. Il fattore di scala $P(e|H_0)/P(e)$ può essere interpretato come una misura dell’impatto che l’osservazione di e ha sul grado di confidenza del ricercatore nell’ipotesi nulla, rappresentato a sua volta dalla probabilità a priori $P(H_0)$; se è altamente inverosimile che sia osservato, a meno che H_0 non sia proprio vera, il fattore di scala sarà elevato. La probabilità (confidenza) a posteriori, di conseguenza, combina le convinzioni che il ricercatore ha a priori con quelle derivanti dall’osservazione del dato empirico.

Possiamo distinguere tra le seguenti strutture di inferenza(Cfr. [20]):

1. inferenza causale: dato il verificarsi di una certa causa si calcola la probabilità dell’evento collegato. Le altre cause non vengono osservate. Figura 2.1 (a);
2. inferenza diagnostica: dato il verificarsi di un certo evento, qual’è la probabilità che sia dovuto ad una certa causa: si passa dall’effetto alla causa. Figura 2.1 (b);
3. inferenza intercausale: supponiamo di conoscere più cause di uno stesso effetto che possono essere anche indipendenti tra loro: se scopriamo una qualche evidenza su una delle cause dell’evento, allora le altre cause diventano meno probabili; l’evidenza su una causa rende meno probabili le altre. Figura 2.1 (c);
4. inferenza mista: combinazione delle inferenze enunciate. Figura 2.1 (d).

3.2.2 Apprendimento delle reti bayesiane

Per poter formulare il problema dell’apprendimento nelle reti bayesiane, è necessario innanzitutto stabilire cosa si vuole apprendere. Può succedere che la struttura sia nota o meno e che le variabili della rete siano osservabili oppure nascoste. Esempi di apprendimento possono fornire dati per tutte le variabili della rete, o solo per alcune. Se si conosce la struttura e tutte le variabili sono osservabili, allora l’allenamento è equivalente a quello del Naive Bayes, vi è solo da calcolare la tabella delle probabilità associate ai nodi della rete stessa: è il caso del classificatore Naive Bayes.

²ipotesi nulla: in un test statistico viene verificata in termini probabilistici la validità di un’ipotesi statistica, detta appunto ipotesi nulla, di solito indicata con H_0 .

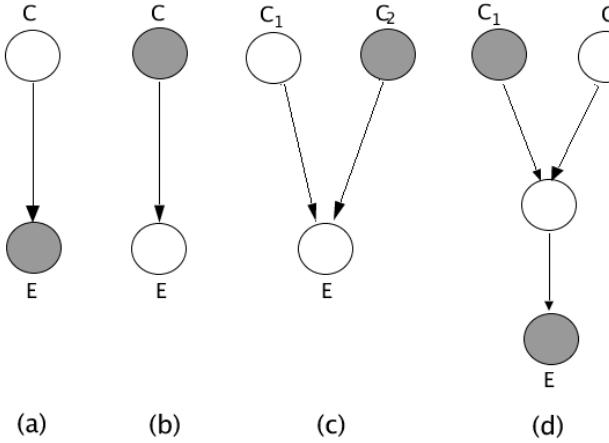


Figura 3.1. Tipi di inferenza in una rete Bayesiana

3.3 Naive Bayes

Il Naive Bayes è un algoritmo di classificazione basato sulla legge di Bayes. Come dice il nome stesso è basato su un'assunzione semplificativa, ossia che tutti gli attributi X_1, \dots, X_n che descrivono una certa istanza siano tutti condizionalmente indipendenti tra loro data la categoria a cui appartiene l'istanza Y. Il valore di questa assunzione semplifica drasticamente la rappresentazione di $P(X|Y)$ e il problema di calcolo del valore dall'insieme di dati messo a disposizione al modello (Cfr. [21]). Consideriamo, per esempio, la variabile casuale $X = \{X_1, X_2\}$. In questo caso:

$$\begin{aligned} P(X|Y) &= P(X_1, X_2|Y) \\ &= P(X_1|X_2, Y)*P(X_2|Y) \\ &= P(X_1|Y)*P(X_2|Y) \end{aligned} \quad (3.13)$$

Possiamo notare come nella seconda riga dell'equazione rispettiamo le regole probabilistiche, mentre nella terza facciamo un'assunzione di indipendenza condizionale. Generalizzando la definizione sopra, quando la variabile casuale X contiene un numero generico di attributi, che sono condizionalmente indipendenti, allora possiamo definire:

$$P(X_1, \dots, X_n|Y) = \prod_{i=1}^n P(X_i|Y) \quad (3.14)$$

Estendendo quanto detto sopra al caso di una variabile casuale Y discreta, così come gli attributi X_1, \dots, X_n . L'idea da seguire è quella di avere un classificatore che dia in uscita la distribuzione di probabilità su tutti i possibili valori di Y, per ciascuna

nuova istanza X_i che viene classificata. La legge di Bayes possiamo interpretarla come:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k) * P(X_1, \dots, X_n | Y = y_k)}{\sum_{j=1}^n P(Y = y_j) * P(X_1, \dots, X_n | Y = y_j)} \quad (3.15)$$

Supponendo che gli attributi X_i siano condizionalmente indipendenti da Y ed applicando la 3.13 otteniamo:

$$P(Y = y_k | X_1, \dots, X_n) = \frac{P(Y = y_k) * \prod_{i=1}^n P(X_i | Y = y_k)}{\sum_{j=1}^n P(Y = y_j) * \prod_{i=1}^n P(X_i | Y = y_j)} \quad (3.16)$$

La formula 3.16 è la fondamentale equazione per il classificatore Naive Bayes. Data una nuova variabile casuale $X^{new} = (X_1, \dots, X_n)$ questa equazione mostra come calcolare la probabilità che Y assuma per ogni valore, conoscendo i valori degli attributi osservati e date le distribuzioni $P(Y)$ e $P(X_i | Y)$ calcolati dalla conoscenza regressa. Se siamo interessati solo alla massima probabilità di Y, la formula può essere riscritta come segue:

$$Y \leftarrow \arg \max_{y_k} \frac{P(Y = y_k) * \prod_{i=1}^n P(X_i | Y = y_k)}{\sum_{j=1}^n P(Y = y_j) * \prod_{i=1}^n P(X_i | Y = y_j)} \quad (3.17)$$

semplificando il denominatore, poiché non è dipendente da y_k , otteniamo (rappresenta per tale motivo una costante di normalizzazione per l'input considerato):

$$Y \leftarrow \arg \max_{y_k} P(Y = y_k) * \prod_{i=1}^n P(X_i | Y = y_k) \quad (3.18)$$

3.3.1 Utilizzo del Naive Bayes per input discreti

Consideriamo l'algoritmo di “learning” del Naive Bayes attraverso la descrizione dei parametri che devono essere calcolati e come possiamo calcolarli.

Gli attributi in ingresso, X_i , possono prendere j possibili valori, mentre Y è una variabile discreta che varia su K possibili valori, quindi il compito di apprendimento riguarda la stima dei due set di parametri. Il primo è:

$$\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k) \quad (3.19)$$

per ciascun attributo di X_i , ciascun dei suoi possibili valori di x_{ij} e ciascun dei possibili valori y_k di Y.

Successivamente dobbiamo calcolare la probabilità a priori su Y:

$$\pi_k = P(Y = y_k) \quad (3.20)$$

con K parametri, di cui $K - 1$ sono condizionalmente indipendenti.

Noi possiamo calcolare questi parametri usando sia il calcolo della massima probabilità (basato sul calcolo delle relative frequenze dei differenti eventi nella base di conoscenza) oppure usando la stima della MAP (massima a posteriori). La massima probabilità associata a θ_{ijk} dato un insieme di dati D su cui fare l'addestramento è ottenuta:

$$\begin{aligned}\theta_{ijk} &= P(X_i = x_{ij} | Y = y_k) \\ &= \frac{\#D(X_i = x_{ij} \wedge Y = y_k)}{\#D(Y = y_k)}\end{aligned}\tag{3.21}$$

in cui $\#D(x)$ indica la cardinalità degli elementi nell'insieme dei dati D che soddisfa la proprietà x . Il problema di questo metodo è che spesso il risultato di θ dà come risultato zero se l'attributo di input non soddisfa i dati contenuti nel dominio di conoscenza. Per prevenire questo tipo di comportamento si utilizza una stima modificata o anche detta smorzata. Essa somma al numeratore un numero chiamato *visione* e si suppone che questo numero sia diffuso per tutta la variabile casuale. La stima precedente viene così modificata:

$$\begin{aligned}\theta_{ijk} &= P(X_i = x_{ij} | Y = y_k) \\ &= \frac{\#D(X_i = x_{ij} \wedge Y = y_k) + l}{\#D(Y = y_k) + lJ}\end{aligned}\tag{3.22}$$

in cui J normalizza il rapporto su tutta la variabile casuale X e l determina la lunghezza dello smorzamento. Questo risultato rappresenta una stima MAP di θ_{ijk} se noi utilizziamo una distribuzione di probabilità a priori di Dirichlet³ sui θ_{ijk} parametri equivalenti. Se poniamo l pari ad uno, la formula 3.22 prende il nome di *smorzamento di Laplace*.

La massima probabilità per π_k viene così calcolata:

$$\pi_k = P(Y = y_k) = \frac{\#D(Y = y_k)}{|D|}\tag{3.23}$$

in cui $|D|$ è il numero di elementi nella conoscenza regressa di D . Possiamo ottenere una stima modificata o anche definita stima MAP utilizzando la legge di Dirichlet sui π_k parametri assumendo uguale probabilità a priori su ciascun π_k , utilizzando la seguente espressione:

$$\pi_k = P(Y = y_k) = \frac{\#D(Y = y_k) + l}{|D| + lK}\tag{3.24}$$

³Peter Gustav Lejeune Dirichlet: fu un matematico tedesco ricordato soprattutto per la moderna definizione formale di funzione e per gli studi condotti sulla teoria dei numeri.

in cui K è il numero dei distinti valori che Y può avere e l determina la lunghezza dello smorzamento sui dati osservati D .

3.3.2 Il classificatore Naive Bayes

Il dominio applicativo del classificatore Naive Bayes riguarda la classificazione di istanze che possono essere descritte mediante un insieme di attributi di cardinalità anche molto elevata (Cfr. [20]). Innanzitutto si devono stimare i parametri del modello utilizzando i dati a disposizione. La stima di questi parametri corrisponde ad aver identificato un modello tra tutti quelli presenti nello spazio delle ipotesi. A differenza di altri algoritmi di apprendimento, non c’è un’esplicita ricerca nello spazio delle possibili ipotesi, ma l’ipotesi viene definita semplicemente contando la frequenza degli attributi negli esempi di addestramento. Il modello così definito permette di classificare le nuove istanze che gli vengono presentate. Sia $A = \{A_1, \dots, A_n\}$ un insieme di variabili che modellano gli attributi delle istanze da classificare e sia $C = \{c_1, \dots, c_{|C|}\}$ una variabile i cui stati rappresentano le categorie a cui appartengono le istanze. Si tratta di stimare una funzione che, applicata ad ogni dato di ingresso o istanza, fornisca la classe di appartenenza dell’istanza stessa, partendo da un insieme di dati di addestramento D in cui ogni elemento $d_i = \{a_1, \dots, a_n\}$ descritto tramite i valori dei suoi attributi e tramite la sua classe c_i .

Si veda innanzitutto come può essere classificata una nuova istanza partendo dalla sua descrizione in termini di attributi. Secondo l’approccio bayesiano, si calcolano le probabilità posteriori $P(c_j|d_i)$ con $j = 1, \dots, |C|$ e si determina la categoria c_j che massimizza tale probabilità:

$$c_{MAP} = \arg \max_{c_j \in C} P(c_j|d_i) = \arg \max_{c_j \in C} P(c_j|a_1, \dots, a_n) \quad (3.25)$$

Il teorema di Bayes permette di esprimere la probabilità a posteriori in funzione della verosimiglianza e della probabilità a priori:

$$\begin{aligned} c_{MAP} &= \arg \max_{c_j \in C} \frac{P(d_i|c_j)P(c_j)}{P(d_i)} \\ &= \arg \max_{c_j \in C} \frac{P(a_1, \dots, a_n|c_j)P(c_j)}{P(a_1, \dots, a_n)} \quad (3.26) \\ &= \arg \max_{c_j \in C} P(a_1, \dots, a_n|c_j)P(c_j) \end{aligned}$$

Eliminando dalla 3.18 il denominatore e applicando l’ipotesi semplificativa di indipendenza dei dati della 3.14 (mediante la quale riduciamo lo spazio delle stime) otteniamo:

$$c_{NB} \simeq c_{MAP} = \arg \max_{c_j \in C} P(c_j) \prod_{i=1}^n P(a_i|c_j) \quad (3.27)$$

Se l’ipotesi semplificativa è valida allora il $c_{NB} \equiv c_{MAP}$. Generalmente si calcola $P(a_i|c_i)$ come segue:

$$P(a_i|c_i) = \frac{n_c + m*p}{n + m} \quad (3.28)$$

in cui:

n = il numero di esempi per cui $c = c_i$;

n_c = il numero di esempi per cui $c = c_i$ e $a = a_i$;

p = probabilità a posteriori di $P(a_i|c_i)$;

m = costante chiamata taglia del campione equivalente che determina il peso di p e dei dati osservati.

Per riassumere:

$$p = \frac{1}{|c_i|} \quad m = |c_i| \quad (3.29)$$

ma generalmente m viene utilizzato come costante di smorzamento in tutta la classificazione. Infine n rappresenta il numero totale dei documenti a disposizione, mentre n_c rappresenta il numero di volte che la classe c_j compare nei dati: Per riassumere:

$$n = |D| \quad n_c = \sum_{i=1}^{|D|} P(c_j|d_i) \quad (3.30)$$

Il classificatore Naive Bayes e più in generale il problema della classificazione, possono essere associati ad una rete bayesiana che codifica le relazioni di indipendenza presenti tra le variabili che descrivono il dominio. Nel caso del Naive Bayes, le variabili di interesse sono gli attributi $A = \{A_1, \dots, A_n\}$ che descrivono le istanze da classificare e la categoria $C = \{C_1, \dots, C_{|C|}\}$ i cui stati rappresentano le classi delle istanze. Sia quindi $U = \{X \cup Y\}$ l’universo di variabili della rete bayesiana. La distribuzione congiunta globale $P(U)$ viene codificata nella topologia della rete, mediante la presenza o la mancanza di archi orientati tra i nodi del grafo. Innanzitutto, come si può vedere dalla 3.26, la variabile che rappresenta la categoria ha un’influenza diretta sull’istanza; supponendo di modellare l’istanza con una variabile X che condensa tutti gli attributi, la topologia della rete assume la forma di figura 3.2. Con $X = \{a_1, \dots, a_n\}$ e C la classe, otteniamo una rete di credenza rappresentata da un grafo aciclico. Graficamente si nota che la direzione dell’arco, che va dalla classe all’istanza, ha l’effetto di appartenere ad una certa categoria e quindi la categoria ha un’influenza diretta sulla generazione dell’istanza. La variabile casuale che esprime gli attributi al suo interno è descritta mediante un grafo aciclico. Adottando l’ipotesi semplificativa del Naive Bayes considerando gli attributi indipendenti data la classe, possiamo ridisegnare la rete di credenza come riporta la figura 3.3.

Si noti che non vi sono archi tra gli attributi come conseguenza dell’ipotesi semplificativa. L’apprendimento e la classificazione del Naive Bayes possono essere interpretati alla luce della corrispondente rete bayesiana. I parametri della rete da



Figura 3.2. Rete di credenza ottenuta dall’approccio Naive Bayes

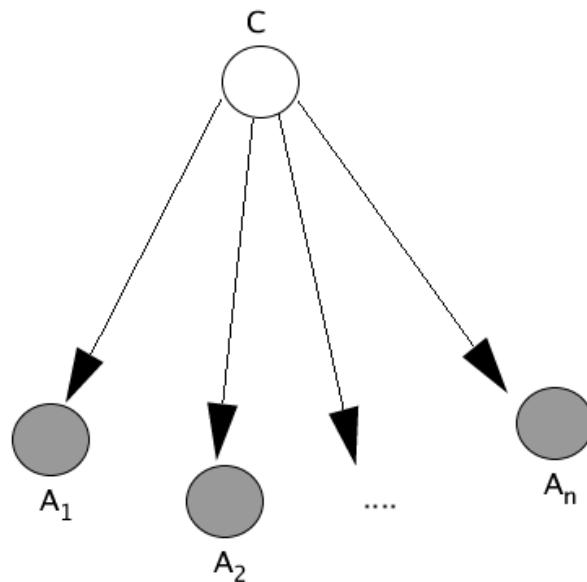


Figura 3.3. Rete di credenza con gli attributi indipendenti dalle classi

apprendere sono le probabilità $P(c_j)$ e $P(a_i|c_j)$. Una generica istanza descritta dai valori dei suoi attributi e dalla corrispondente classe, può essere vista come l’assegnazione dell’evidenza a tutte le variabili presenti nella rete e costituisce uno dei dati

completi che fanno parte dell’insieme di addestramento. L’apprendimento non è quindi altro che la stima dei valori delle tabelle di probabilità associate con i nodi della rete. Il problema della classificazione può invece essere interpretato come un caso particolare di inferenza nella rete. In questa circostanza si tratta di assegnare l’evidenza agli attributi come riportato in figura 3.3 che rappresentano l’istanza (nodi in grigio) e di propagarla verso il nodo che modella la classe: si calcola quindi la probabilità di ogni categoria, data una particolare realizzazione dell’istanza, mediante la procedura di inferenza nella rete.

3.4 Inferenze del Naive Bayes e catene di Markov

Una forte limitazione dell’algoritmo del “Naive Bayes” è l’assunzione sull’indipendenza statistica delle parole. Come si è commentato nel paragrafi precedenti, questo nei modelli linguistici non è propriamente vero. A tal fine, si è cercato di sviluppare degli algoritmi che potessero integrare la semplicità computazionale del “Naive”, aggiungendo alcune caratteristiche con le quali rappresentare meglio il modello linguistico. Si elenca un algoritmo tra tutti, il “Tree Augumented Naive Bayes Classifier”(Cfr. [?]) con il quale si è cercato di costruire alberi linguistici caratterizzati dalla dipendenza da radici nascoste e le variabili osservate. Questo problema, talvolta, assume complessità elevate fornendo delle prestazioni molto lontane dal modello del “Naive Bayes”. Si è cercato, quindi, di semplificare l’albero del precedente algoritmo, restringendo le dipendenze delle variabili osservate da una catena di Markov(Cfr. [22]) invece di un albero. Una catena di Markov può essere considerata come una rete bayesiana dinamica, nella fig. 3.4 vi è un esempio. Il modello linguistico prende il nome di CAN⁴. L’utilizzo di questo modello è completato con l’uso del concetto di “n-gram”⁵. In un regolare modello di Markov, lo stato non è direttamente visibile all’osservatore e perciò le probabilità della transizione dello stato sono gli unici parametri. In un modello di Markov nascosto lo stato non è direttamente visibile, ma le variabili influenzate dagli stati lo sono. Ciascuno stato ha una distribuzione di probabilità su tutte le possibili uscite. Quindi la sequenza delle uscite fornisce delle informazioni sulla sequenza degli stati.

In generale lo scopo del modello linguistico è di predire la probabilità di una naturale sequenza di parole o, molto semplicemente, esprimere con un’alta probabilità le sequenze di parole che occorrono di frequente, il contrario per le parole che non occorrono mai. Si esprime con perplessità la qualità del modello linguistico. Esso si

⁴CAN: è il nome con cui viene definito il modello Chain Augmented Naive Bayes.

⁵n-gram: nello studio dei modelli linguistici, questo termine è utilizzato per indicare parti di una generica espressione o parola.

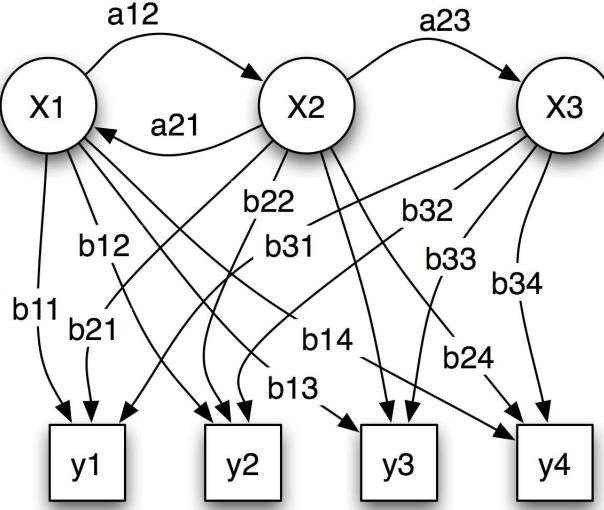


Figura 3.4. Esempio di una catena di Markov (Cfr. [31])

esprime:

$$Perplexity = \sqrt[T]{\prod_{i=1}^T \frac{1}{P(\omega_i | \omega_1 \dots \omega_{i-1})}} \quad (3.31)$$

L’obiettivo è quello di ridurre il valore della perplessità. Il più semplice e vincente modello linguistico è quello basato sugli n-gram. La regola di probabilità di una singola sequenza diventa:

$$P(\omega_1 \omega_2 \dots \omega_T) = \prod_{i=1}^T \frac{1}{P(\omega_i | \omega_1 \dots \omega_{i-1})} \quad (3.32)$$

Il modello n-gram approssima questa probabilità con l’assunzione che le sole parole rilevanti nel predire il modello sono le $n - 1$. Questa viene utilizzata dall’assunzione del modello di Markov:

$$P(\omega_1 | \omega_1 \dots \omega_{i-1}) = P(\omega_i | \omega_1 \dots \omega_{i-1}) \quad (3.33)$$

Un modo diretto per la stima della massima probabilità da un testo è dato dall’analisi del numero degli n-gram osservati come segue:

$$P(\omega_1 | \omega_1 \dots \omega_{i-1}) = \frac{\#(\omega_{i-n+1} \dots \omega_i)}{\#(\omega_1 \dots \omega_{i-1})} \quad (3.34)$$

L’utilizzo degli n-gram può, però, portare a problemi di complessità elevati. Infatti, si prenda per esempio una dimensione di vocabolario di W , usando una stima basata su n-gram di dimensione n , gli eventi da calcolare sono W^n .

Il ruolo di un classificatore di testi è quello di identificare attributi che siano in grado di distinguere un documento su differenti categorie. Il modello n-gram può essere applicato alla classificazione di testi in una maniera simile a quella del modello del Naive Bayes. In questo caso:

$$\begin{aligned}
 c^* &= \arg \max_{c \in C} \{P(c|d)\} \\
 &= \arg \max_{c \in C} \{P(d|c)P(c)\} \\
 &= \arg \max_{c \in C} \{P(d|c)\} \\
 &= \arg \max_{c \in C} \{\prod_{i=1}^T P(\omega_i|\omega_{i-n+1} \dots \omega_{i-1}, c)\}
 \end{aligned} \tag{3.35}$$

Nelle prime due uguaglianze dell'eq. (3.35) si assume una distribuzione uniforme sulle categorie, mentre nell'ultima si utilizza un'assunzione basata sulla catena di Markov e sugli n-gram. Il principio per l'uso di un modello linguistico basato sugli n-gram come classificatore di documenti è di determinare la categoria che meglio approssima il documento analizzato su tutta la base di conoscenza di n-gram suddivisa in n-gram. La rete di inferenze che si crea ha la forma del Naive Bayes, ma presenta una particolare caratteristica di dipendenza degli n-gram generati che la rendono una forma semplificata dell'albero di Markov.

Analizzando la fig. 3.5 si può notare una sostanziale differenza rispetto alla fig. 3.3, data dalla dipendenza dall'elemento A_n dal suo precedente A_{n-1} e dalla stessa radice. Questo rappresenta un esempio di un classificatore “Bi-Gram”, ossia dipendente dal “n-gram” precedente.

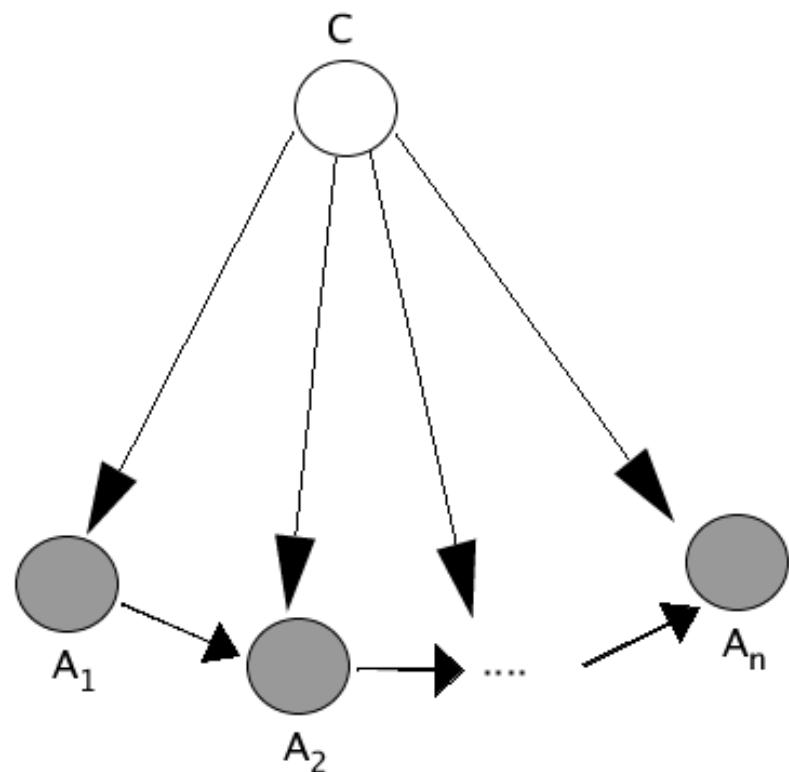


Figura 3.5. Bi-gram del classificatore Naive Bayes

Capitolo 4

Support Vector Machine

Le Support Vector Machine furono introdotte per la prima volta nel 1963 da Vladimir Vapnik¹ per la creazione di classificatori lineari di tipo supervised binari (Cfr. [26] [27]). L'algoritmo di una SVM prevede di trovare un iperpiano che separa gli esempi di training positivi da quelli negativi in modo da avere la massima distanza possibile tra l'iperpiano stesso e gli esempi più vicini ad esso. L'iperpiano individuato da questo algoritmo prende il nome di iperpiano a massimo margine.

Successivamente nel 1992 Bernhard Boser², Isabelle Guyon³ e Vapnik estesero le SVM alla classificazione non lineare. L'idea di base era di aggiungere un kernel all'algoritmo per il calcolo dell'iperpiano a massimo margine: sostituendo ai prodotti scalari una funzione kernel non lineare è possibile trovare un iperpiano all'interno di uno spazio delle feature trasformato. La trasformazione non lineare dello spazio delle feature permette di usare un classificatore lineare in uno spazio di dimensione più elevata per ottenere classificazioni non lineari all'interno dello spazio delle feature originale.

Il lavoro più interessante riguardante le SVM fu realizzato nel 1995 sempre da Vapnik con la collaborazione di Corinna Cortes⁴. I due introdussero il metodo del

¹ *Vladimir Naumovich Vapnik*: nato nell'URSS negli anni Trenta, conseguì nel 1958 la laurea in matematica presso l'università statale Usbek a Samarcanda e nel 1964 il dottorato in statistica presso l'istituto delle scienze del controllo a Mosca. Lavora presso questo istituto dal 1961 fino al 1990 quando diventa direttore del dipartimento di ricerca delle scienze del computer. Nel 1995 diventa professore di scienze del computer e di statistica presso la Royal Holloway University di Londra. Dal 1991 al 2001 circa lavora all'AT&T della Bell Labs.

² *Bernhard Boser*: Professore al Department of Electrical Engineering and Computer Science di Berkeley e Direttore di Berkeley Sensor & Actuator Center.

³ *Isabelle Guyon*: consulente di ingegneria indipendente specializzata in analisi statistica di dati, pattern recognition e tecniche di machine learning.

⁴ *Corinna Cortes*: Google Research New York, Google Labs.

Soft Margin, attraverso il quale si può calcolare il massimo margine anche nei casi di esempi di training classificati erroneamente. Nel caso in cui sia impossibile trovare un iperpiano in grado di separare gli esempi positivi da quelli negativi, il metodo del Soft Margin sceglie quello che divide gli esempi nel modo più pulito, continuando a massimizzare la distanza tra gli esempi chiaramente separati.

La giustificazione teorica sulla quale le SVM si appoggiano si basa sul principio della minimizzazione del rischio strutturale, che deriva dalla teoria dell'apprendimento statico (Cfr. [25]). Rispetto ad altri metodi che si basano sull'idea della minimizzazione dell'errore di apprendimento, le SVM minimizzano un limite superiore sull'errore di generalizzazione, minimizzando quindi il rischio strutturale. Dal punto di vista implementativo, il metodo consiste nella soluzione di un problema di ottimizzazione quadratico (*Quadratic Programming*). Con la teoria dei kernel si risolve contemporaneamente il problema computazionale e quello di generalizzazione. Questa si basa sull'idea che, proiettando i dati in uno spazio delle feature di dimensione più elevata, si aumenta il potere computazionale delle macchine per l'apprendimento lineare.

Nelle sezioni successive si fornirà una breve ma esaustiva spiegazione degli aspetti che hanno permesso a Vapnik e ai suoi collaboratori la formulazione e i successivi miglioramenti delle macchine a vettori di supporto.

4.1 La teoria dei kernel

Le funzioni di classificazione lineari non sono in grado di descrivere tutti i problemi che si affrontano nella pratica. Occorre quindi introdurre qualche tipo di non linearità per modellare il problema. A questo scopo si può utilizzare un'appropriata funzione kernel, cioè una funzione che esprima un prodotto scalare, che implicitamente crei una corrispondenza non lineare verso uno spazio delle feature di dimensioni maggiori.

Definizione 4.1.1 *Un kernel è una funzione K tale che per ogni $x,y \in X$*

$$K(x,z) = \langle \phi(x), \phi(y) \rangle$$

dove $\phi : X \rightarrow F$ è una corrispondenza tra lo spazio di ingresso X e uno spazio delle feature F (tale per cui il prodotto scalare è definito).

In base a ciò la funzione kernel deve essere perciò simmetrica; di conseguenza la matrice $K = (K(x_i, x_j))_{i,j}$ è simmetrica anch'essa e può essere rappresentata nella forma $K = V\Lambda V^T$, dove V è una matrice ortogonale e Λ è la matrice diagonale che contiene gli autovalori (λ_l) associati agli autovettori $v_t = (v_{ti})$ della trasformazione

(di cui t è la dimensione dello spazio di uscita) (Cfr. [8]).

Un importante teorema nella teoria dei kernel è il seguente:

Teorema 4.1.1 (di Mercer) *Sia X un sottoinsieme compatto di \mathbb{R}^n . Si supponga K sia una funzione simmetrica continua tale che*

$$\int_{X \times X} K(x,z) f(x) f(z) dx dz \geq 0 \quad (4.1)$$

per ogni $f \in L_2(X)$. Si può quindi espandere $K(x,z)$ in una serie uniformemente convergente del tipo

$$K(x,z) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(z) \quad (4.2)$$

dato $\phi_j \in L_2(X), \|\phi_j\|_{L_2} = 1$ e $\lambda_j \geq 0$.

Quanto enunciato implica che la matrice del kernel simmetrica definita positiva può essere utilizzata come matrice di un kernel. L'equazione 4.2 è l'espressione in autofunzioni del kernel di Mercer. Essa mostra come gli autovettori assumano il ruolo di feature: in effetti per un kernel $K(x,z) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(z)$ lo spazio delle feature è dato da $(\sqrt{\lambda_1} \phi_1(x), \dots, \sqrt{\lambda_i} \phi_i(x), \dots)$ dove λ_i e ϕ_i sono rispettivamente gli autovalori e le autofunzioni di $K(x,z)$.

Teorema 4.1.2 (Reproducing Kernel Hilbert Space) *Per ogni kernel di Mercer $K(x,z)$ definito sul dominio $X \in \mathbb{R}^d$ esiste uno spazio hilbertiano di riproduzione del kernel H (Reproducing Kernel Hilbert Space) di funzioni definite su X per le quali K è il kernel riproducente.*

Il teorema appena enunciato dimostra come sia possibile effettuare implicitamente corrispondenze non lineari usando la rappresentazione duale degli spazi hilbertiani. Vale anche l'inverso del problema. Esistono altre proprietà come la seguente.

Proposizione 4.1.1 *Siano K_1 e K_2 in $X \times X$ e valgono le condizioni $X \in \mathbb{R}^n$, $a \in \mathbb{R}^+$, f una funzione a valore reale in X ,*

$$\phi : X \rightarrow \mathbb{R}^m$$

con K_3 kernel in $\mathbb{R}^m \times \mathbb{R}^m$ e B sia una matrice $n \times n$ simmetrica semidefinita positiva.

Le funzioni elencate in seguito, in base a quanto detto precedentemente, sono quindi funzioni kernel:

1. $K(x,z) = K_1(x,z) + K_2(x,z)$
2. $K(x,z) = aK_1(x,z)$
3. $K(x,z) = K_1(x,z)K_2(x,z)$
4. $K(x,z) = f(x)f(z)$
5. $K(x,z) = K_3(\phi(x),\phi(z))$
6. $K(x,z) = x^T B z$

4.2 Teoria della generalizzazione

Le Support Vector Machine differiscono dagli altri metodi di classificazione principalmente perché hanno il loro fondamento nella teoria della generalizzazione: se si riesce a garantire una buona generalizzazione per una funzione di classificazione si può evitare un eccessivo adattamento agli esempi di training, ossia un overfitting⁵. Si è quindi cercato di ovviare a questo problema ed una soluzione è stata trovata con la teoria di Vapnik-Chervonenkis, in cui si cerca di trovare un limite superiore per l'errore di classificazione, ossia un valore che sia un buon compromesso tra capacità di apprendimento e performance della funzione di classificazione.

Nelle righe seguenti si dà un esempio di come un apprendimento è considerato probabilmente approssimativamente corretto.

Si considerino degli esempi di training indipendente e uniformemente distribuiti

$$(x,y) \in X \times \{1, -1\}$$

secondo \mathcal{D} sui quali si vuole apprendere l'associazione $x \rightarrow y$. Dato H , spazio delle funzioni di classificazione $h \in H$, e chiamata h_s la funzione ottenuta con l'addestramento. Imparare l'associazione vuol quindi dire scegliere una funzione h_s che minimizza l'errore di classificazione:

$$err(h_s) = \mathcal{D} \{(x,y) \mid h_s(x) \neq y\}$$

Supponendo di avere un insieme di “training” composto da n esempi:

$$S = ((x_1, y_1), \dots, (x_n, y_n))$$

⁵overfitting: in statistica si parla di overfitting (eccessivo adattamento) quando un modello statistico si adatta ai dati osservati (il campione) usando un numero eccessivo di parametri. Un modello assurdo e sbagliato può adattarsi perfettamente se è abbastanza complesso rispetto alla quantità di dati disponibili. Spesso si sostiene che l'overfitting è una violazione della legge del Rasoio di Ockham.

si dice che h_s probabilmente è approssimativamente corretta se la probabilità di errore è limitata:

$$\mathcal{D}\{S \mid \text{err}(h_s) < \epsilon(n, H, \delta)\} > \delta$$

Se la funzione ipotesi h_s è anche consistente, allora possiamo scrivere che la probabilità che abbia un errore maggiore di ϵ può essere limitata in questo modo

$$\mathcal{D}\{S \mid \text{consistente}(h_s) \wedge \text{err}(h_s) > \epsilon\} |H| (1 - \epsilon)^n < |H| e^{-\epsilon n}$$

$$\text{con } \epsilon = \epsilon(n, H, \delta) = \frac{1}{n} \ln \frac{|H|}{\delta}$$

4.3 Teoria di Vapnik-Chervonenkis

La teoria di Vapnik-Chervonenkis (anche conosciuta come la teoria del VC) fu sviluppata tra il 1960 e il 1990 da Vladimir Vapnik e Alexey Chervonenkis⁶. Questa è una forma della teoria di apprendimento computazionale che cerca di spiegare un processo di apprendimento da un punto di vista statistico. Tale teoria, spiegata in [?], copre principalmente quattro punti:

- Teoria della consistenza dei processi di apprendimento: quali sono le condizioni necessarie e sufficienti per la consistenza di un processo di apprendimento basato sul principio della minimizzazione del rischio empirico?
- Teoria della non asintoticità del valore di convergenza di un processo di apprendimento: quanto è veloce il valore di convergenza di un processo di apprendimento?
- Teoria del controllo della capacità di generalizzazione di un processo di apprendimento: come si può controllare il valore di convergenza di un processo di apprendimento?
- Teoria della costruzione di una macchina di apprendimento: come si possono creare algoritmi che possono controllare l'abilità di generalizzazione?

In quest'ultimo punto vengono introdotte proprio le Support Vector Machines. Il primo passo utile da fare è la comprensione della dimensione VC (teoria di Vapnik Chervonenkis)

⁶ *Alexey Chervonenkis*: matematico russo. È professore di Computer Science presso Computer Learning Research Centre.

Definizione 4.3.1 La dimensione VC è una proprietà dell’insieme H delle funzioni h . Dato un insieme di punti $(x_1 \dots x_n)$ per cui valga

$$h(x_1) \dots h(x_n) \mid h \in H = \{-1,1\}^n$$

l’insieme si dice frantumato (shattered) da H . La dimensione VC è definita come la dimensione massima dell’insieme degli esempi di addestramento che può venire frantumato da H .

In altri termini si afferma che la dimensione VC dello spazio di ipotesi H è un numero naturale che corrisponde al più grande numero di punti che possono essere separati in tutti i modi possibili dall’insieme di funzioni f_γ . Ossia, dato un insieme di l punti, se per ognuna delle 2^l possibili classificazioni $(-1; +1)$ esiste una funzione $\{f_\gamma\}$ che assegna correttamente le classi, allora si dice che l’insieme di punti viene separato dall’insieme di funzioni. Si può quindi dire che la dimensione VC è una misura della complessità dell’insieme H .

Avendo enunciato la definizione della dimensione VC si può andare ad analizzare l’aspetto più importante per i nostri fini della teoria di Vapnik Chervonenkis, dove viene mostrato come, per un insieme infinito di ipotesi, il problema di “overfitting” sia evitabile e limita la generalizzazione di una ipotesi costante in modo dipendente dalla distribuzione.

Teorema 4.3.1 (di Vapnik Chervonenkis) Sia H lo spazio delle ipotesi con dimensione VC pari a d . Per qualunque distribuzione \mathcal{D} su $X \times \{-1,1\}$ con probabilità $1 - \delta$ su n esempi casuali S , ogni ipotesi $h \in H$, che è consistente con S , ha un errore non più grande di:

$$\text{err}(h) = \epsilon(n, H, \delta) = \frac{2}{n} \left(d \log \frac{2en}{d} + \log \frac{2}{\delta} \right)$$

a condizione che $d \leq n$ e $n \geq 2/\epsilon$.

Il teorema dimostra come l’apprendimento in spazi di feature di dimensione molto elevata non sia possibile senza conoscere la distribuzione degli esempi. È quindi necessario che ci sia una distribuzione “benigna”, che faciliti l’apprendimento se lo spazio delle feature ha dimensione elevata.

Il teorema di Vapnik Chervonenkis può essere analizzato in termini di differenza tra errore di classificazione e quello di addestramento. Data una funzione $h \in H$, la costruzione di un classificatore SVM consiste nel trovare la h per cui sia possibile garantire il più basso errore $\text{err}(h)$ di classificazione dato l’insieme di addestramento $S = ((x_1, y_1) \dots (x_n, y_n))$. Il teorema dimostra che esiste un limite superiore che lega

l’errore vero dell’ipotesi $h(\text{err}(h))$ con $\text{err}_{\text{train}}(h)$ (errore di training) e la complessità di h :

$$\text{err}(h) \leq \text{err}_{\text{train}}(h) + O\left(\frac{d \ln \frac{n}{d} - \ln \delta}{n}\right) \quad (4.3)$$

Questo vincolo vale con una probabilità di almeno $(1 - \delta)$. La diseguaglianza 4.3 mostra il tradeoff che si trova tra complessità dello spazio delle ipotesi ed errore di addestramento. Se lo spazio delle ipotesi fosse semplice, quindi bassa dimensione VC, si avrebbe un grande errore di training e quindi di classificazione. Se invece la dimensione VC fosse elevata si avrebbe da un lato un basso errore di “training”, ma aumenterebbe il secondo termine dell’addizione. Ciò può implicare che con uno spazio con dimensione VC elevata ed errore di addestramento basso possa andare bene solo con gli esempi di training, ma in realtà non si sa se è in grado di predirne di nuovi in modo corretto.

Per ottenere l’errore teorico minimo bisogna quindi minimizzare sia l’errore empirico, sia il rapporto tra la dimensione VC e il numero di punti (h/l). Solitamente l’errore empirico è una funzione decrescente di h , quindi, per ogni dato numero di punti, esiste un valore ottimale della dimensione VC (il tradeoff descritto in precedenza). Per costruire un classificatore SVM occorre quindi risolvere efficacemente questo problema scegliendo uno spazio delle ipotesi con una complessità accuratamente valutata. Secondo la teoria della minimizzazione del rischio strutturale questo è possibile definendo una struttura annidata di spazi di ipotesi H , costruita in modo che la dimensione VC pari a d aumenti:

$$H_1 \subset H_2 \subset H_3 \dots \subset H_n, \forall i : d_i \leq d_{i+1} \quad (4.4)$$

Questa struttura deve essere costruita a priori, senza fare ipotesi sull’insieme di training. L’obiettivo per la costruzione del classificatore diventa quindi quello di trovare l’indice i^* dello spazio che minimizza 4.3.

Concretamente le SVM apprendono usando funzioni lineari di tipo soglia come:

$$h(x) = \text{sign}(\omega \cdot x + b) = \begin{cases} +1 & , \text{se } \omega \cdot x + b > 0 \\ -1 & , \text{altrimenti} \end{cases} \quad (4.5)$$

che corrispondono ad iperpiani all’interno dello spazio delle feature. Ciò vuol dire che le SVM usano spazi delle ipotesi di funzioni lineari euclidei, che hanno dimensione VC pari a $r + 1$, dove r è la dimensione dello spazio delle feature, e che la classificazione è determinata semplicemente da quale parte dell’iperpiano si trovi un punto nello spazio delle feature. La figura 4.1 mostra la separazione di dati linearmente separabili. I dati da una parte dell’iperpiano sono per esempio quelli che soddisfano la classificazione, quelli dalla parte opposta, invece, non la soddisfano. In altri termini, inserendo il discorso all’interno della posta elettronica, in particolar modo per quanto riguarda i messaggi indesiderati, l’iperpiano divide i dati che sono

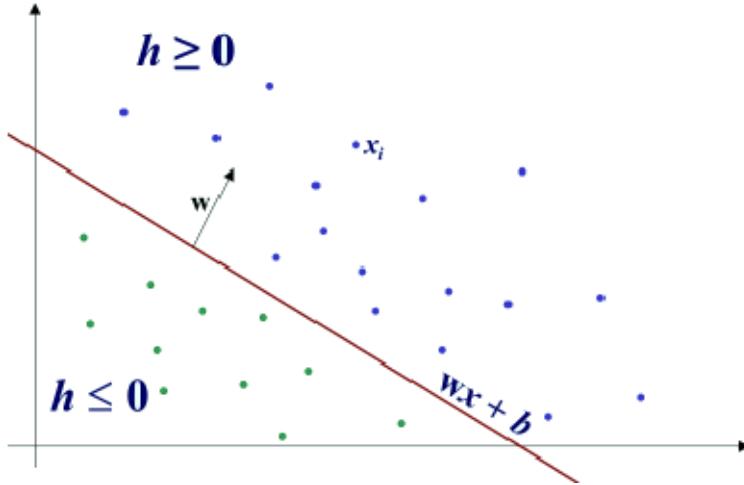


Figura 4.1. Separazione Lineare

spam da quelli che non lo sono.

Vapnik ha mostrato che il margine, ossia la distanza γ tra l'iperpiano e gli esempi ad esso più vicini, possa essere usato per limitare la dimensione VC in modo indipendente dal numero di feature. Questa indipendenza non garantisce che le prestazioni siano buone. Intuitivamente, però, si può affermare che se il numero di feature è elevato non necessariamente implica che l'SVM fallisca a priori. Assumiamo che l'insieme di training sia separabile da un iperpiano h' , ossia esista un vettore di pesi ω' e un termine noto b' tali che gli esempi positivi si trovano da una parte dell'iperpiano e quelli negativi dalla parte opposta (come mostrato nella figura 4.1). Questo può essere espresso come:

$$y_i[\omega' \cdot x' + b'] \geq 0, \forall i \quad (4.6)$$

In generale ci possono essere molteplici iperpiani che soddisfano la condizione espresa in 4.6. La figura 4.2 mostra come possano esserci molteplici iperpiani che dividono gli esempi di training positivi da quelli negativi. Quindi tra i vari possibili iperpiani si sceglie quello con margine maggiore.

La figura 4.3 mostra quindi la scelta dell'iperpiano con massimo margine. Infatti può esserci solo un iperpiano che ha la distanza massima tra gli esempi di training. In particolare gli esempi più vicini a tale iperpiano, ossia quelli che distano esattamente γ , sono chiamati *support vector*.

Trovare l'iperpiano con massimo margine può essere tradotto nel risolvere il seguente problema di ottimizzazione minimizzando:

$$\Phi(\omega, b) = \frac{1}{2} * \|\omega\|^2 \quad (4.7)$$

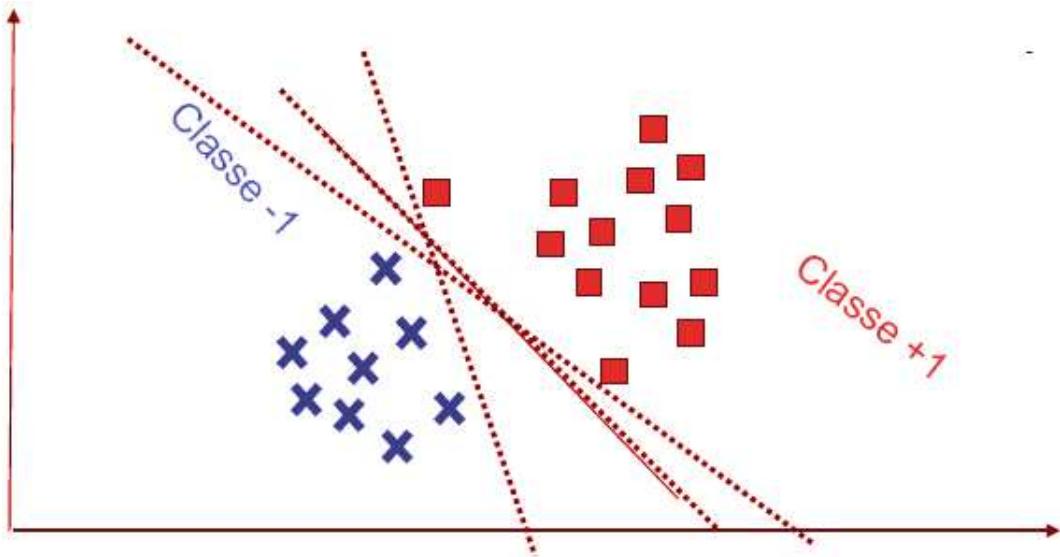


Figura 4.2. Alcuni possibili iperpiani di separazione in uno spazio bidimensionale

con il vincolo:

$$\forall_{i=1}^n : y_i[\omega' \cdot x' + b'] \geq 1 \quad (4.8)$$

Il vincolo 4.8 formalizza che tutti gli esempi si debbano trovare dalla parte corretta

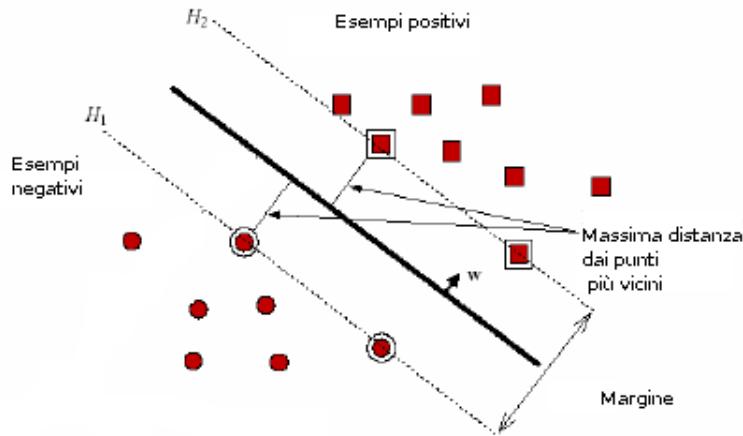


Figura 4.3. Iperpiano con distanza massima dagli esempi di training più vicini.

dell'iperpiano. Avere un termine ad uno al secondo membro della disequazione impone che ci sia una distanza γ dall'iperpiano, ossia il cosiddetto margine. Si è

verificato che:

$$\gamma = \frac{1}{\|\omega\|^2} \quad (4.9)$$

La minimizzazione di $\|\omega\|^2$ equivale a massimizzare il margine. In base a ciò si può notare che il vettore di pesi ω e il termine b descrivono l'iperpiano a massimo margine. La proposizione seguente, sviluppata da Vapnik, indica il legame che vi è tra il margine e la dimensione VC.

Proposizione 4.3.1 (dimensione VC di iperpiani con margine) *Si considerino gli iperpiani $h(x) = \text{sign}(\omega \cdot x + b)$ in uno spazio r dimensionale come funzioni di ipotesi. Se tutti i vettori di esempio x_i sono contenuti in una sfera di diametro R e per ogni esempio x_i vale che:*

$$\text{abs}(\omega \cdot x + b) \geq 1 \quad (4.10)$$

allora l'insieme di iperpiani ha una dimensione VC limitata da

$$d \leq \min([\frac{R^2}{\gamma^2}, r]) + 1 \quad (4.11)$$

dove $[.]$ è l'operatore di parte intera.

La proposizione 4.3 indica che la dimensione VC diminuisce all'aumentare del margine. Come affermato in precedenza la dimensione VC degli iperpiani a massimo margine non dipende necessariamente dal numero delle feature, bensì dalla lunghezza euclidea $\|\omega\|$ del vettore di pesi ottimizzato dalla SVM. Ciò implica che l'errore vero di un iperpiano di separazione a massimo margine è simile all'errore di “training” se il suo vettore è piccolo. L'affermazione vale anche nel caso in cui lo spazio sia ad elevato numero di dimensioni. Questo problema di ottimizzazione può risultare di difficile implementazione, per cui nelle applicazioni reali viene risolto il problema duale di Wolfe.

4.4 Duale di Wolfe

Il concetto di dualità è molto utilizzato nella programmazione matematica. In relazione a determinati problemi, la teoria della dualità consiste nel definire formulazioni alternative che, da un lato possono essere risolte più efficientemente da un punto di vista computazionale, dall'altro possono fornire elementi teorici rilevanti.

L'idea base della teoria della dualità è quella di costruire, in corrispondenza ad un problema assegnato di minimo, detto problema primale:

$$\min_{x \in S} f(x) \quad (4.12)$$

un problema di massimo, detto problema duale (definito generalmente su uno spazio differente)

$$\max_{u \in U} \psi(u) \quad (4.13)$$

in modo tale che valga almeno la condizione, detta proprietà di *dualità debole*:

$$\inf_{x \in S} f(x) \geq \sup_{u \in U} \psi(u) \quad (4.14)$$

Dove si riesce a stabilire questa corrispondenza è possibile fornire utili caratterizzazioni della soluzione del primale attraverso lo studio del duale.

In particolare dalla teoria della dualità è possibile ricavare:

- stime del valore ottimo;
- condizioni di ottimalità;
- metodi di soluzione basati sulla considerazione del problema duale.

Per alcune classi di problemi si riesce anche a stabilire la condizione detta di *dualità forte*

$$\inf_{x \in S} f(x) = \sup_{u \in U} \psi(u) \quad (4.15)$$

La possibilità di costruire un duale che soddisfi l'equazione 4.15 sussiste per una classe ristretta di problemi di ottimo che comprende tipicamente molti problemi di programmazione convessa. Si consideri il problema

$$\begin{aligned} & \min f(x) \\ & g_i(x) \leq 0, \quad i = 1, \dots, m \\ & c_j^T x - d_j = 0 \quad j = 1, \dots, p \end{aligned} \quad (4.16)$$

in cui $f : R^n \rightarrow R$, $g_i : R^n \rightarrow R$, con $i = 1, \dots, m$ sono funzioni convesse continuamente differenziabili. Sia

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j (c_j^T x - d_j) \quad (4.17)$$

Proposizione 4.4.1 *Si assume che il problema 4.16 ammetta almeno una soluzione ottima x^* e che esista almeno una coppia di moltiplicatori di Lagrange⁷*

⁷Joseph-Louis Lagrange: (Torino, 25 gennaio 1736 – Parigi, 10 aprile 1813) è stato un matematico e astronomo italiano, sicuramente uno tra i maggiori e più influenti matematici del XVIII secolo. La sua più importante opera è il testo *Mécanique analytique*, pubblicato nel 1788

(λ^*, μ^*) . Allora la tripla (x^*, λ^*, μ^*) è soluzione del seguente problema

$$\begin{aligned} & \max_{x, \lambda, \mu} L(x, \lambda, \mu) \\ & \nabla_x L(x, \lambda, \mu) = 0 \\ & \lambda \geq 0. \end{aligned} \tag{4.18}$$

Inoltre, il gap di dualità è nullo, ossia $f(x^*) = L(x^*, \lambda^*, \mu^*)$.

Il problema definito in 4.18 è normalmente chiamato come *problema duale di Wolfe*.

In generale, data una soluzione $(\bar{x}, \bar{\lambda}, \bar{\mu})$ del duale di Wolfe, non si possono trarre conclusioni sui vettori \bar{x} e $(\bar{\lambda}, \bar{\mu})$.

4.5 Programmazione quadratica

Si consideri il seguente problema quadratico

$$\min f(x) = \frac{1}{2} x^T Q x + c^T x \tag{4.19}$$

$$Ax - b \leq 0$$

dove $Q \in R^{n \times n}$, $A \in R^{m \times n}$, $c \in R^n$, $b \in R^m$. Posto $L(x, \lambda) = f(x) + \lambda^T(Ax - b)$, il duale di Wolfe è definito come segue:

$$\begin{aligned} & \max_{x, \lambda} L(x, \lambda) \\ & \nabla_x L(x, \lambda) = 0 \\ & \lambda \leq 0. \end{aligned} \tag{4.20}$$

Vale il risultato seguente.

Proposizione 4.5.1 Si assuma che la matrice Q sia semidefinita positiva. Sia $(\bar{x}, \bar{\lambda})$ una soluzione del duale di Wolfe 4.20. Allora esiste un vettore x^* (non necessariamente uguale a \bar{x}) tale che

- $Q(x^* - \bar{x}) = 0$;
- x^* è soluzione del problema 4.19;
- $(x^*, \bar{\lambda})$ è una coppia (minimo globale - vettore di moltiplicatori di Lagrange).

Dopo questa breve introduzione sulla programmazione quadratica si può elaborare il discorso alle support vector machine.

4.6 Programmazione quadratica per SVM lineari

Determinare una SVM lineare equivale a trovare l'iperpiano ottimo (Cfr. [18]), ossia risolvere il seguente problema

$$\begin{aligned} & \max \rho(\omega, b) \\ & \text{tali che } \omega^T x^i + b \geq 1, \quad \forall x^i \in A \\ & \quad \omega^T x^j + b \leq -1, \quad \forall x^j \in B \end{aligned} \tag{4.21}$$

dove A e B sono due insiemi di punti contenuti in R^n linearmente separabili. Si è visto che il problema 4.21 è equivalente al problema di programmazione quadratica convessa riportato in 4.22.

$$\begin{aligned} \min F(\omega) &= \frac{1}{2} \|\omega\|^2 \\ &\text{tali che } \omega^T x^i + b \geq 1, \quad \forall x^i \in A \\ & \quad \omega^T x^j + b \leq -1, \quad \forall x^j \in B \end{aligned} \tag{4.22}$$

Associando quindi in quest'ultimo problema l'etichetta $y^i = +1$ ai vettori $x^i \in A$ e l'etichetta $y^j = -1$ ai vettori $x^j \in B$ possiamo quindi scrivere:

$$\begin{aligned} \min F(x) &= \frac{1}{2} \|\omega\|^2 \\ &\text{tali che } y^i[\omega^T x^i + b] - 1 \geq 0, \quad i = 1, \dots, l \end{aligned} \tag{4.23}$$

essendo l il numero di punti presenti in $A \cup B$. Della 4.23 si consideri il problema duale poiché:

- i vincoli di 4.23 sono sostituiti con altri “più” semplici sui moltiplicatori di Lagrange;
- nella formulazione duale, i vettori di training compariranno attraverso prodotti scalari tra i vettori stessi e permetterà di generalizzare la procedura al caso di insiemi che non sono linearmente separabili.

La Langrangiana del 4.23 risulta essere:

$$L(\omega, b, \lambda) = \frac{1}{2} \|\omega\|^2 - \sum_{i=1}^l \lambda_i [y^i(\omega^T x^i + b) - 1] \tag{4.24}$$

il cui problema duale risulta essere

$$\begin{aligned} \max L(\omega, b, \lambda) &= \frac{1}{2} \|\omega\|^2 - \sum_{i=1}^l \lambda_i [y^i (\omega^T x^i + b) - 1] \\ \text{tali che } \omega &= \sum_{i=1}^l \lambda_i y^i x^i \\ \sum_{i=1}^l \lambda_i y^i &= 0 \\ \lambda_i &\geq 0 \quad i = 1, \dots, l \end{aligned} \tag{4.25}$$

che può essere riscritto come

$$\begin{aligned} \max S(\lambda) &= -\frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y^i y^j (x^i)^T x^j + \lambda^i \lambda^j + \sum_{i=1}^l \lambda_i \\ \text{tali che } \sum_{i=1}^l \lambda_i y^i &= 0 \\ \lambda_i &\geq 0 \quad i = 1, \dots, l \end{aligned} \tag{4.26}$$

che a sua volta è equivalente a

$$\begin{aligned} \max \Gamma(\lambda) &= \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y^i y^j (x^i)^T x^j + \lambda^i \lambda^j - \sum_{i=1}^l \lambda_i \\ \text{tali che } \sum_{i=1}^l \lambda_i y^i &= 0 \\ \lambda_i &\geq 0 \quad i = 1, \dots, l \end{aligned} \tag{4.27}$$

Da queste equazioni si osserva che:

- l'esistenza della soluzione ottima (ω^*, b^*) del problema 4.23 e le proposizioni precedenti assicurano che il problema 4.27 ammetta almeno una soluzione λ^* ;
- da 4.25 si ricava che ω^* può essere calcolato come:

$$\omega^* = \sum_{i=1}^l \lambda_i^* y^i x^i$$

- ω dipende solamente dai vettori di training x^i , ossia i vettori di supporto, i cui moltiplicatori λ^i non sono nulli;
- dal terzo punto di 4.5 assicura che $(\omega^*, b^*, \lambda^*)$ costituisce una coppia (soluzione ottima e vettore di moltiplicatori di Lagrange), per cui vale la seguente condizione di complementarietà:

$$\lambda_i^* [y^i ((\omega^*)^T x^i + b^*) - 1] = 0 \quad i = 1, \dots, l \tag{4.28}$$

- noto ω^* e considerato un qualsiasi moltiplicatore $\lambda_i^* \neq 0$, lo scalare b^* può essere determinato utilizzando la corrispondente condizione definita nella 4.28
- il problema 4.27 è un problema di programmazione quadratica convessa. Ponendo $X = [y^1x^1, \dots, y^lx^l]$, $\lambda^T = [\lambda^1, \dots, \lambda^l]$, il problema assume la forma:

$$\begin{aligned} \min \Gamma(\lambda) &= \frac{1}{2} \lambda^T X^T X \lambda - e^T \lambda \\ \text{tali che } \sum_{i=1}^l \lambda_i y^i &= 0 \\ \lambda_i &\geq 0 \quad i = 1, \dots, l \end{aligned} \tag{4.29}$$

dove $e^T = [1, \dots, l]$;

- la funzione di decisione risulta essere:

$$f(x) = \text{sign}((\omega^*)^T x + b^*) = \text{sign}\left(\sum_{i=1}^l \lambda^* y^i (x^i)^T x + b^*\right)$$

4.7 Il caso elementi non separabili linearmente

Nel caso di elementi non separabili linearmente possiamo ad esempio considerare due insiemi A e B disgiunti in R^n e naturalmente questi due insiemi non sono linearmente separabili. Quindi, ad esempio, il sistema

$$\begin{cases} \omega^T x^i + b \geq 1, & \forall x^i \in A \\ \omega^T x^j + b \leq -1, & \forall x^j \in B \end{cases} \tag{4.30}$$

non ammette soluzione. Nel sistema 4.30 si introducono quindi delle variabili positive dette di *slack* ξ_h , con $h = 1, \dots, l$. Il sistema precedente diventa quindi:

$$\begin{aligned} \omega^T x^i + b &\geq 1 - \xi_i, \quad \forall x^i \in A \\ \begin{cases} \omega^T x^j + b \leq -1 - \xi_j, & \forall x^j \in B \\ \xi_h \geq 0, & h = 1, \dots, l \end{cases} \end{aligned} \tag{4.31}$$

Da questo sistema si nota che, se un vettore di ingresso x^i è classificato erroneamente, la corrispondente variabile ξ^i risulta essere maggiore di 1. Quindi il termine $\sum_{i=1}^l \xi_i$ è un upper bound del numero di errori di classificazione dei vettori di training.

Risulta quindi naturale inserire a 4.23 un termine pari a $C \sum_{i=1}^l \xi_i$ in cui $C \geq 0$ che “pesa” l’errore di training. Con questo passaggio il nostro problema risulta essere:

$$\begin{aligned} \min F(x) &= \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^l \xi_i \\ \text{tali che } &y^i[\omega^T x^i + b] - 1 + \xi_i \geq 0, \quad i = 1, \dots, l \\ \xi_i &\geq 0 \quad i = 1, \dots, l \end{aligned} \tag{4.32}$$

Dopo vari calcoli matematici ricaviamo il problema duale che risulta essere:

$$\begin{aligned} \max \Gamma(\lambda) &= \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l y^i y^j (x^i)^T x^j + \lambda^i \lambda^j - \sum_{i=1}^l \lambda_i \\ \text{tali che } &\sum_{i=1}^l \lambda_i y^i = 0 \\ &0 \leq \lambda_i \leq C \quad i = 1, \dots, l \end{aligned} \tag{4.33}$$

Da questo si osserva che:

- il vettore ω^* può essere ricavato come:

$$\omega^* = \sum_{i=1}^l *l \lambda_i^* y^i x^i$$

- in corrispondenza della soluzione ottima $(\omega^*, b^*, \xi^*, \lambda^*)$ valgono le condizioni di complementarietà:

$$\lambda_i^* [y^i((\omega^*)^T x^i + b^*) - 1 + \xi_i^*] = 0 \quad i = 1, \dots, l \tag{4.34}$$

$$\mu_i^* \xi_i^* = 0 \quad i = 1, \dots, l \tag{4.35}$$

- noto ω^* e considerando un qualsiasi moltiplicatore $0 \leq \lambda_i^* \leq C$, lo scalare b^* può essere determinato utilizzando la corrispondente condizione definita in 4.34;
- nel caso in cui $\lambda_i^* \in 0, C$ per $i = 1, \dots, l$ la soluzione è detta *degenera*;
- il problema 4.33 è un problema di programmazione quadratica convessa;
- la funzione decisionale, infine, risulta essere:

$$f(x) = \text{sign}((\omega^*)^T x + b^*) = \text{sign}\left(\sum_{i=1}^l \lambda_i^* y^i (x^i)^T x + b^*\right)$$

4.8 SVM per problemi di regressione

In un problema di regressione, ogni osservazione del “training set” è costituita da una coppia (x_i, y_i) , in cui il pattern $x_i \in R^n$ e l’etichetta $y_i \in R$. Se il modello di ingresso-uscita lineare, ossia una funzione $f : R^n \rightarrow R$ assume la forma:

$$f(x, \omega, b) = \omega^T x + b$$

Sia $\epsilon \geq 0$ il grado di precisione desiderato con il quale si vuole approssimare la funzione rappresentata dai campioni del “training set” mediante il modello f . La stima del modello relativa ad un pattern x^i è considerata buona se risulta:

$$|y^i - f(x, \omega^T x^i, b)| \leq \epsilon$$

La *loss function* risulta essere:

$$|y^i - f(x, \omega^T x^i, b)|_\epsilon = \max \{0, |y^i - f(x, \omega^T x^i, b)| - \epsilon\} \quad (4.36)$$

e l’errore di “training” è

$$E = \sum_{i=1}^l |y^i - f(x, \omega^T x^i, b)|_\epsilon \quad (4.37)$$

L’errore di training risulta essere nullo solamente nel caso in cui il sistema di disequazioni successivo è soddisfatto:

$$\begin{cases} y^i - \omega^T x^i - b \leq \hat{\epsilon} \\ i = 1, \dots, l \end{cases} \quad (4.38)$$

La quantità $\sum_{i=1}^l (\epsilon + \hat{\epsilon})$ costituisce un “upper bound” dell’errore di “training”. Si può quindi procedere come nel caso delle SVM lineari e quindi dopo alcuni passaggi matematici arriviamo ad avere

$$\begin{aligned} \min_{\lambda, \hat{\lambda}} &= \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l (\hat{\lambda}_i - \lambda_i)(\hat{\lambda}_j - \lambda_j)(x^i)^T x^j - \sum_{i=1}^l (\hat{\lambda}_i - \lambda_i)y^i + \epsilon \sum_{i=1}^l (\hat{\lambda}_i + \lambda_i) \\ &\sum_{i=1}^l (\hat{\lambda}_i - \lambda_i) = 0 \\ &0 \leq \lambda \leq C \quad i = 1, \dots, l \\ &0 \leq \hat{\lambda} \leq C \quad i = 1, \dots, l \end{aligned} \quad (4.39)$$

Grazie alla teoria della dualità e l'utilizzo delle funzioni kernel si può generalizzare la trattazione al caso di modelli di regressione non lineari. In particolare il problema di addestramento di una support vector machine per la regressione non lineare risulta essere definito come:

$$\begin{aligned} \min_{\lambda, \hat{\lambda}} = & \frac{1}{2} \sum_{i=1}^l \sum_{j=1}^l (\hat{\lambda}_i - \lambda_i)(\hat{\lambda}_j - \lambda_j) k(x^i, x^j) - \sum_{i=1}^l (\hat{\lambda}_i - \lambda_i) y^i + \epsilon \sum_{i=1}^l (\hat{\lambda}_i + \lambda_i) \\ & \sum_{i=1}^l (\hat{\lambda}_i - \lambda_i) = 0 \\ & 0 \leq \lambda \leq C \quad i = 1, \dots, l \\ & 0 \leq \hat{\lambda} \leq C \quad i = 1, \dots, l \end{aligned} \tag{4.40}$$

dove $k(x, j)$ è una funzione kernel. Tuttavia, in pratica, l'addestramento di una SVM per la regressione, visto che occorre trovare simultaneamente i valori “ottimali” di ϵ e C , è più complesso che l'addestramento di una SVM per la classificazione. In genere tali valori vengono trovati con tecniche di *cross-validation*.

Capitolo 5

OWL per la descrizione di ontologie

Nell'ambito informatico l'ontologia rappresenta un formulazione di schemi di rappresentazione di un dato dominio.

“L'ontologia è una specificazione di una concettualizzazione”

Dalla definizione di T. R. Gruber¹ si può affermare che in genere si tratta di costruire una struttura dati gerarchica caratterizzata da una serie di regole, di assiomi ed inferenze che identificano gli elementi di un dominio e permettono a programmi o ad automi di poter ricorstruire l'albero delle relazioni.

Nello studio filosofico della persona, la parola ontologia si riferisce allo studio del proprio io e delle relazioni che un uomo instaura con altre persone. Spesso, il concetto di ontologia viene travisato con un'analisi epistemologica della conoscenza o anche definita teoria di conoscenza (contesti che rispondono alle domande cosa è la conoscenza oppure come sia la conoscenza di). Generalmente si usa il termine ontologia per definire in un contesto condiviso, una ben definita specificazione di concettualizzazione. Secondo gli esperti, una vera ontologia non deve limitarsi ad una gerarchia di concetti organizzati con la relazione di sussunzione (spesso chiamata “isa”, cioè “is-a” in lingua inglese, “sottoclasse”, o “sottotipo”), ma deve includere anche altre relazioni semantiche che descrivono in che modo i concetti sono relazionati tra loro. Una delle relazioni più comuni, oltre a quella di sussunzione, è

¹ *T. R. Gruber*: È un innovatore nelle tecnologie che cercano di emulare l'intelligenza umana. Lavora nel settore della creazione di ambienti regolati da intelligenza collettiva. Nella Stanford University è stato uno dei pionieri per l'uso del Web per la condivisione e la collaborazione. Ha avviato il “DARPA Knowledge Sharing Library”, uno scambio basato su Web per ontologie, software e base di conoscenza.

la relazione “parte-di”. Quindi l’ontologia non dà una descrizione di cosa esiste e cosa non esiste, ma crea una base dati contenitore (Cfr. [16]).

La distinzione tra il concetto filosofico e l’applicazione informatica si basa sul dominio di applicazione di un’ontologia. Mentre in filosofia si cerca di dare una spiegazione a ciò che è in senso assoluto, nell’ambito pratico, soprattutto in quello informatico, si cerca di dividere il macro-problema in sottoproblemi cercando di creare una piccola ontologia che rispecchi la descrizione del nostro scenario. Sono stati proposti dei linguaggi per la rappresentazione di ontologie, tra i quali RDF ed OWL.

5.1 Resource Description Framework

In origine il World Wide Web è stato pensato per essere utilizzato dagli uomini, quindi i dati in esso contenuti sono stati costruiti in modo da essere di facile comprensione per l’essere umano. Sebbene ogni elemento possa essere leggibile dalle macchine, non sempre i contenuti possono essere comprensibili da quest’ultime. A causa della grande quantità di dati presenti sul web è difficile automatizzare manualmente qualsiasi cosa su di esso. Si cerca quindi di ovviare a questo problema e il W3C² propone l’utilizzo dei metadati per descrivere quanto contenuto in una pagina web. I metadati in questione sono dei “dati che descrivono i dati” ossia può essere ad esempio un catalogo di una biblioteca, che descrive le pubblicazioni.

L’RDF, Resource Description Framework, nasce come linguaggio alla base per il trattamento dei metadati fornendo quindi interoperabilità tra le applicazioni che scambiano informazioni sul Web (Cfr. [30]). RDF principalmente si concentra sui metodi che consentono l’elaborazione automatica di risorse Web. Lo scopo principale di questo “framework” è realizzare un meccanismo pronto a descrivere le risorse senza essere basato su qualche dominio particolare di applicazione e senza definire a priori la semantica di qualche dominio di applicazione. Da quanto detto si evince che la definizione del meccanismo deve essere neutrale rispetto ai domini, ma allo stesso tempo questo deve essere adattabile alla descrizione di qualsiasi dominio. L’RDF è una delle proposte che sono alla base del web semantico.

Il web semantico è l’ambiente in cui i documenti pubblicati, ad esempio le pagine HTML, sono associati a delle informazioni e dei dati che ne specificano il contenuto in un formato che può essere interpretato e interrogato da delle macchine. Come si può notare, questi dati aggiuntivi sono proprio i metadati a cui si è accennato in

² W3C: associazione nata con lo scopo di migliorare gli esistenti protocolli e linguaggi per il WWW e di aiutare il Web a sviluppare tutte le sue potenzialità .

precedenza. Grazie a questi metadati comunque si prospetta ad esempio di avere in futuro delle ricerche molto evolute rispetto alle attuali che, partendo da semplici parole chiave, potranno costruire una rete di relazioni e interconnessioni più sofisticate rispetto al semplice link ipertestuale.

5.1.1 Principi base di RDF

Il Resource Description Framework si basa su tre principi chiave:

- Qualsiasi cosa può essere identificata da una URI, ossia una Uniform Resource Identifier
- Il principio del least power che consiste nel definire un linguaggio meno espansivo per definire una qualunque cosa
- Qualsiasi cosa può dire qualcosa su qualunque cosa.

Qualsiasi cosa descritta da RDF viene chiamata risorsa, le quali corrispondono a delle tradizionali coppie attributo-valore. Le stesse proprietà rappresentano anche relazioni tra risorse, perciò il modello risultante di RDF può paragonarsi a un diagramma entità-relazione. Come anticipato nei punti precedenti, il modello RDF è un modo di rappresentare espressioni RDF indipendentemente dalla sintassi utilizzata. L'equivalenza di due espressioni RDF si ha solamente nel caso in cui le loro rappresentazioni del modello dei dati sono uguali. Questa relazione permette di variare sintatticamente l'espressione, senza alterarne il significato. Il modello di base dei dati è costruito su tre tipi di oggetto:

Risorsa	Qualsiasi cosa descritta utilizzando delle risorse RDF viene definita come risorsa. Questa può essere ad esempio un'intera pagina web o una parte di quest'ultima. Non necessariamente una risorsa descritta da RDF deve appartenere al web, può essere qualsiasi cosa. Ogni risorsa è identificata da una URI, un identificatore univoco che può essere nella forma di una URL ³ o di una URN ⁴ .
---------	--

³ *URL*: Uniform Resource Locator. È una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa in Internet, come un documento o un'immagine.

⁴ *URN*: Uniform Resource Name. È un tipo di Uniform Resource Identifier che identifica una risorsa mediante un “nome” in un particolare dominio di nomi detto “namespace”.

Proprietà	Una proprietà è una caratteristica o una relazione utilizzata per descrivere una risorsa. Ogni proprietà contiene un significato proprio in cui vengono definiti i valori permessi, i tipi di risorse a cui può riferirsi e specifica altresì la relazione che assume con le altre proprietà .
Asserzione	L'unione di risorsa, proprietà e valore riferito alla proprietà stessa compone un'asserzione RDF. Queste tre parti compongono il classico schema <i>soggetto - predicato - oggetto</i> . L'oggetto, ossia l'asserzione, può essere un'altra proprietà oppure un letterale: nel primo caso sarà la URI della risorsa mentre nel caso del letterale sarà una stringa di caratteri o un elemento primitivo definito nell'XML.

5.2 Web Ontology Language

È un'estensione del linguaggio RDF, creato per rappresentare ontologie. Ha avuto vita nel progetto Web semantico. In esso si cerca di associare ai contenuti classici del web come documenti html, file, immagini, anche descrizioni che permettono di risalire all'autore, alla data di creazione e ad una serie di proprietà definite in un'ontologia.

In OWL si è cercato di rappresentare la logica predicativa del primo ordine più di quanto potesse fare il linguaggio RDF. In esso, infatti, la logica utilizzata è molto contenuta, si limita a regole di sussunzione. L'OWL, invece, ha cercato di utilizzare un sottoinsieme della logica formale o Aristotelica, chiamata logica descrittiva. In essa sono definiti i concetti rilevanti per quel dominio e, di seguito, utilizzando questi concetti, si specificano le proprietà degli oggetti e degli individui appartenenti al dominio (Cfr. [28]).

Possiamo definire tre principali novità della logica utilizzata da OWL nella rappresentazione del dominio rispetto a RDF. Prendono il nome di relazione:

- equivalenza, si intende la possibilità di poter affermare che due o più URI rappresentano lo stesso elemento;
- inversa, si intende la possibilità di dire che se è vero (soggetto, predicato, oggetto), allora è anche vero (oggetto, predicato inverso, soggetto);

- derivazione, si intende la possibilità di poter affermare che se A è relazionato a B, B con C allora A è ha un'inferenza con C.

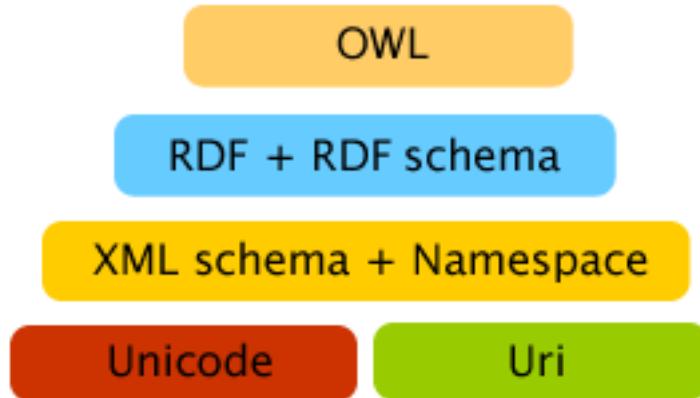


Figura 5.1. Livelli di astrazione del web semantico.

Capitolo 6

Architettura di sistema

In questo capitolo si affronterà la scelta architettonica utilizzata per prelevare la posta elettronica dal generico MTA, le tecniche di classificazione mirate a conservare l'entropia del messaggio e, di conseguenza, a capire il contenuto della stessa per classificarle fino a gestire l'archiviazione delle informazioni di elaborazioni e l'esposizione dei contenuti verso l'utente.

6.1 Protocolli di comunicazione

I messaggi di posta elettronica risiedono su delle macchine che non sono di proprietà dell'utente, per cui è necessario un meccanismo che permetta il recupero di questi dati. In modo particolare, allo stato attuale delle cose, sono presenti due protocolli che svolgono questo compito e sono degli standard effettivi: il POP e l'IMAP. Gli aspetti tecnici relativi a tali protocolli sono stati descritti in 2.4.1 e 2.4.2. Nello sviluppo dell'applicativo si è deciso di concentrarsi su uno dei due protocolli e in particolar modo di scegliere quello che permetteva una maggiore flessibilità per eventuali sviluppi futuri. La scelta è quindi ricaduta sull'IMAP che permette di avere una sincronizzazione con gli elementi in locale, ma soprattutto potrebbe permettere un'eventuale memorizzazione di dati direttamente sul server che si occupa dell'archiviazione dei messaggi. In questo modo potrebbe essere sfruttato lo spazio dedicato all'utente, non solo con i messaggi di posta elettronica giunti nella mailbox, ma altresì con varie informazioni di classificazione. Si può decidere di archiviare queste informazioni con l'uso dei “flag”, i quali possono rendere il lavoro delle applicazioni più flessibile senza dover ricorrere alle cartelle. Il POP, invece, offre solo un meccanismo di “download” dei messaggi presenti nella casella di posta dell'utente e questo ne limita il suo utilizzo per gli scopi previsti dall'infrastruttura.

6.2 Classificazione del testo

Classificare documenti in formato elettronico implica la generazione di categorie attraverso una fase di apprendimento, nella quale vengono forniti al sistema dei dati utili ai fini della comprensione della classificazione stessa. Il documento, per poter essere classificato, deve essere trasformato in una rappresentazione comprensibile all'elaboratore, ad esempio in una rappresentazione vettoriale estraendo gli attributi che sono più significativi, ossia che sono più rappresentativi all'interno del documento. Il tipo di classificazione si può suddividere in tre tipi:

- binaria;
- multiclasse;
- multipla.

La classificazione binaria si può usare quando si deve scegliere tra due classi possibili. Brevemente si può affermare che la classificazione binaria segue lo schema *o questo o quello*. Lo scenario più conosciuto di classificazione binaria nella posta elettronica è scoprire se un messaggio arrivato nella nostra mailbox è spam o non spam. Quando si effettuerà la fase di apprendimento si dovranno suddividere gli esempi in due classi, quelli appartenenti alla classe di classificazione e quelli che non vi appartengono.

La classificazione multiclasse altro non è che una generalizzazione del modello binario descritto in precedenza, dove però occorre scegliere una classe tra molte. Visto che molti metodi di classificazione sono limitati al caso binario oppure la loro implementazione multi-classe è poco efficiente, normalmente la classificazione multiclasse si ottiene utilizzando un insieme di classificatori binari. In particolare sono possibili due approcci per classificare multiclasse utilizzando i classificatori binari:

one against the rest : date n classi di classificazione, si costruiscono altrettanti classificatori binari. Il classificatore i -esimo divide tra gli elementi appartenenti alla classe i e quelli che non vi appartengono. In output si sceglierà la classe che ha ottenuto il peso più alto. Tale metodo è giustificato dalla regola di Bayes.

one against one : vengono generati $\frac{l(l-1)}{2}$ classificatori binari e ognuno di questi classificatori sceglie tra ogni coppia possibile di classi. Per scegliere la classe finale si utilizzano metodi come, ad esempio, la votazione a maggioranza.

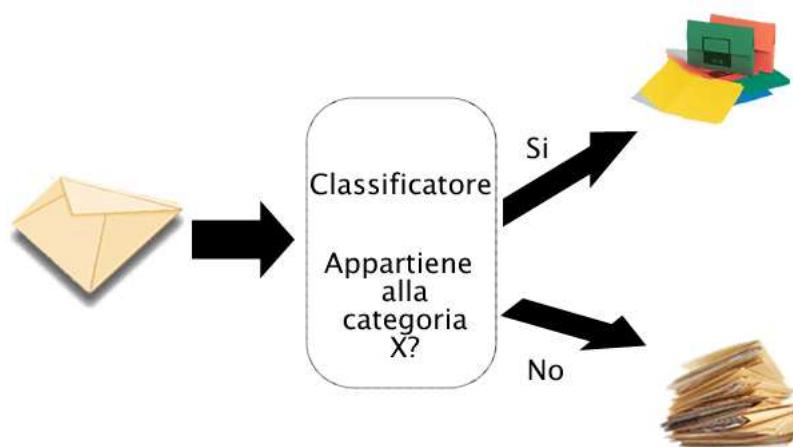


Figura 6.1. Esempio di classificazione binaria

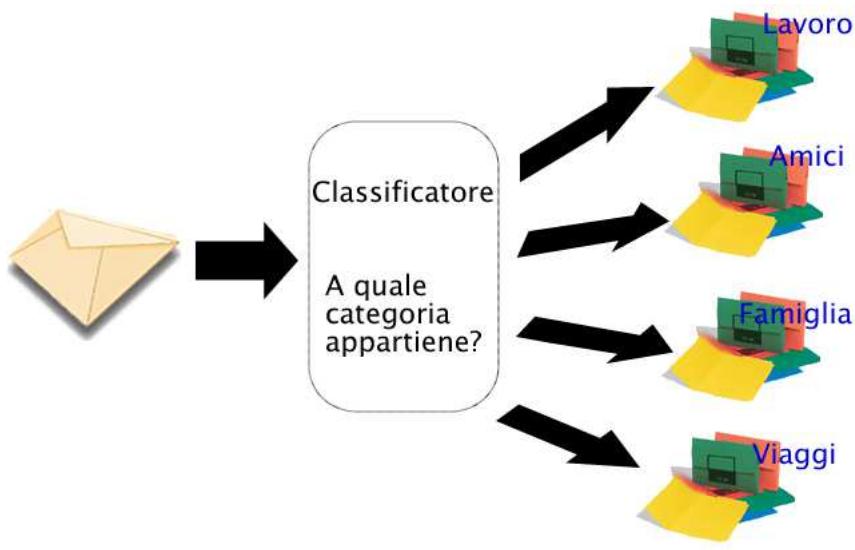


Figura 6.2. Esempio di classificazione multipla

La classificazione multipla si utilizza quando si ha la necessità o la possibilità che un documento possa appartenere a nessuna, una o a molte tra le classi possibili. Si

ha quindi in uscita un vettore di n elementi, dove per ogni classe si ha un valore di appartenenza del documento alla classe in questione. Anche in questo caso si può utilizzare un insieme di classificatori binari indipendenti uno dall’altro. Ogni classe viene trattata indipendentemente dalle altre e quindi il suo classificatore lavora parallelamente con quelli delle altre classi. Quindi si utilizza una classificazione del tipo *one against the rest* descritta in precedenza per il caso multiclass. Questo tipo di approccio assume però il fatto che le classi siano indipendenti una dall’altra e non sempre questo può essere vero. Ad esempio in una modello gerarchico questo tipo di assunzione è falsa.

6.3 Apprendimento

Data una generica rete logica, l’apprendimento si definisce supervisionato quando vengono date alla rete un insieme di informazioni, chiamate “*training set*”, preparate da un supervisore esterno, conoscendo già a priori l’uscita. Formalizzando il tutto, dato un generico ingresso i_n , si calcola l’uscita o_n ; per l’input i_n si apprende l’errore che è dato dalla differenza tra $o_n - \bar{o}_n$, quest’ultimo chiamato valore atteso. Ciò serve a supervisionare l’apprendimento per far capire alla rete quanto si sbaglia nel calcolare quell’output. Pertanto è possibile modificare il peso dell’apprendimento cercando di ridurre l’errore.

L’apprendimento si definisce non supervisionato quando alla rete vengono presentati i valori di input e la rete li suddivide autonomamente in gruppi usando misure di similarità, senza usare confronti con output noti e cercando di mettere input simili nello stesso gruppo. È un apprendimento autonomo e non c’è nessun esperto del dominio a cui è demandato il compito di controllore esterno.

Analizzando il funzionamento del filtro “Spam and Ham” creato da John Graham-Cumming(Cfr. [15]), si può notare come presi un insieme di dati venga costruito il “*training set*” il quale rimane l’unica base di conoscenza sugli esempi positivi e negativi. Un utilizzo di questo tipo può essere un’ottima possibilità, ma limita di molto le capacità della macchina di apprendere nuove informazioni dal’insieme di dati in ingresso. Utilizzando la definizione di “*machine learning*”, l’obiettivo è quello di imparare dagli esempi di classificazione. Un grosso problema che si apre è quello di decidere chi è l’esperto del dominio in grado di associare ad un messaggio la sua esatta classificazione. Lasciando decidere al filtro si ha il rischio di avere dei casi limiti come divergenze sul set di apprendimento. Oppure, nel caso di classi che contengono un numero diverso di mail, questo può condurre ad un’associazione che è condizionata dal loro numero nella base dati e meno dal contenuto. Ad esempio, si è preso come riferimento tre insiemi C_1 , C_2 , C_3 e partendo con un dominio di conoscenza che prevede un numero di mail di esempio pari a cinque in tutte le classi. Se le mail da classificare vengono associate a una classe in particolare, si creano degli

sbilanciamenti di classificazione per le mail che non appartengono alla stessa classe. Di conseguenza si deve cercare di creare un apprendimento supervisionato, il quale deve tendere a massimizzare le prestazioni del classificatore favorendolo mediante la crescita della sua conoscenza, ma allo stesso tempo non porti alla divergenza e quindi all'impossibilità di classificazione. Si è deciso di demandare all'utente, utilizzatore del servizio, di svolgere il ruolo di esperto nel dominio. Infatti, egli conosce l'uscita in base al generico ingresso quindi è in grado di effettuare un ragionamento di correttezza sul dominio di raggruppamento, dato l'errore ottenuto. In tal modo si modifica la base di conoscenza del filtro, andando così a migliorare le prestazioni di classificazione.

6.4 Filtri di classificazione

Una mail, a differenza di un generico documento testuale, presenta vari livelli entropici i quali possono condizionare la scelta di un utente di associarla ad una categoria. Si supponga di ricevere un messaggio di posta; esso presenta all'utilizzatore comune almeno due elementi distintivi: il mittente ed il corpo del messaggio. Si può

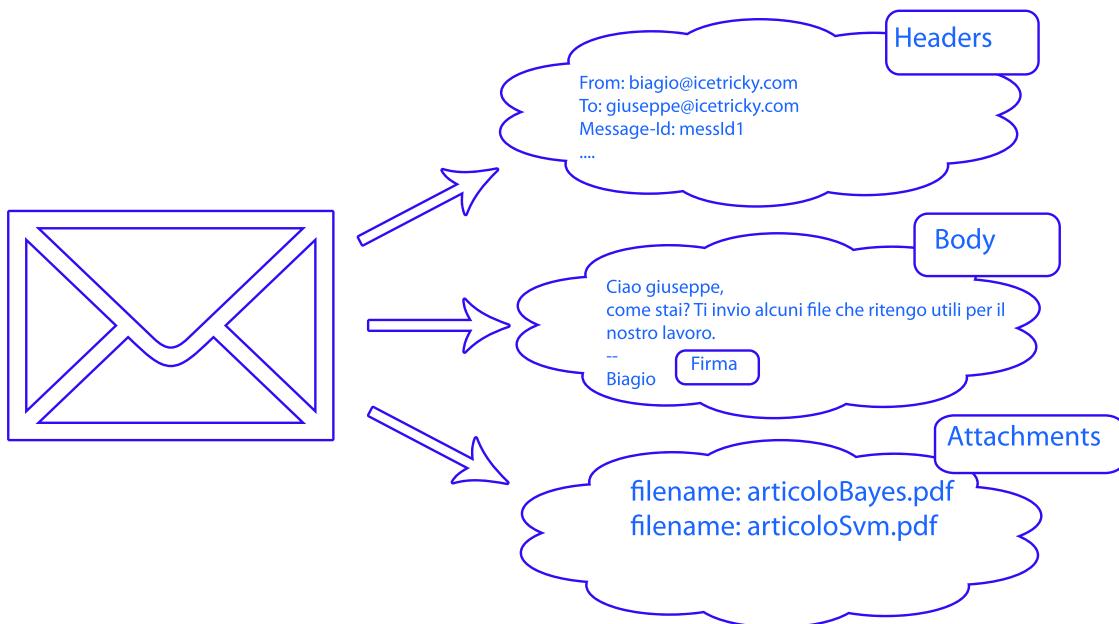


Figura 6.3. Livelli entropici di una mail

quindi effettuare una prima classificazione manuale in base al mittente oppure in base al contenuto della mail vera e propria. Si può scomporlo in due macroblocchi, il primo rappresenta la prima parte del messaggio, quella che generalmente non è

visibile all’utente, gli “*hearders*” e il suo contenuto, il “*body*”. Quest’ultimo contiene il testo, generalmente in chiaro, ossia leggibile direttamente dall’utente senza successiva elaborazione rif. 6.5.1. Si è deciso in prima analisi di effettuare delle analisi di classificazione sul corpo della mail, utilizzando così i filtri in uno scenario completo con un’alta occorrenze di parole in più insiemi di classificazione.

6.4.1 Bayes

Come descritto nel cap. 3, un generico classificatore di Bayes può avere dei problemi di convergenza, in quanto è possibile ricadere in problemi non risolubili. Si è deciso di seguire l’implementazione del “Naive Bayes”, la quale opera delle assunzioni sull’indipendenza statistica delle parole presenti in una generica conversazione. Questa assunzione ci permette di ridurre la complessità del problema, dovendo ricorrere a semplici prodotti. Anche se questa restrizione rappresenta un fattore limitante (basti pensare che le parole in una generica lingua sono tra loro correlate, dipendenti dal contesto che le precede e che le segue), le prestazioni che si possono raggiungere con questi classificatori sono ottime in rapporto alla velocità stessa dell’elaborazione. Si può affermare che questa applicazione sia un ottimo compromesso tra correttezza nei risultati e prestazioni in termine di velocità. Il classificatore di Bayes opera un’analisi solo del testo della mail e, in base agli insiemi di conoscenza costruiti in fase di “learning”, esso opera un’analisi sul numero di occorrenze di parole simili tra il “*body*” delle mail e quelle che ha dentro la sua base dati.

Nelle tabelle in basso vi sono degli esempi di classificazione, attraverso i quali si è data una dimostrazione del funzionamento a regime del classificatore. Per semplicità di visualizzazione, si è deciso di creare due categorie: Amici e Tesi, le quali a loro volta contengono due messaggi. La fase di “*training*” per un filtro bayesiano consiste nel leggere le parole di ciascuna mail, inserirle in opportune categorie e costruire un albero delle occorrenze. La fase di classificazione viene effettuata sull’analisi di una quinta mail. Il ruolo ultimo del classificatore è quello di decidere a quale delle due categorie il messaggio appartiene. La rappresentazione scelta è quella a livello insiemistico, ossia data una generica mail, si è costruito un albero di parole e occorrenze delle stesse dentro l’insieme da analizzare. È possibile notare come la punteggiatura è stata esclusa dall’insieme considerato, poiché può rappresentare un’informazione ridondante. Inoltre si è utilizzato il “*case-insensitive*”, quindi le parole maiuscole vengono maneggiate come se fossero minuscole. Nelle fig. 6.4 e 6.5 vi è la rappresentazione grafica degli insiemi di addestramento, mentre nelle tab. 6.4.2 e 6.4.2 l’analisi delle occorrenze dell’insieme.

È possibile notare la presenza di parole che nella lingua italiana sono classificate come congiunzioni. Esse, in prima analisi, possono rappresentare il riempimento di una frase e generalmente la loro presenza non modifica l’entropia di una discussione o di una frase.

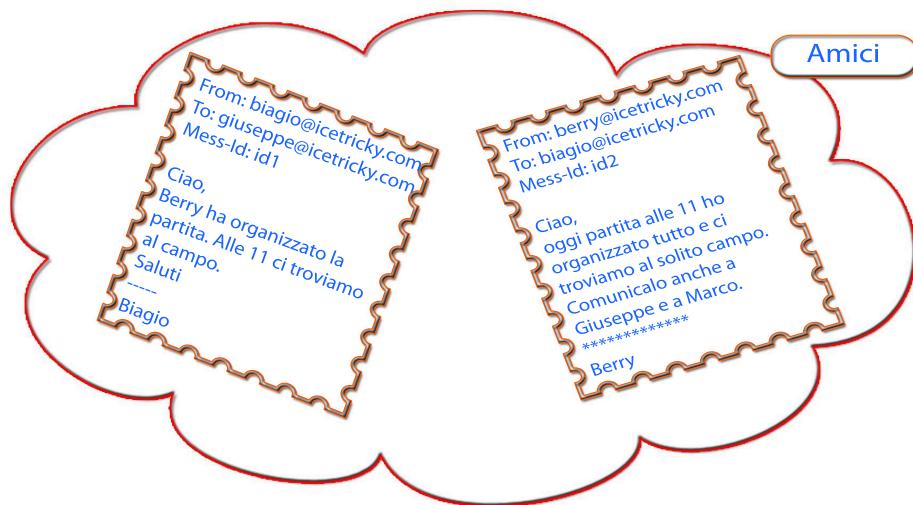


Figura 6.4. Training Set Amici

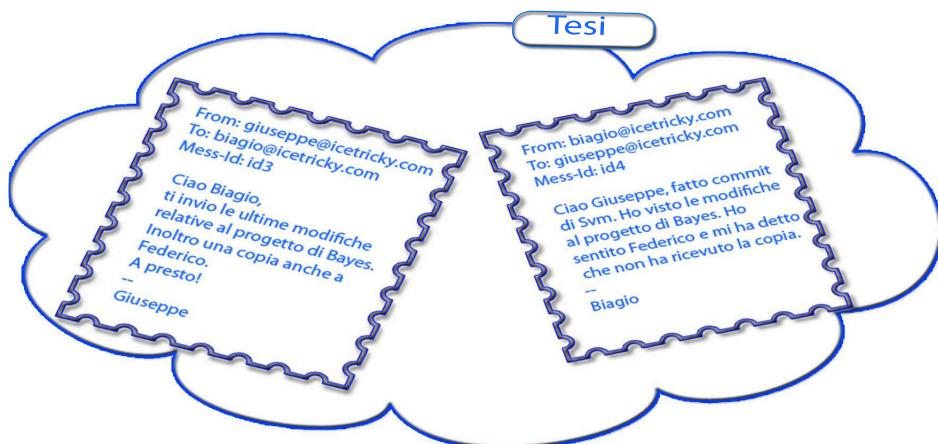


Figura 6.5. Training Set Tesi

Possiamo definire queste parole “stopwords”¹, nel par. 6.5.2 verrà data maggiore attenzione al problema. Gli esempi trattati in questo paragrafo non prenderanno in

¹ *stopwords*: il termine fu coniato da Hans Peter Luhn, uno dei pionieri dell’ “information retrieval”. Per un umano la cancellazione di termini ridondanti è un processo automatico, mentre per sistemi automatizzati la cancellazione di questi elementi può comportare miglioramenti di prestazioni.

<i>word</i>	<i>occurrency</i>
ciao	2
berry	2
...	...
marco	1
<i>Totale</i>	36

Tabella 6.1. Category Amici, riferimento alla tabella completa in appendice B.5

<i>word</i>	<i>occurrency</i>
ciao	2
biagio	2
...	...
non	1
ricevuto	1
<i>Totale</i>	48

Tabella 6.2. Category Tesi, riferimento alla tabella completa in appendice B.6

considerazione la cancellazione delle “stopwords”.

La mail da classificare è rappresentata in figura 6.6.

<i>word</i>	<i>occurrency</i>	<i>peso_{Amici}</i>	<i>peso_{Tesi}</i>
ciao	1	2	2
biagio	1	0	2
...
mando	1	0	0
un	1	0	0
commit	1	0	1
giuseppe	1	1	2

Tabella 6.3. Simulazione di arrivo di nuova mail, riferimento alla tabella completa in appendice B.7

Nella tabella 6.3 si riporta l’analisi statistica che svolge il filtro bayesiano quando deve effettuare una generica classificazione. Nella colonna di sinistra si riporta il nome della categoria, in quella centrale il numero di parole dell’insieme preso in

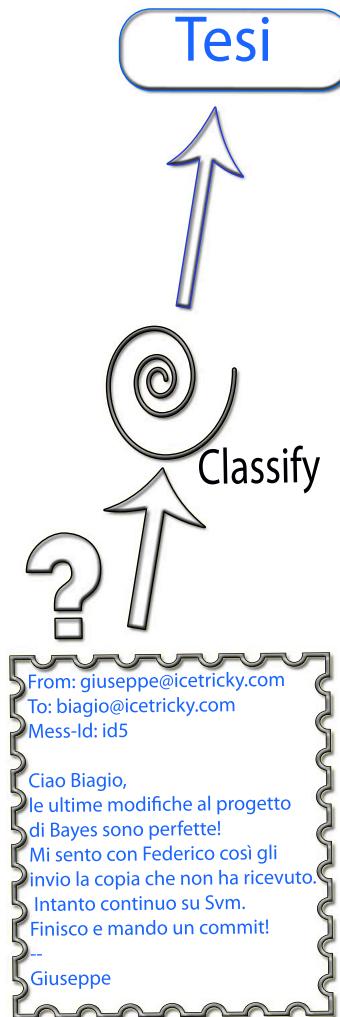


Figura 6.6. Mail da classificare

considerazione e nell’ultima le occorrenze delle parole, ossia il peso di quella classe nella decisione di etichettare la mail. Come si è discusso nel cap. 3 si opera un’assunzione d’indipendenza tra le parole, questo prende il nome di Naive Bayes. Usando

<i>Category</i>	<i>#Words</i>	<i>Occurrency</i>
Amici	36	9
Tesi	48	34

Tabella 6.4. Analisi statistica

la regola di Graham-Cumming[15], suddividiamo gli insiemi in “ham” e “not-ham”. Applicando la seguente regola si ottiene un valore percentuale che esprime una considerazione statistica sull’appartenenza di quel documento, mail, alla generica classe.

$$P_{ham} = \frac{P_{ham}}{P_{ham} + P_{not-ham}} \quad (6.1)$$

mentre:

$$P_{not-ham} = \frac{P_{not-ham}}{P_{ham} + P_{not-ham}} \quad (6.2)$$

Il classificatore secondo l’esempio presentato assegna una maggiore probabilità di

<i>Category</i>	<i>Score</i>
Amici	26,09%
Tesi	73,91%

Tabella 6.5. Classificazione

appartenenza alla classe Tesi, come era prevedibile dal suo contenuto, mentre un valore molto basso per la classe Amici. Queste prestazioni possono essere migliorate con l’utilizzo di strumenti di eliminazione delle “stopwords”, che sarà affrontato nel par. 6.5.2.

6.4.2 N-grams Bayes

Per ridurre al minimo la perplessità dovuta al modello del “Naive Bayes”, come descritto nel par. 3.4, si è cercato di costruire una catena di Markov, con la quale è possibile rappresentare le dipendenze tra le parole mediante l’utilizzo degli n-gram². Anche se questa tecnica è decisamente più onerosa in termini di tempo di calcolo, essa può garantire prestazioni migliori in termine di classificazione (Cfr. [4]), dovuto all’eliminazione dell’assunzione di indipendenza delle parole. Il classificatore N-Gram ha così bisogno di una fase di “learning” differente rispetto al classificatore del “Naive Bayes”, in quanto deve scomporre il corpo dei messaggi in “n-gram” e costruire così l’insieme di questi in ogni categoria. La classificazione vera e propria utilizzerà l’algoritmo del “Naive Bayes”, basandosi, però, sugli n-gram. Nelle tabelle in basso vi è un esempio di “training” basato sulle mail delle fig. 6.4 e 6.5 utilizzando un 4-gram. Nelle tab. 6.4.2 e 6.4.2 vi sono alcuni n-gram indicati tra virgolette.

²n-gram: parte di una parola. Generalmente il termine n indica la lunghezza stessa del “gram” della parola stessa.

<i>word</i>	<i>occurrency</i>
“ 11 ”	2
“ a g”	1
“ a m”	1
...	
ci	2
cia	2
ciao	2
iao	2
...	

Tabella 6.6. Category Amici

Si è utilizzata questa notazione per dare risalto alla presenza di spazi. Infatti, essi, sono una parte importantissima nel modello linguistico per ogni tipo di lingua. Ad essi va aggiunta la punteggiatura. Negli esempi riportati, per motivi di semplicità, non è presa in considerazione. Infine, effettuando la stessa scomposizione in n-gram

<i>word</i>	<i>occurrency</i>
“ a f”	1
“ a p”	1
“ al ”	2
...	
ci	2
cia	2
ciao	2
iao	2
eppe	2
...	

Tabella 6.7. Category Tesi

anche per la mail in fig. 6.6 ed applicando poi le formule 6.1 e 6.2, si ottiene un valore in percentuale con il quale si può affermare quale delle due categorie meglio rappresenta la nuova mail.

6.4.3 Support Vector Machine

A differenza dei classificatori descritti precedentemente le Support Vector Machine non si basano direttamente su un insieme di parole che descrivono una categoria di classificazione, ma cercano, attraverso l'uso di funzioni matematiche, di identificare ogni singolo elemento all'interno del piano e, in base ai vari esempi di apprendimento, di individuare delle zone di appartenenza per ogni singola categoria di classificazione. Queste, per essere identificabili in uno spazio geometrico, necessitano di una ulteriore astrazione, o meglio di una indicizzazione, che permetta un collegamento univoco tra una parola e un numero. Questo legame può essere superato attraverso l'utilizzo di un dizionario.

Questo collegamento può essere instaurato attraverso l'utilizzo di un vocabolario in cui sono memorizzate le parole conosciute a cui viene associato, per ognuna di esse, un identificativo numerico univoco. Un classificatore costruito attraverso l'uso di un filtro SVM, da un punto di vista teorico, permette di avere classificazioni migliori rispetto ai filtri bayesiani descritti in precedenza. I risultati migliori, però, vanno a discapito delle prestazioni. Infatti un'elaborazione con SVM necessita di una maggiore quantità di tempo macchina rispetto ai filtri bayesiani e a questo si aggiungono le tempistiche necessarie per l'ulteriore trasformazione del testo nel formato richiesto dal classificatore.

Anche nel filtro SVM i messaggi vengono trattati come “case-unsentive” e la punteggiatura viene omessa ai fini del riconoscimento. In ogni messaggio si contano le occorrenze di ogni singola parola in modo da valutarne il peso all'interno della classificazione. Possiamo prendere ad esempio il messaggio con message-id pari a “id2” all'interno della figura 6.4. Verrà costruita una struttura come riportato in tabella 6.8. Ogni singola parola, inoltre, verrà sostituita dal corrispondente indice presente

word	occurrency
ciao	1
oggi	1
partita	1
...	...

Tabella 6.8. Messaggio della categoria Amici con message-id pari a “id2”, riferimento alla tabella completa in appendice B.5

nella base dati e sarà introdotta da un numero che indica la categoria di classificazione (0 nel caso in cui il messaggio sia da classificare). Nella tabella 6.9 viene mostrato un esempio di vocabolario. Andando quindi a recuperare le informazioni

1	e
2	Berry
3	ha
4	organizzato
5	la
6	partita
...	...

Tabella 6.9. Vocabolario

dal vocabolario, i messaggi d'esempio descritti prima in 6.4.1 possono avere una rappresentazione delle feature come quella riportata in tabella 6.10 . In base a questa rappresentazione il filtro SVM individuerà in quale iperpiano si trova e restituirà quindi la categoria di appartenenza. Ad esempio il messaggio con message-id “id1”

feature	1	2	3	4	5	6	7	8	9	10	...	22	23	24
id1	-	1	1	1	-	1	1	1	1	1	...	-	-	-
id2	2	1	-	1	-	1	1	1	1	1	...	-	-	-
id3	-	-	-	-	-	-	-	-	-	1	...	1	1	1

Tabella 6.10. Esempi di formato svm.

avrà una rappresentazione del tipo 0 2:1 3:1 4:1... .

Come già descritto nei classificatori bayesiani, anche in questo caso vi è la possibilità di utilizzare dei classificatori binary in parallelo, che dovrebbero fornire risposte più precise, oppure un unico classificatore multi-classe che ha la conoscenza di tutte le categorie presenti. I dettagli riguardanti le scelte adottate per queste soluzioni sono spiegati nel cap. 7. Ogni categoria dovrà necessariamente contenere al suo interno un insieme di esempi che sono utilizzati per l'addestramento. Dopo la fase di apprendimento, il filtro produce un modello che identifica l'iperpiano che suddivide lo spazio in zone rappresentanti ognuna una categoria differente di classificazione. Quindi, riprendendo l'esempio precedente descritto in 6.4.1, il filtro riceve in ingresso la rappresentazione del messaggio da classificare. Il compito quindi si limiterà a calcolare dove si colloca all'interno dello spazio, ad identificare a quale categoria appartiene ed eventualmente fornire un valore di probabilità che indica il grado di affidabilità del risultato. Anche in questo caso si rimanda alle sezioni seguenti per un'analisi dei miglioramenti possibili grazie alla manipolazione del corpo del messaggio.

6.5 Gestione dei messaggi

Ogni messaggio di posta contiene al suo interno diversi tipi di contenuti che possono essere più o meno interessanti ai fini della classificazione. In questa sezione vengono analizzati tali aspetti.

6.5.1 Corpo del messaggio

Generalmente lo strato centrale della mail si compone di testo in chiaro, ossia leggibile direttamente dall'utente senza eventuale manipolazione di un interprete software. Negli ultimi anni si è molto diffuso l'utilizzo di codice “HTML” nel body. Questo fornisce all'utente la possibilità di aggiungere del contenuto grafico che rende la mail stessa più bella da vedere e migliora la leggibilità. Una mail che utilizza del codice di formattazione dati “HTML” presenta uno strato aggiuntivo rispetto alla mail chiamata “plain text” ossia con testo direttamente leggibile. Secondo lo standard questo strato fa parte del pacchetto MIME.

I classificatori sopra enunciati hanno bisogno di lavorare su testo in chiaro, di conseguenza occorre effettuare una pulizia o meglio una cancellazione dei contenuti non direttamente leggibili, effettuando così un'elaborazione (anche detta “parsing”) che ci permette di preservare solo il contenuto testuale.

6.5.2 Riconoscimento lingua e le stopwords

In ogni lingua sono presenti caratteristiche particolari per rendere il discorso fluido e comprensibile. In particolar modo si fa uso di un certo insieme di parole che vengono utilizzate all'interno di ogni frase. Basti pensare a tutti le congiunzioni e gli articoli presenti nella lingua italiana. In realtà queste parole, se prese singolarmente, non contengono al loro interno un significato importante per il contenuto che si vuole esprimere, al contrario sono indispensabili per rendere leggibile e comprensibile il discorso a un umano. Quello che però si cerca di effettuare all'interno di un tentativo di classificazione automatica consiste nel ricercare, nel contenuto dei vari testi, delle caratteristiche comuni attraverso le quali si possa assumere una categoria di appartenenza. Queste parole prendono il nome di “stop words”, termine coniato da Hans Peter Luhn³ e in generale sono pronomi, congiunzioni, articoli e forme coniugate del verbo essere. In linea teorica questi elementi non dovrebbero avere in realtà considerevoli effetti sulla classificazione visto che si trovano in ogni documento, tuttavia alcune saranno più presenti e altre meno. Bisogna anche considerare il

³Hans Peter Luhn (1 luglio 1896 - 19 Agosto 1964): ingegnere presso l'IBM e creatore dell'algoritmo omonimo e dell'indicizzatore KWIC (Key Words In Context).

fatto che un messaggio di posta elettronica normalmente è un testo corto e quindi la presenza delle “stop words” potrebbe essere deviante ai fini della classificazione. Oltretutto, viste le considerazioni precedenti, possono essere rimosse dal testo del messaggio e lavorare quindi su insiemi di parole più piccoli, diminuendo quindi il carico di lavoro dei filtri.

Ogni lingua contiene un insieme di queste “stop words” per cui occorre conoscerle a priori affinchè si possano identificare all’interno dei messaggi di posta elettronica. Bisogna quindi esaminare prima di tutto con quale lingua sia stato scritto il documento e in seconda analisi andare a eliminare le “stop words” confrontandole da un elenco specifico della lingua in questione.

Si è visto, attraverso studi, che in una lingua le lettere si succedono secondo una specifica frequenza, ad esempio in italiano la lettera *a* ha una frequenza pari a 11.74%. Nel grafico in 6.7 si possono notare le varie frequenze per quanto riguarda l’italiano. Il grafico riportato in 6.9 mostra la frequenze delle lettere nella lingua

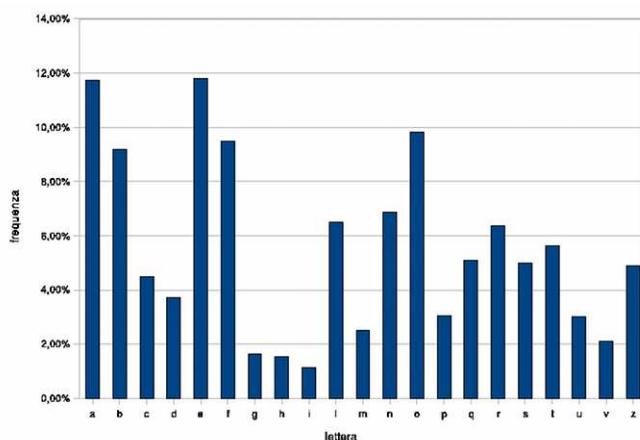


Figura 6.7. Frequenza delle lettere nella lingua italiana scritta.

inglese. Si può riconoscere la lingua in cui è scritto il testo conteggiando la frequenza di ogni singola lettera e in base a questo calcolare la percentuale di queste occorrenze. In questo modo si potrà andare a confrontare il risultato ottenuto con quello noto a priori del set di lingue che si vuole analizzare e, attraverso dei rapporti, si ottiene un risultato di ”somiglianza“ rispetto a una lingua, così si ricava la lingua alla quale si avvicina di più.

Un’altra particolare tecnica è quella che si basa sulla frequenza degli n-grams. Gli n-grams sono dei gruppi di lettere con cui si compongono le parole o meglio il testo in generale. Ad esempio analizzando la seguente frase

“il gatto rincorre il topo”

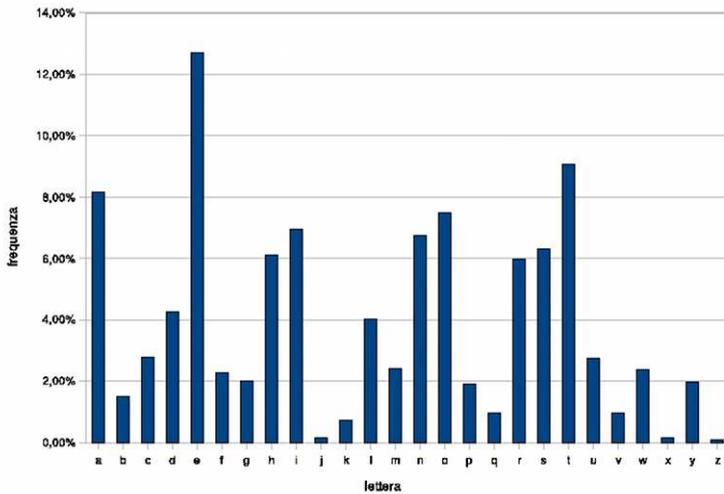


Figura 6.8. Frequenza delle lettere nella lingua scritta inglese.

gli n-grams di lunghezza 2 sono:

”il” - ”l” - ” g” - ”ga” - ”at” - ”tt” - ”to”

e così via. Si è visto che, come nel caso descritto precedentemente, le parole si compongono secondo una frequenza particolare per gli n-grams, in altri termini, in una lingua un n-gram sarà più ”probabile“ rispetto ad un altro, o meglio, sarà utilizzato di più nelle parole rispetto a qualche altro. Nell’implementazione del progetto si è deciso di appoggiarsi a questa particolare tecnica e per calcolare queste statistiche sugli n-grams si è deciso di prendere un testo per ogni singola lingua attraverso il quale analizzare il contenuto e calcolare gli n-grams e le frequenze a questi associati. In questo modo per aggiungere una lingua diventa molto semplice: sarà sufficiente utilizzare un file scritto nella lingua in questione. Naturalmente il testo utilizzato per apprendere le statistiche dovrà essere rappresentativo per la lingua in questione ed essere abbastanza corposo, altrimenti si rischierebbe di analizzare una lingua con dei dati non rappresentativi. Ad esempio, per analizzare la lingua italiana, si è preso un testo di un racconto che si compone di circa millequattrocento parole. Ovviamente questo calcolo verrà eseguito solamente una volta e i risultati saranno memorizzati in modo da non dover ricalcolarli tutte le volte, evitando quindi di eseguire delle operazioni molto costose che sarebbe inutile ripetere. Lo stesso discorso di scomposizione in n-grams e calcolo della frequenza di questi ultimi dovrà essere effettuato sul testo del messaggio da analizzare. Le statistiche che si ottengono da questa analisi vengono confrontate con quelle tipiche di ogni lingua e si calcola un punteggio per ogni lingua. Quando tutti i punteggi sono stati calcolati si prenderà quello superiore che sarà relativo alla lingua in cui è scritto il testo.

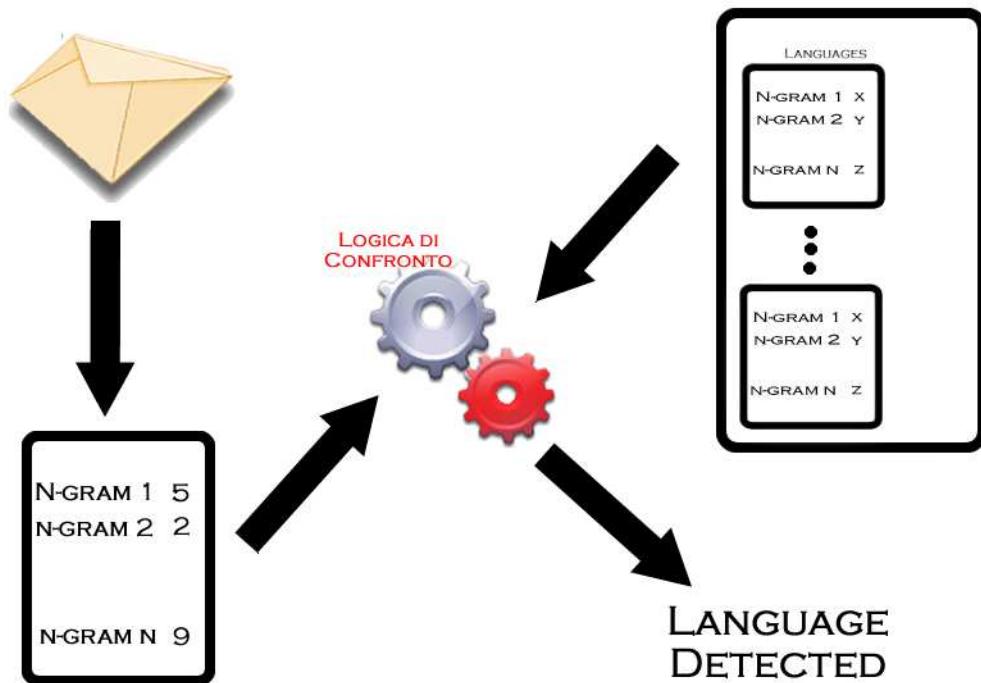


Figura 6.9. Motore di riconoscimento della lingua.

Nella tabella 6.12 riportiamo alcune delle “stop-words” tipiche della lingua italiana. Si prenda ad esempio il testo di uno dei messaggi analizzati in precedenza. Ad

a	agli	e	alcun	alla
al	ai	alquanto	altro	che
ciò	chissa	ciascun	con	cogli

Tabella 6.11. Alcune stop words della lingua italiana.

esempio il secondo messaggio contiene:

Ciao,
oggi partita alle 11 ho organizzato tutto e ci troviamo al solito campo.
Cumunicalo anche a Giuseppe e a Marco.

In totale il testo contiene diciannove parole differenti. Andando a pulire il messaggio otteniamo quanto riportato nella tabella ??: Si può notare che eliminando le “stop-

oggi	partita	11	organizzato	tutto
solito	campo	comunicalo	Giuseppe	Marco

Tabella 6.12. Alcune stop words della lingua italiana.

words” si contano solamente dieci parole, quindi si ha riduzione degli elementi da analizzare pari circa al 45%.

Un’altra strategia molto semplice per ridurre il numero delle “feature” prende il nome di “pruning” e consiste nell’eliminare le parole che compaiono nel training set al di sotto di una certa soglia. La scelta di un valore limite può essere decisiva in alcuni domini, ma occorre prestare attenzione ad alcuni particolari. Ad esempio una parola che compare una sola volta può essere determinante ai fini della classificazione se appartiene, ad esempio, al soggetto del messaggio. Nel caso della posta elettronica questa tecnica potrebbe essere applicata su più messaggi contemporaneamente perché, vista la breve lunghezza del testo di un singolo messaggio, nella maggior parte dei casi potrebbe andare ad eliminare termini che in realtà sono molto importanti ai fini della classificazione. Infatti in un testo di poche righe e immediato come quello di una mail è poco probabile che un termine venga ripetuto spesso. Invece, se si prendono molti elementi appartenenti allo stesso insieme di categorizzazione, i termini “chiave”, in linea di massima, vengono ripetuti più volte.

6.5.3 Stemming

Lo stemming è il processo utilizzato per ridurre parole derivate da una radice, anche detto “stem”. Questo procedimento permette di elaborare un insieme di parole più contenuto, in quanto si concentrano le occorrenze di termini derivati dalla stessa radice e che meglio possono contraddistinguere una categoria. Si prenda ad esempio tre parole in lingua inglese: *stem*, *stemmer* e *stemming*. La radice delle tre parole è “stem”. Effettuando l’operazione di riduzione nella forma radice, una categoria presenterebbe un maggiore peso dalla radice “stem” in quanto le varie occorrenze della radice dei testi da classificare conterebbero come parole che si trovano dentro l’insieme. Si prenda come esempio il termine inglese “stemmed”; esso è derivato dal termine “stem”. Se una categoria dell’insieme di conoscenza presenta il termine “stem”, il nuovo termine verrà considerato appartenente a quella categoria. Senza effettuare lo “stemming” questo non accadeva, in quanto la nuova parola introdotta era diversa da quella presente nella categoria.

Il primo “stemmer” pubblicato è stato scritto negli anni sessanta da Julie Beth

Lovins. Nel corso degli anni ci sono stati altri ricercatori che hanno avanzato algoritmi di “stemming”, ma l’unico inconveniente dovuto da questi algoritmi era quello di essere scritti per una lingua “ad-hoc”. Questo per diverse ragioni:

1. diversità lessicale delle lingue;
2. interesse diverso per sviluppare “stemmer” per alcune lingue.

6.5.4 Firme

Nella maggior parte dei messaggi di posta elettronica il mittente è solito inserire la propria firma. Normalmente questo dato non contiene informazione utile ai fini dell’analisi del testo e quindi si può pensare di eliminarlo dal testo del messaggio. Vari esempi si possono notare nella fig. 6.4 e nella fig. 6.5. Solitamente le firme consistono in una serie di caratteri come “*” o “-” seguiti da informazioni inserite dal mittente come ad esempio nome, cognome e numero di telefono. I programmi di posta elettronica sono dotati di opzioni automatiche che, se abilitate, inseriscono in coda al “body” la firma del messaggio quando questa viene inviata. Le firme, comunque, possono assumere qualsiasi forma e non esiste un modo per racchiuderle tutte. Lo standard MIME prevede che la firma sia preceduta dalla sequenza di caratteri “– CR+LF⁴”, quanto previsto per gli standard di cifratura come PGP o S/MIME. Ovviamente in un messaggio di posta elettronica la firma del mittente può anche essere assente.

Oltrettutto possono anche presentarsi dei dati ulteriori inseriti dai gestori che offrono servizi per la spedizione dei messaggi di posta. Solitamente sono delle pubblicità relative al gestore del servizio, il quale le incolla al fondo del messaggio. Anche queste informazioni non sono importanti ai fini della classificazione, anzi inseriscono una maggior quantità di dati che possono andare ad influenzare i risultati.

Si prenda ad esempio:

Gentile studente/ssa,
si invia in allegato un comunicato da parte dell’ E.Di.S.U
Cordiali saluti,

Ufficio Tasse e Diritto allo Studio
Servizio gestione Didattica
POLITECNICO DI TORINO
C.so Duca degli Abruzzi, 24
10129 TORINO (TO) - ITALY

⁴CR+LF: “carriage return and line field”, nuova linea e andata a capo

fax: +39.011.564.5947
e-mail: diritto.studio@polito.it
<http://didattica.polito.it>

Questo messaggio di posta ad esempio contiene più parole all'interno della firma di quante ne esprimono il contenuto. Risulta quindi facile capire che una eventuale classificazione, senza effettuare nessuna pulizia, sarà pilotata soprattutto in base al contenuto della firma. Questo può portare a classificare messaggi inviati dallo stesso mittente come appartenenti alla stessa categoria, quando in realtà essi hanno al loro interno contenuti molto differenti tra loro. In prima analisi, da un punto di vista teorico, sembra che effettuare una pulizia da questi dati possa portare dei benefici alla classificazione. In altri casi, invece, risulta essere svantaggioso in quanto potrebbero essere presenti dettagli che identificano il messaggio poiché contengono ad esempio, come anche nella mail precedente, l'indicazione di informazioni aggiuntive, come l'ufficio di provenienza o l'associazione di cui il mittente fa parte. Questi potrebbero rivelarsi dei validi aiuti se una categoria scelta dall'utente contenga messaggi esclusivamente provenienti da quell'ufficio o il processo di apprendimento è stato effettuato con pochi elementi. Un miglioramento delle prestazioni potrebbe quindi dipendere anche dal tipo di suddivisione delle categorie scelto dall'utente e da come è stato effettuato l'addestramento iniziale.

6.6 Rubrica

Nell'utilizzo del servizio di posta, spesso, si ricorre al concetto di persona. Essa, infatti, viene descritta da un indirizzo di posta, seguito da un'etichetta. Nello scambio di messaggi queste informazioni possono essere usate dal ricevente o mittente per archiviare una mail o un gruppo di esse. Queste informazioni si trovano nel livello più alto di un messaggio, riferimento alla fig. 6.3. Diventa importante, pertanto, effettuare una gestione degli indirizzi, costruendo un cammino che ci possa permettere di ricostruire i messaggi inviati o ricevuti in base alla conoscenza del mittente o destinatario. Un problema che si può riscontrare deriva dalla diversità di indirizzo che un generico utente può utilizzare. Infatti, si è cercato di astrarsi il più possibile dall'uguaglianza del campo indirizzo e si è cercato di applicare algoritmi riportati nella sezione successiva.

6.6.1 Levenshtein: distanza tra due espressioni

A volte può risultare utile riuscire a capire automaticamente se due stringhe sono simili tra loro tanto da esprimere lo stesso concetto. Inoltre gli utenti possono possedere più indirizzi di posta elettronica e in una rubrica indirizzi si vuole tenere traccia di ogni persona fisica memorizzando i vari indirizzi che utilizza. Normalmente, soprattutto nei programmi di posta elettronica, viene associato oltre all'indirizzo mail anche un “visualized name” che può essere ad esempio un soprannome o il nome completo del mittente. Ovviamente possono essere impostati differenti “visualized name” tra i vari indirizzi. Una tecnica che permette di individuare stringhe simili da loro consiste nel calcolare “la distanza di Levenshtein”. Tale calcolo mira a indicare quanto due sequenze di caratteri differiscono l'una dall'altra. La distanza indica quante operazioni elementari sono necessarie per trasformare la stringa A in B . Le operazioni effettuabili sono:

- cancellazione di un carattere,
- sostituzione di un carattere con un altro,
- inserimento di un carattere.

Prendiamo, ad esempio, due parole come “bar” e “biro”. La loro distanza di Levenshtein è pari a 2. Le operazioni sono

1. la sostituzione della a in i,
2. l'aggiunta del carattere o.

La distanza di Levenshtein ha limiti superiori ed inferiori. Il risultato è almeno la differenza di lunghezza tra le stringhe e comunque non supera mai la lunghezza della stringa più lunga. Si ottiene 0 solamente nel caso in cui le due stringhe siano identiche. Nel caso in cui le due stringhe siano di pari lunghezza, la distanza di Levenshtein non supporta la distanza di Hamming, invece se differiscono in lunghezza il limite superiore è la distanza di Hamming aumentata della differenza delle lunghezze. Con distanza di Hamming si intende il numero di posizioni in cui due stringhe differiscono tra loro, ossia si contano il numero di posizioni in cui i simboli differiscono tra loro. Naturalmente tale valore può essere calcolato solamente nel caso in cui le stringhe siano di egual misura.

Questi teoremi possono essere d'aiuto quando si cerca di automatizzare un riconoscimento automatico di appartenenza degli indirizzi alle persone. Prendiamo ad esempio due indirizzi come <biagiomeirone@gmail.com> e <biagio.meirone@yahoo.it>. Questi indirizzi probabilmente appartengono alla stessa persona visto che differiscono tra loro solamente per il carattere “.”, ovviamente escludendo quello che si trova dopo “@”. Se si calcola la distanza di Levenshtein è risulta essere pari a uno. Si può

quindi ragionevolmente assumere che questi indirizzi possono essere associati alla stessa persona nella rubrica. Considerazioni simili si possono effettuare sui “visualized name”. In alcuni casi, però, un semplice calcolo della distanza tra due stringhe può non bastare. Basti pensare al caso in cui si ha nome e cognome e nell’altro cognome e nome. Applicando le regole descritte in precedenza risulta che le due stringhe sono molto diverse tra di loro ma in realtà si tratta solo di due termini invertiti. Naturalmente, anche in questo caso, l’utente ha piena libertà di creare indirizzi mail e visualizzare quello che vuole, quindi si complica notevolmente il riconoscimento automatico di questi aspetti. Si possono comunque individuare delle soluzioni che possono funzionare nella maggioranza dei casi e soprattutto ristretti a un ambito specifico, ma comunque ad oggi non esiste una soluzione che produce i risultati desiderati nella totalità dei casi.

6.7 Base dati

Come è stato descritto nel cap. 5, un’ontologia permette una standardizzazione dei contenuti che vengono esposti sul web. Gli oggetti così creati possono essere riferiti da tutti coloro che volessero utilizzare gli elementi della ontologia stessa. Fornisce anche una maggiore flessibilità rispetto al salvataggio in pure tabelle relazionali, in quanto il dominio di conoscenza può essere esteso con più facilità. Altro importante comportamento di un’ontologia è la possibilità di operare ragionamenti sulla base dati.

Nella figura 6.10 vi è un esempio di ontologia. I nodi della rete sono le persone, il cui tipo è indicato nell’ontologia. Le inferenze tra i nodi rappresentano le relazioni tra le stesse persone. Si può notare come è possibile operare dei ragionamenti anche di tipo “sussunzione” cap. 5, applicando la logica formale di I livello. Ad esempio, si possono fare dei ragionamenti del tipo: Giuseppe “is friend of” Biagio, Biagio “is friend of” Daniele, allora si può assumere che ci sia una relazione tra Giuseppe e Daniele dello stesso tipo.

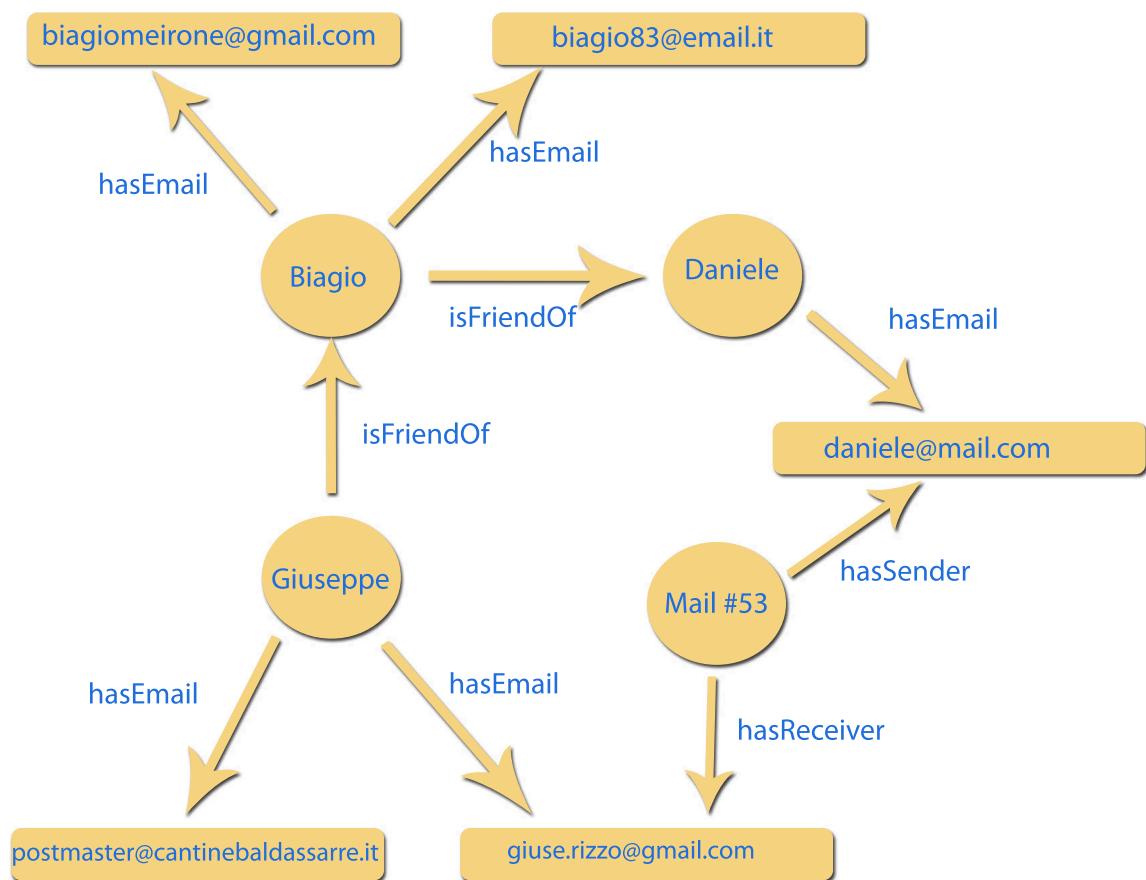


Figura 6.10. Esempio di ontologia

Capitolo 7

Sviluppo dell'applicativo

In questo capitolo saranno trattate le problematiche riscontrate nello sviluppo del progetto. L'implementazione ha riguardato i seguenti punti:

1. la creazione di una libreria IMAP in grado di collegarsi alla risorsa server, contenitore dei messaggi;
2. la creazione di “parsing” per la lettura dei contenuti informativi della mail. Come espresso nel par. 6.4, la mail è composta da livelli, i quali concorrono in maniera diversa alla categorizzazione della stessa. Pertanto si è ritenuto necessario creare un meccanismo che permetesse di suddividere i vari livelli entropici;
3. implementazione dei filtri di classificazione, nei quali si è deciso di accentrare tutte le problematiche di apprendimento, classificazione, riclassificazione e creazione dei tag;
4. una libreria in grado di gestire le informazioni riguardo i messaggi, la rubrica e le etichette;
5. l'implementazione di flussi di discorsi, anche chiamati nel seguito “Thread”;
6. si è, pertanto, affrontato il problema del salvataggio di queste informazioni, utilizzando una base dati RDF.
7. infine si è presentato il problema della formattazione e della presentazione dei risultati. Si è immaginato uno scenario di servizio web, con l'esposizione di contenuti prettamente dinamici su fogli xhtml.

Riassumendo è possibile affermare che, utilizzando uno schema a blocchi, gli input del progetto sviluppato sono le mail, mentre gli output sono i risultati, mentre la funzione che permette di associare gli ingressi alle uscite costituisce il blocco centrale o cuore dell'infrastruttura.

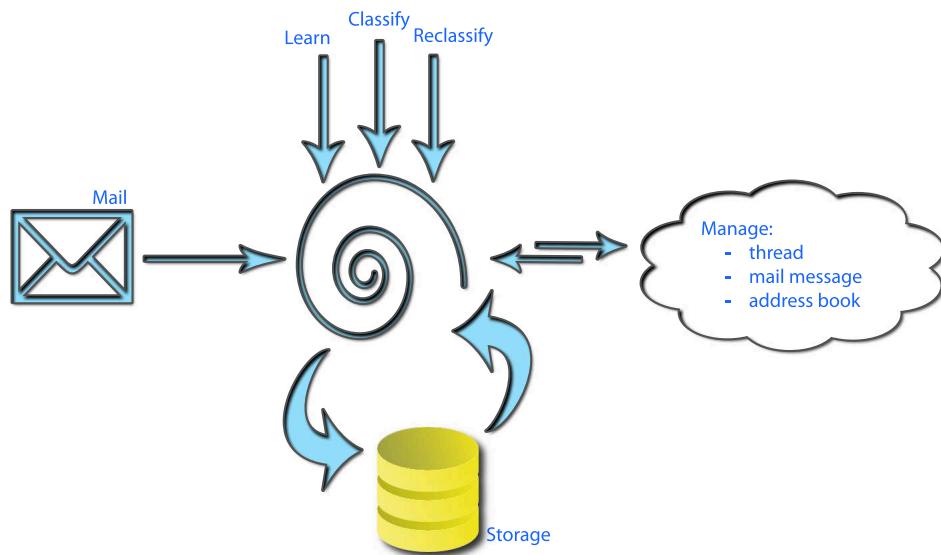


Figura 7.1. Schema a blocchi dell'infrastruttura

7.1 Progettazione del software

L'infrastruttura è stata creata in modo gerarchico. L'idea è stata quella di implementare un progetto che potesse essere scalabile e crescere con il passare del tempo, senza stravolgere la struttura creata. Il linguaggio scelto per l'implementazione è stato il C sharp. La scelta è ricaduta su questo poiché è “object-oriented” e permette di esprimere concetti e procedure di alto livello in maniera più semplificata rispetto ad un codice più spartano come può essere considerato il C. Attualmente il C sharp fa parte dello standard ECMA-334¹, quindi è portabile su molte piattaforme.

Si è organizzata la “solution” sotto un'unica radice, *Icetricky*. I progetti sono stati organizzati in cartelle gerarchiche come segue:

GreenGrass : una libreria che svolge la funzionalità di interfacciamento verso un'una base dati relazionale. Trasforma le triple delle inferenze RDF in “query” e oggetti comprensibili alla base dati. È un progetto GPL² sviluppato da Federico Di Gregorio e Pierluigi Di Nunzio.

¹ECMA: È un'associazione industriale fondata nel 1961 e dedicata alla standardizzazione nel campo dell'ICT, “Information and Communication Technology”.

²General Public License: è una licenza di software libero, scritta nel 1989 da Richard Stallman e Eben Moglen, per distribuire i programmi creati dal Progetto GNU.

Optnik : libreria anch’essa sviluppata da Federico Di Gregorio, progetto GPL.

Permette di avviare comodamente la parte di test di *Icetriky* mediante linea di comando, fornendo la possibilità di un comodo inserimento di parametri e variabili.

SharpMimeTools : progetto GPL, si occupa di effettuare il “parsing” di una mail, anche in formato MIME. È stato sviluppato da Angel Marin.

Icetriky : è una libreria che permette di effettuare le operazioni di creazione dei flussi di conversazione e di gestione dell’interfacciamento motore RDF con l’applicativo creato.

Icetriky.Core.Composer : in essa sono annidate tutte le operazioni di formattazione del body, pulizia del body dall’eventuale presenza di codice HTML e da stopwords. In essa, inoltre, è implementato il riconoscimento lingua.

Icetriky.Core.MessageFactory : si occupa della gestione dei messaggi. In essa si crea il concetto di *MailMessage* oltre che tutti gli algoritmi di sorting.

Icetriky.Core.PeopleFactory : è l’implementazione alla base del concetto di rubrica. Infatti le persone inserite in questa base dati sono organizzate in base al loro nome e hanno una rete di inferenze sugli eventuali indirizzi mail associati.

Icetriky.Core.Similarity : in essa si svolgono le funzionalità di *Levenshtein* e algoritmi per capire quanto due espressioni siano tra loro simili.

Icetriky.Engine.Bayes : fa parte del “cuore” dell’infrastruttura. In essa vi è la classe *Classifier* che implementa l’algoritmo del classificatore del Naive Bayes. Questa libreria ha la funzione anche di creare, mediante il suo apposito *Learning*, la sua base di conoscenza e la creazione di *Category*.

Icetriky.Engine.NBayes : è il secondo filtro statistico implementato. Esso, alla pari del filtro puramente bayesiano, al suo interno svolge le funzioni di apprendimento, *Learning*, creazione delle categorie, *Category* e implementa l’algoritmo del Naive Bayes associato alla catena di Markov, rif. 3.4.

Icetriky.Engine.Svm : in questa libreria vengono implementate tutte le funzioni del filtro SVM, oltre, come per i due precedenti, l’implementazione dell’apprendimento, *Learning*, la creazione delle categorie, *Category* e la classificazione *Classifier*. Per i meccanismi interni delle macchine a vettori di supporto è stata utilizzata una versione di “libsvm”. Ulteriori dettagli in A

Icetriky.Imap : è una libreria che implementa tutte le funzioni di interfacciamento verso server IMAP. L'implementazione prevede quasi totalmente lo standard RFC 3501 4rev1.

Icetriky.Owl.Configuration : questa libreria, generata da GgTool, un applicativo di GreenGrass, presenta delle classi che forniscono alla libreria GreenGrass la serializzazione delle classi dentro l'ontologia. I dati che tratta sono prettamente di configurazione.

Icetriky.Owl.Core : anch'essa è una libreria di classi che viene generata dall'applicativo GgTool. I dati tradotti fanno parte dell'ontologia associata ai messaggi, persone.

Icetriky.Web : è la libreria del Web Service, in essa vi è implementato lo schema del Singleton e la gestione dell'account per ogni utente.

Rainbow : essa svolge le funzionalità di test. Utilizza Optnik per l'interfacciamento da linea di comando.

Tests : svolge le funzioni di test.

7.2 Accesso alle risorse

Con accesso alle risorse si intende quel processo che si occupa di prelevare i messaggi di posta elettronica memorizzati in un supporto di memorizzazione che può essere diverso a seconda delle esigenze. Si è scelto di permettere l'accesso delle risorse per due casi:

- prelevamento messaggi da mbox,
- raccolta dati da server imap.

Nel primo caso si tratta di un file che contiene al suo interno un insieme di messaggi di posta elettronica. All'interno tutti i messaggi sono concatenati e memorizzati come “plain text” in un singolo file. L'inizio di ogni nuovo elemento è indicato da una linea che inizia con “From” seguito da uno spazio. In una Mbox i messaggi vengono memorizzati nel loro formato originali, conformi alla RFC 2822³, in una area direttamente accessibile all'utente. Alcuni programmi di posta elettronica, come ad esempio Evolution o Thunderbird, utilizzano questo formato per memorizzare i messaggi utilizzando un file differente per ogni cartella in cui sono organizzati. Un esempio di come un messaggio viene inserito all'interno di una Mbox è:

³RFC 2822: Internet Message Format

```
From god@heaven.af.mil Sat Jan 3 01:05:34 1996
Return-Path: <god@heaven.af.mil>
Delivered-To: djb@silverton.berkeley.edu
Date: 3 Jan 1996 01:05:34 -0000
From: God <god@heaven.af.mil>
To: djb@silverton.berkeley.edu (D. J. Bernstein)
```

How's that mail system project coming along?

La linea finale è sempre completamente vuota, anche senza spazi o tabulazioni. Bisogna tener presente che possono anche presentarsi delle linee vuote all'interno del corpo del messaggio per cui il riconoscimento della fine di un messaggio consisterà nel trovare una linea vuota seguita dalla stringa "From " nella successiva. Eventuali "From " all'interno del messaggio mail vengono quotati attraverso l'uso del carattere '>'. Il secondo caso, invece, prevede l'accesso alla casella dell'utente dalla quale prelevare i messaggi di posta presenti. Da entrambi i casi si arriva ad un lista di MailMessage, ossia la classe che identifica il singolo messaggio di posta elettronica, che verrà analizzato e manipolato dal processo di classificazione.

Il compito di lettura e parsing dei dati è affidato alla classe *Composer* che contiene al suo interno la possibilità di scegliere la sorgente delle risorse. Nel caso della mbox legge il file specificato e, in base alla considerazione precedente, riconosce il testo di un singolo messaggio e lo manda in analisi al parser. Per quest'analisi si è deciso di utilizzare una libreria già scritta chiamata "SharpMimeTool". Rilasciata sotto licenza GNU, si occupa di analizzare il contenuto di uno stream ed identificare i vari campi di un messaggio di testo secondo lo standard definito nella RFC 2045⁴. L'elaborazione produce una classe che sarà utilizzata per recuperare i campi d'interesse e andare a costruire un nuovo oggetto di tipo MailMessage, che sarà analizzato e manipolato in fase di apprendimento o classificazione.

Nel secondo caso invece i dati risiedono, normalmente, su un server al quale si può accedere attraverso l'uso del protocollo IMAP, rif. 2.4.2. In questo caso è stata scritta una libreria che si occupa della comunicazione e del trasferimento dei dati tra l'applicativo e il server in questione. La libreria è ottimizzata per fare richiesta solamente dei dati che servono in fase di classificazione. Sfruttando i comandi dell'IMAP si evita quindi di prelevare completamente il contenuto di un messaggio evitando quindi un notevole spreco di risorse. Basti pensare al caso in cui siano contenuti una lista di allegati. Ai fini della classificazione non interessa, almeno in

⁴ RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies

questa parte, conoscere il contenuto del file ma solamente il suo nome. In questo modo, ossia memorizzando esclusivamente il nome dell'allegato, si evita una eccessiva occupazione delle risorse a disposizione. Per questo caso specifico si identifica all'interno del testo del messaggio il boundary che suddivide nella varie parti il corpo del messaggio di posta. In seguito si cercherà l'intestazione in cui viene “introdotto” un allegato e qui prelevato il suo nome. Per quanto riguarda invece gli “header” si prelevano solamente quelli di interesse andando a effettuare richieste mirate. Queste azioni sono effettuate attraverso delle richieste *FETCH* al server sul messaggio che si sta analizzando e andando a specificare il campo di interesse. Il parser si occuperà quindi in questo caso di richiedere i dati che desidera e creerà l'oggetto *MailMessage* come nel caso precedente. L'oggetto *MailMessage* conterrà i campi:

- indirizzi dei destinatari e del mittente;
- dati relativi alla spedizione come l'identificativo del messaggio e le “*References*”;
- il soggetto del messaggio, la data di spedizione, eventuali dati del server *imap* da cui stato prelevato;
- il body del messaggio.

Queste informazioni sono la base per tutte le fasi successive come la classificazione, il raggruppamento in conversazioni ed eventuali ordinamenti richiesti dall'utente. La fig. 7.2 riassume la fase dell'accesso alle risorse.

7.3 Apprendimento

La fase di “learning” si basa sulla costruzione delle categorie. Quest'ultima viene creata solo quando l'utilizzatore dell'applicativo decide di creare un insieme contenitore in cui mettere i suoi esempi positivi e negativi dai quali è possibile prendere le “feature” che meglio possono rappresentare l'insieme (Cfr. [3]). Per capire meglio quanto detto, si pensi all'utilizzo di un generico programma di posta. In esso, talvolta, si gestisce l'arrivo della posta in modo gerarchico, con l'utilizzo di cartelle. La creazione della cartella avviene solo se l'utilizzatore ne ha realmente bisogno e crea un contenitore che meglio può rappresentare mediante la sua etichetta (il nome della cartella) il suo contenuto. Il discorso è totalmente analogo, ma si è cercato di astrarsi dal problema cartella in quanto poco flessibile ad alcune operazioni, sia esse di ricerca, sia essa di “tagging”⁵, rif. 7.4 .

⁵ *tagging*: operazione che permette di associare ad una categoria una etichetta

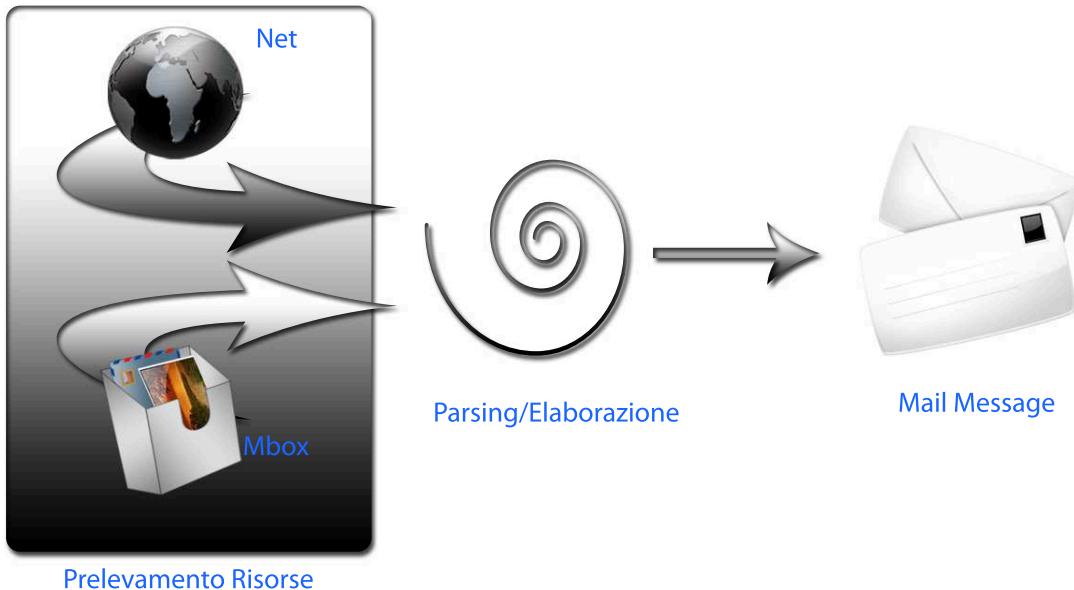


Figura 7.2. Accesso alle risorse

Si è cercato di utilizzare un formato di dati simile per i diversi stadi di apprendimento dovuti alla diversità dei filtri impiegati. La classe Learning è contenuta nei progetti *Icetricky.Core.Engine.Bayes*, *Icetricky.Core.Engine.NBayes*, *Icetricky.Core.Engine.Svm*. I metodi esposti dalla classe di Learning sono rappresentati nel diagramma delle classi, appendice E. La diversità maggiore riguarda la classe Learning contenuta in *Svm*, in quanto la fase di “learning” prevede la costruzione di esempi positivi e negativi, quello che invece non avviene nei filtri statistici in quanto si è deciso di utilizzare come esempi negativi gli esempi di tutte le altre classi. Questo può rappresentare un costo maggiore in fase di esecuzione, ma allo stesso tempo permette un maggiore insieme di caratteristiche che meglio identificano una categoria.

La classe Category è la diretta conseguenza dell’apprendimento e generata come sua derivazione. Così come è stato fatto per la classe Learning, si è cercato di rendere simili le classi Category contenute dentro i progetti dei filtri. Una quasi uguaglianza di implementazione è prevista per Category dei filtri statistici, mentre vi è una differenza decisa nell’implementazione della classe nel progetto del filtro Svm, in quanto si è deciso di inserire il modello della fase di apprendimento dentro la categoria. Il problema, che si è posto affrontando la modellazione di questa classe, era gestire l’esclusività di una concettualizzazione. Per rappresentare questo concetto si faccia riferimento alla fig. 7.3. Dato un insieme di categorie se in esso è presente una categoria esclusiva, allora la scelta di classificazione può essere solo di tipo puntuale. Nel progetto di Icetricky la scelta di avere una categoria esclusiva implica l’eventuale

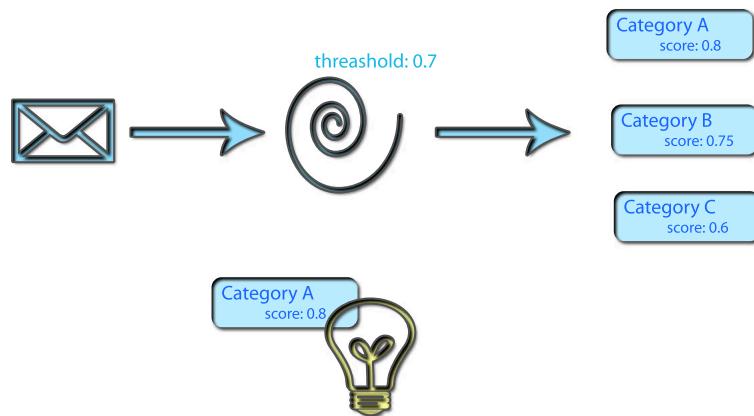


Figura 7.3. Esempio di categorie esclusive

assegnazione della singola etichetta che meglio rappresenta l'ingresso. Se una classe Category ha il “flag” Exclusive non settato, allora implica che essa non è esclusiva. Per essa varrà il ragionamento contrario a quanto detto sopra. Si faccia riferimento alla fig 7.4.

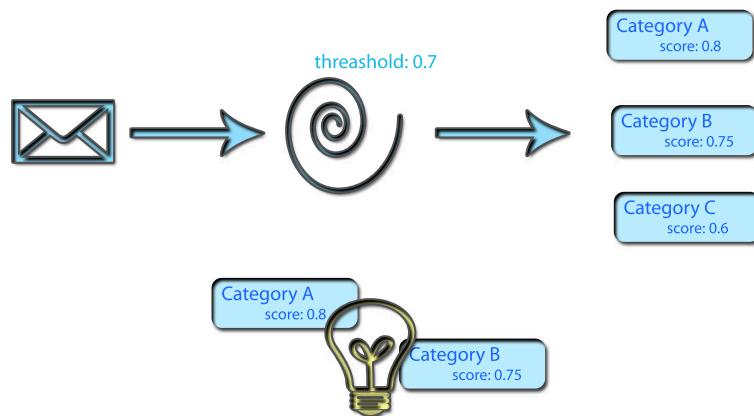


Figura 7.4. Esempio di categorie non esclusive

Passando dalla fase di “learning” si arriva alla creazione delle categorie, mediante un nome ed un insieme di mail. Questa fase porta come risultato la creazione della *Base di conoscenza*⁶. Una categoria è identificata in modo univoco dal suo nome,

⁶ *Base di conoscenza*: indicata con KB, essa rappresenta una base dati in cui si conserva la conoscenza, ossia tutta l'esperienza regressa.

pertanto non sarà possibile avere due categorie con ugual nome che appartengono alla KB dello stesso filtro.

Come accennato sopra, l’addestramento varia in base al tipo di filtro. Per il filtro bayesiano puro, il learning consiste nell’archiviazione dell’insieme delle parole da cui sono costituite le mail acquisite in fase di “learning”. Ogni categoria pertanto avrà un insieme di vocaboli, indicizzati mediante una struttura a dizionario. Gli esempi positivi per questo filtro sono le mail che costituiscono il KB locale, mentre quelli negativi sono le mail appartenenti alle restanti categorie. I casi limite di questa scelta sono dettati da una struttura intrinsecamente sbilanciata, pertanto si potrebbe incorrere in classificazioni non totalmente rappresentative del KB. Infatti, considerando l’analisi che effettua un generico classificatore di Bayes, dati tre insiemi C_1 , C_2 , C_3 con una cardinalità sbilanciata, si supponga $\#C_1 = 4$ e $\#C_2 = \#C_3 = 10$ la probabilità che le “feature” caratterizzanti per l’insieme C_1 siano contenute in $C_2 \cup C_3$. Questo potrebbe abbassare il peso di classificazione ottenuto in base al calcolo bayesiano, nonostante si applichi una struttura pesata. L’assunzione maggiore fatta è che l’utilizzatore nella creazione delle categorie e gestione di esse, tenda sempre a mantenere un generale bilanciamento. Nella fig. 7.5 si ha una dimostrazione dei possibili problemi di non convergenza, gli elementi circolari rappresentano le eventuali “feature” di una KB.

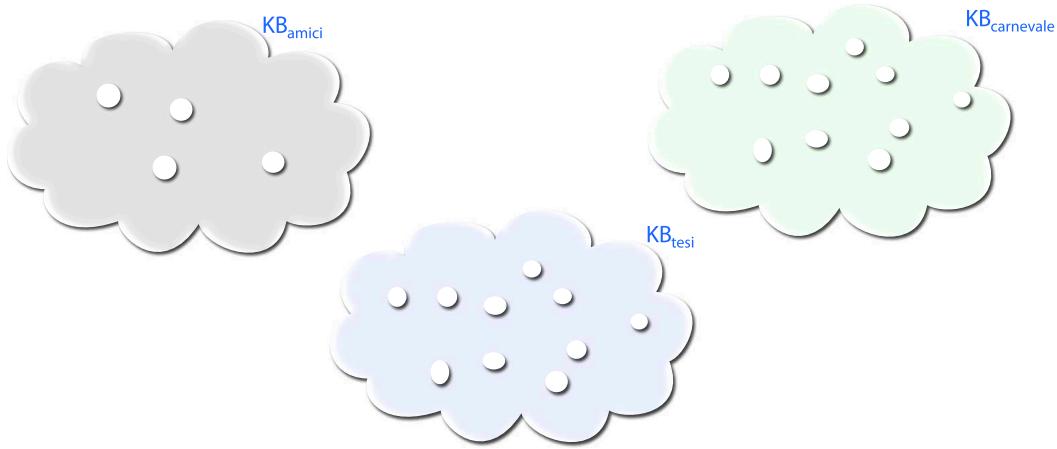


Figura 7.5. Esempio di dominio di conoscenza sbilanciato

Occorre pertanto esprimere il concetto di bilanciamento. Esso, in generale, indica una somiglianza, almeno per ordine di grandezza, della cardinalità di alcuni insiemi appartenenti ad un generico dominio. Pertanto la parola bilanciamento, nella nostra infrastruttura, può essere riferita al numero di occorrenze di parole o mail che caratterizzano insiemi in un KB.

Il filtro NBayes rappresenta la versione migliorata del Naive Bayes, esso pertanto

eredita la struttura del filtro padre, ma in più presenta miglioramenti dovuti all'utilizzo della catena di Markov. Lo schema del learning è pertanto identica a quella vista sopra, a meno della costruzione degli ngram. Pertanto l'insieme che rappresenta una categoria al suo interno conterrà, oltre alle parole, anche gli ngram generati. Anche la struttura sbilanciata sfavorisce il funzionamento del filtro.

Differenti sono i criteri per il filtro Svm. Si è implementata una struttura bilanciata, aventi lo stesso numero (numero di mail utilizzate per rappresentare il dominio di conoscenza relativo alla data categoria) di esempi positivi e negativi. Assumendo che due mail sono differenti nel numero di parole contenuto, ossia nella cardinalità delle "feature" che contengono, il problema si dimostra pertanto sbilanciato. L'assunzione utilizzata prevede insiemi costituiti da un numero uguale di mail, che si possono compensare con il numero di cardinalità di "feature" contenute. L'altra differenza sostanziale della fase di learning, prevista per questo filtro, è rappresentata dall'etichetta esclusiva. In base a questa, varia anche la fase di apprendimento. Se una categoria viene definita esclusiva per questo filtro, allora per l'addestramento saranno utilizzate tutte le mail che rappresentano esempi positivi e negativi per la stessa categoria e tutti gli esempi positivi delle altre categorie, mentre se la categoria è definita non esclusiva, allora vengono prese in considerazione solo le mail che rappresentano esempi positivi e negativi della categoria stessa. Occorre però ricordare che per avere un comportamento migliore, la categoria dovrebbe contenere esempi negativi e positivi in numero confrontabile in modo che non sia sbilanciata la fase di apprendimento per la ricerca dell'iperpiano ottimo e quindi la suddivisione dello spazio delle "feature". Si verificheranno nei test le prestazioni in caso di apprendimento sbilanciato, in modo da capire se effettivamente provoca un notevole calo dell'affidabilità di classificazione.

La fase di apprendimento rappresenta il cuore dell'intelligenza di tutti i motori di classificazione. Pertanto, la decisione di associare ad una classe o categoria un nuovo esempio positivo o negativo comporta un incremento in prestazione dei filtri, in quanto si tende ad affinare e a migliorare il dominio di conoscenza relativo al particolare settore etichettato dal nome della categoria stessa. I filtri reagiscono in maniera diversa all'inserimento di una nuova mail nel "training set". I filtri statistici, lavorando sul semplice confronto di parole o ngram, appena una nuova mail viene appresa, modificano subito il loro modello di base di conoscenza, in quanto, appunto, caratterizzato dall'insieme di "feature". Pertanto la classificazione successiva subirà le modifiche del learning precedente, questa tecnica prende anche il nome di *forward*, 7.5. Diverso, invece, è il discorso per il filtro basato sulle support vector machine, per il quale un'aggiunta di una mail nel "training set" non modifica direttamente il modello della base di conoscenza, ma deve essere esplicitamente richiesto la ricreazione del modello stesso.

Mentre per i filtri statistici il modello è rappresentato dall'insieme di termini che meglio rappresentano la classe (sia esempi positivi che negativi), per il filtro Svm il

modello rappresenta delle informazioni aggiuntive calcolate “ad-hoc” utilizzate per rappresentare la categoria nel dominio delle possibili soluzioni, sia esso bidimensionale o multidimensionale. Queste informazioni sono onerose in termini di occupazione di memoria e anche in termini di calcolo computazionale, pertanto un’ulteriore ottimizzazione è stata quella di assegnare un modello a tutte le categorie utilizzate per classificazioni binarie, mentre un unico modello se si svolge un calcolo di tipo multi-classe, si veda par. 6.2 e 7.4.

Una generica mail contiene numerose informazioni inutili, anche chiamate “stop-words”, rif. 6.5.2 e rif. 6.5.4, pertanto si è scelto di eliminarle, il compito è demandato alla classe *Format*. La cancellazione però comporta il riconoscimento, in quanto per ogni modello linguistico vi sono un differente vocabolario di termini poco determinanti in una comunicazione, rif. 6.5.2 e viene demandato alla classe *Identifier*. La fase di apprendimento pertanto richiede uno stadio aggiuntivo che prevede questa funzione, affinche al filtro arrivi solo del testo in chiaro, anche chiamato “text plain” o “ngram text plain”. Vi è anche da aggiungere la pulizia del corpo del messaggio dalla presenza eventuale di html, rif. 6.5.1 ed è svolta dalla classe *HtmlGrub*.

7.4 Classificazione

La fase di classificazione prende in ingresso un messaggio e, attraverso la sua analisi, produce in uscita delle assunzioni di appartenenza dell’elemento ad una o più delle categorie create in precedenza. Può essere vista come il momento in cui l’utente, davanti al suo programma di posta elettronica, legge i messaggi e, in base al contenuto, li inserisce nelle cartelle già create per tenere organizzata la sua casella di posta. È chiaro che questa fase sarà successiva ad almeno una fase di apprendimento, descritta in 7.3, altrimenti non si avrebbero modelli a disposizione per effettuare l’operazione. Il processo di classificazione presenta alcune differenze tra i filtri di tipo bayesiano e le support vector machine. Innanzitutto il tipo di classificazione dipende dal tipo di apprendimento eseguito in precedenza. Come già detto in 6.2 possono presentarsi tre casi: binario, multi-classe o multipla. Nel caso delle “support vector machine” è particolarmente utile conoscere questo particolare in quanto permette di capire dove recuperare il modello di classificazione. Invece nei filtri bayesiani il modello di classificazione è contenuto all’interno della base di conoscenza associata alla generica categoria. Nel caso binario si dividerà lo spazio delle possibili soluzioni in due parti, la parte che rappresenterà l’“Ham”⁷ e la parte che rappresenta tutti gli esempi negativi di caratterizzazione della classe. Nel multi-classe, invece, si effettua un’analisi di probabilità in parallelo su tutte le classi. Si prende come riferimento

⁷ *Ham*: termine usato per la prima volta da Graham per indicare i messaggi di posta che non rappresentavamo spam

la classe interessata e si calcola la probabilità attesa, considerando tutto l'insieme delle possibili occorrenze. Ritornando alle SVM, nel caso binario, ogni categoria avrà un proprio modello e quindi, per ogni categoria, si utilizzerà quello salvato per capire se il messaggio appartiene o meno alla categoria. Il singolo filtro, quindi, dà una stima se il messaggio in entrata appartiene o meno alla categoria da cui è stato addestrato. Il singolo funzionamento è quindi l'analogo del filtro “Spam and Ham”, già citato più volte. L'operazione viene ripetuta con tutte le categorie e infine sarà presente un meccanismo che avrà il compito di valutare i risultati ottenuti. Diverso invece il discorso per quanto riguarda il multi-classe. In questo caso abbiamo un unico modello a conoscenza di tutte le categorie presenti. Le categorie sono quindi identificate attraverso un indice univoco in formato numerico crescente. Nel caso multi-classe, il modello viene recuperato dalla categoria con indice 1, le altre infatti, per evitare ridondanza, non contengono al loro interno nessun valore in tale campo. Il processo di classificazione restituirà, in questo tipo di classificazione, l'indice della categoria a cui appartiene il messaggio e un valore di probabilità sulla stima.

La classe *Classifier*, contenuta nei progetti *Icetricky.Core.Engine.Bayes*, *Icetricky.Core.Engine.NBayes* e *Icetricky.Core.Engine.Svm*, svolge le azioni descritte in precedenza. Soprattutto in base al tipo di classificazione che viene demandato dal sistema si comporterà in modo differente. Il processo di predizione restituisce un *Dictionary<string, double>* che al suo interno contiene il punteggio dato a ogni categoria da parte filtri. In particolar modo il primo termine sarà il nome della categoria, che rappresenta il suo identificativo univoco visto che logicamente non possono esserci più classi di appartenenza con lo stesso nome. Il secondo termine, invece, contiene la probabilità di appartenenza del messaggio alla categoria. Questa struttura dati è molto importante perché permette, indipendentemente dal tipo di classificazione o da come sono costruite le categorie, di applicare tutti i ragionamenti necessari per applicare i “tag” al messaggio.

La funzione *Reasoner*, presente nelle classi *Classifier*, si occupa di elaborare i risultati precedentemente descritti. In questa fase assume particolare importanza un valore soglia impostato del classificatore. Tale valore identifica un limite minimo per cui decidere se applicare il “tag” della classificazione o meno. La soglia viene confrontata con la probabilità risultante dal processo di classificazione per la singola categoria. Di default si accettano appartenenze con una predizione di almeno il 60%. Questo può anche essere forzato e accettare valori superiori al 50%. Nella fase di applicazione dei tag si prospettano due scenari:

- tra i risultati validi sono presenti categorie esclusive;
- tutte le categorie non sono esclusive.

Come primo passo, in questa fase decisionale si cerca all'interno dei risultati la categoria migliore che ha ottenuto un punteggio maggiore. A questo punto si analizza

la categoria e si vede se essa è esclusiva o meno. Se lo è viene applicato solamente il “tag” relativo e gli altri risultati vengono scartati, sempre se il punteggio ottenuto è superiore alla soglia. Differente è il caso in cui la categoria, che ha ottenuto il miglior punteggio, permetta l’applicazione anche di categorie non esclusive. In questo caso si rilevano tutte quelle che hanno prodotto un risultato superiore alla soglia. A questo punto si analizzano tali risultati e vengono applicati i “tag” delle categorie interessate che non sono esclusive. Nel caso in cui nessun risultato sia soddisfacente non viene effettuata l’operazione di “tagging” e il messaggio di posta non risulterà appartenere a nessun gruppo di classificazione. Nelle figure 7.6 e 7.7 è rappresentato quanto appena descritto. Occorre ricordare che se nella fase di

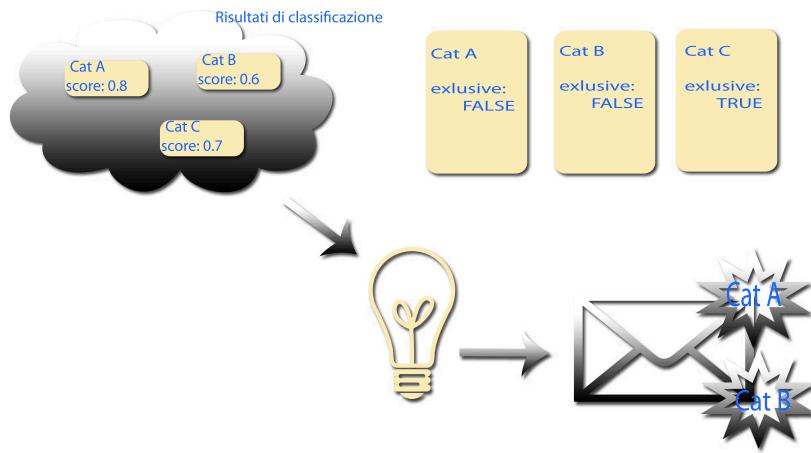


Figura 7.6. Esempio di categorizzazione multipla

apprendimento non sono stati inseriti esempi negativi nella categoria, ossia messaggi che non appartengono ad essa, il processo di classificazione, nel caso binario e con categorie non esclusive, dirà che il messaggio appartiene a tutte le categorie. Ciò è dovuto al fatto che i singoli filtri hanno solo conoscenza di cosa appartiene alla categoria e non di cosa non è di loro competenza quindi non si crea un modello che possa segnare una distinzione tra una zona d’appartenenza e un’altra.

7.5 Strategie di controllo

Come accennato nel par. 7.3, rendere dinamica la parte di apprendimento è alla base del concetto di “Machine Learning”. Infatti, variando l’esperienza regressa con nuove caratteristiche o descrizioni di come si è evoluto il sistema, si fornisce ai filtri nuovo materiale su cui effettuare i ragionamenti. Cambiare però la KB, implica un cambiamento strategico da parte dei filtri, come enunciato sopra. I filtri della

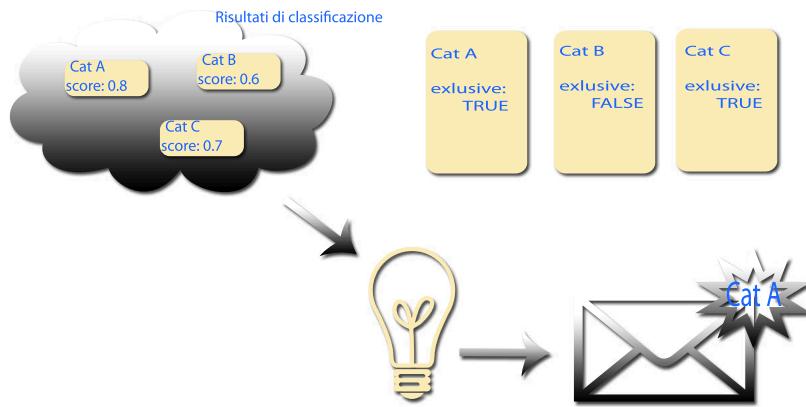


Figura 7.7. Esempio di categorizzazione singola

famiglia bayesiana apprendono subito questa modifica, in quanto il loro modello è annidato dentro l’insieme delle “feature” conosciute. Invece per le “support vector machine” il modello deve essere esplicitamente ricalcolato.

Si apre il problema del dominio di applicazione della nuova conoscenza acquisita. Si supponga di avere una mail che entra nell’infrastruttura e si decide di inserirla nella categoria Amici. Questa, al suo interno, aveva una base di conoscenza sulla quale precedentemente si erano effettuati dei ragionamenti e prese delle decisioni di classificazione per n mail precedenti. L’inserimento di nuove “feature” in questa base dati provoca una modifica delle precedenti assunzioni insiemistiche, pertanto le precedenti classificazioni possono ritenersi non più valide per i cambiamenti occorsi. A tal punto possiamo distinguere tra due possibili strategie, che prendono il nome di strategie di controllo (Cfr. [24]):

1. forward, supponendo di lavorare su una scala temporale, $0 \dots t$, scelto un punto $t_f : 0 < t_f < t$, la strategia di controllo modificata verrà applicata a tutti gli eventi che si succedono nell’intervallo temporale $t_f \dots t$. Possiamo definire che questa è una strategia di controllo in avanti;
2. backward, prendendo come riferimento la scala temporale precedente, $0 \dots t$, scelto un punto $t_b : 0 < t_b < t$, la strategia di controllo modificata verrà applicata a tutti gli eventi dello scopo, ossia a tutti quelli che si sono verificati tra $0 \dots t$. Questa strategia è anche chiamata all’indietro in quanto percorre tutti gli eventi temporali partendo dalla radice.

L’implementazione seguita prevede entrambi i modelli di strategia di controllo per tutti i filtri progettati. Se ad una seconda fase di “learning” viene seguita la

costruzione del model, allora si applica di “default” la strategia di controllo del “forward”. Se, invece, si decide di effettuare una riclassificazione, allora si applica una strategia di controllo di tipo “backward”. Talvolta, però, una strategia di riclassificazione può modificare il flusso temporale degli eventi e di conseguenza la sua riclassificazione. Per questo motivo la strategia di “default” applicata è la prima, mentre l'utilizzatore può decidere di utilizzare la seconda. In aggiunta, il fine cercato era quello di creare una infrastruttura che potesse interagire con un utilizzatore. Il tempo di risposta, deve quindi essere contenuto e non deve superare come ordine di grandezza la decina di secondi, nel caso peggiore. Una riclassificazione completa implica una ricostruzione del modello, nonostante sia poco dispendiosa per i metodi statistici, richiede, invece, molto tempo per il filtro del “support vector machine”. A questa fase deve poi seguire la classificazione e anch'essa rappresenta un'occupazione temporale che cresce linearmente con il numero di messaggi della base di conoscenza.

7.6 Flussi di discussione

Prendendo come riferimento esempi di ricezione ed invio di posta, ci si trova a lavorare con messaggi che spesso rappresentano una risposta ad una richiesta. Si è cercato di organizzare questi flussi di discussione in gruppi, chiamati “Thread”. Un flusso può essere creato dall'insieme di messaggi di invio e risposta, ma anche dal nome dell'oggetto uguale o simile. Si è anche considerato l'utilizzo di messaggi in discussione che presentano lo stesso nome degli allegati. Questo infatti, avviene specialmente in ambiti lavorativi, dove si utilizzano frequentemente messaggi di posta elettronica per trasferire documenti di lavoro. Ad un messaggio di questo tipo si risponde, spesso, creando una nuova mail, ma allegando il file con il vecchio nome, ma aggiornato nel contenuto. Per implementare il concetto di flusso di discussione si è ricorso alla creazione delle classi *Person*, contenute in *People*, *MailMessage* e *Thread*.

La prima problematica affrontata riguarda la costruzione della rubrica personale dell'utente. Quando si incontra un nuovo messaggio all'interno della casella di posta si vuole, oltre a classificare, riuscire a costruire automaticamente la rubrica indirizzi. Le operazioni per svolgere tale compito sono implementate all'interno del namespace *Icetricky.Core.PeopleFactory*. A ogni nuovo messaggio viene analizzato il mittente e si estraggono, ove possibile, il “Visualized Name” e l'indirizzo. In base a questi dati si ricerca all'interno della rubrica se l'indirizzo in questione è già presente. In caso negativo si vuole inserire il nuovo contatto. Prima di tutto si verifica se quell'indirizzo appartiene a qualche contatto già conosciuto. Si cerca, quindi, attraverso l'uso di Levenshtein (sezione 6.6.1), implementato nel namespace *Icetricky.Core.Similarity*, di capire se, confrontando sia il “Visualized Name” con quelli già presenti, sia la

prima parte dell'indirizzo (ossia precedente il carattere '@') se esiste qualche altro elemento che presenta una notevole somiglianza con il termine in analisi. L'elevata somiglianza è stata parametrizzata attraverso un valore soglia, al di sopra del quale si ritiene il confronto aver prodotto un risultato positivo. Nel caso in cui non si presenti nessuna somiglianza soddisfacente, la rubrica si arricchisce di un nuovo elemento. L'“address book” e i suoi elementi vengono realizzati attraverso le tre classi *People*, *Person* e *Email*. La prima è il contenitore che rappresenta l'intera rubrica ed è composta da vari elementi *Person*, che rappresentano il singolo contatto. Questi ultimi, al loro interno contengono, oltre al nome identificativo, la lista dei vari indirizzi, ossia gli oggetti *Email*.

L'astrazione dei messaggi di posta viene effettuata attraverso la classe *MailMessage* presente nel namespace *Icetricky.Core.MessageFactory*. Oltre alle informazioni tipiche del messaggio di testo sono contenute due importanti strutture dati, utili soprattutto per la memorizzazione dei risultati di classificazione. La prima, la *MethodScoreList*, contiene la lista dei risultati del processo di classificazione suddivisi per filtro applicato. Ciò permette scalabilità per l'eventuale inserimento di nuovi filtri di apprendimento. La seconda struttura dati, la *TagList*, contiene tutti i tag che il “reasoner” ha ritenuto validi per essere applicati al messaggio. Il “MailMessage” viene successivamente mappato su una lista di “Thread”, infatti anche il singolo messaggio viene considerato facente parte di un'intera discussione.

7.7 Base Dati

I dati prodotti dall'applicativo necessitano di un supporto di memorizzazione e si è scelto di utilizzare due basi dati indipendenti utilizzando *PostgreSQL*⁸. La prima, di tipo ontologico, è utilizzata per l'archiviazione dei risultati di apprendimento e classificazione. La scelta è dettata dal fatto che sono previsti in futuro ulteriori sviluppi, come ad esempio integrare il processo di classificazione attraverso ragionamenti di tipo ontologico. Per la comunicazione con questo tipo di base dati è stata utilizzato l'applicativo *Greengrass*. Questa permette operazioni di memorizzazione, manipolazione e prelevamento di dati RDF, trasformandole in “query” comprensibili alla base dati. Lo schema delle classi create si trova in F.

Altri dati, che non sono previsti poter essere utilizzati in ragionamenti ontologici, sono memorizzati in un database relazionale. Lo schema delle tabelle generate si trova in D.2. Qui vengono memorizzati i dati necessari per la configurazione del sistema. Si possono trovare:

- dati relativi agli utenti utilizzatori dei servizi,

⁸ *PostgreSQL*: completo database relazionale ad oggetti con licenza libera stile BSD.

- dati relativi alle lingue conosciute dal sistema,
- informazioni di log relative.

Riguardo alle informazioni degli utenti si tiene traccia dei loro dati di registrazione al servizio. Questi dati sono utili soprattutto per essere in grado di recuperare, all'interno del database ontologico, le informazioni di proprietà dell'utente. Oltre a questo vengono memorizzati anche i server smtp che l'utente utilizza per spedire i propri messaggi.

Per quanto riguarda il riconoscimento della lingua e dell'indicizzazione per il formato SVM, occorrono delle tabelle da cui prelevare le informazioni. In questo caso memorizziamo le caratteristiche tipiche di ogni lingua relativamente agli ngrams e le "stop words". Oltre a ciò, sempre nel database *Icetricky-Configuration*, è presente la tabella *bow* in cui è contenuta l'indicizzazione delle parole incontrate. Infine si tiene traccia, nelle tabelle *uilog* e *imaplog*, delle operazioni svolte dall'utente attraverso l'interfaccia grafica realizzata e della comunicazione con i server IMAP.

Capitolo 8

Risultati

Le prove sul funzionamento dei filtri sono state eseguite su una reale “mail box”, pertanto si è messa duramente alla prova l’infrastruttura in quanto non si sono utilizzate mail costruite “ad hoc”. L’insieme è stato suddiviso in cartelle, nelle quali si è cercato di unire le mail per contenuti. Inoltre, dalle stesse cartelle, si è preso in modo pseudo-casuale poche mail: queste hanno rappresentato l’insieme di addestramento dei filtri, pertanto, si sono create nove differenti “mail box”.

La categorizzazione dei messaggi è stato un processo accurato, ma visto la diversità dei contenuti trattati in esse, alcune mail rappresentano inopportunamente gli insiemi nelle quali si trovano. Per tal motivo, alcuni risultati possono essere condizionati da questo. Vi è inoltre da aggiungere che la lingua, fondamentale per la costruzione del vocabolario, varia nella stessa categoria. Questo, anche se di difficile riscontro pratico nei comuni utilizzatori, mette a dura prova il funzionamento dell’architettura. Per ogni categoria partendo da un elevato numero di mail sono stati estratti, in modo casuale, i messaggi che avrebbero dovuto rappresentarla, senza aver seguito alcun flusso temporale o logico. Generalmente lo scenario applicativo di questa architettura, prevederà una fase di apprendimento, nella quale l’utilizzatore dovrà addestrare i filtri con un numero limitato di mail per categoria e l’addestramento seguirà un flusso temporale ben determinato. Quanto fatto in queste simulazioni rappresenta il caso peggiore di utilizzo. Si è ricorsi all’applicativo “Optinik” (rif. 7) per usufruire di una comoda interfaccia testuale, con la quale interagire e comandare eventualmente la fase di addestramento e di classificazione.

8.1 Scenario dei test

Nelle simulazioni eseguite si è cercato di illustrare le differenze di prestazioni dei tre filtri utilizzati. In prima analisi si sono effettuate delle prove su due singole categorie di addestramento e si è calcolata la veridicità effettuando la classificazione

delle mail dei due insiemi, dai quali è stata creata la base di conoscenza. Si sono effettuate delle prove ripetute con un incremento del numero delle mail di apprendimento per categoria. Successivamente si sono effettuate delle simulazioni sugli stessi insiemi utilizzando il riconoscimento lingua e di conseguenza la cancellazione delle “stopword” e l’eliminazione della firma. L’utilizzo di solo due categorie rende il problema della classificazione o del riconoscimento del contesto delle mail un problema comunque differente dalla ricerca dello “spam”, in quanto in un contesto di filtri binari l’uscita possibile può ricadere nell’insieme $S_{binary} = \{A, \bar{A}, B, \bar{B}\}$, rappresentato dalla fig. 8.1, mentre nel contesto multi-classe i possibili insiemi sono $S_{multiclass} = \{A, B, U\}$, U è l’insieme di soluzioni non note, come in fig. 8.2.

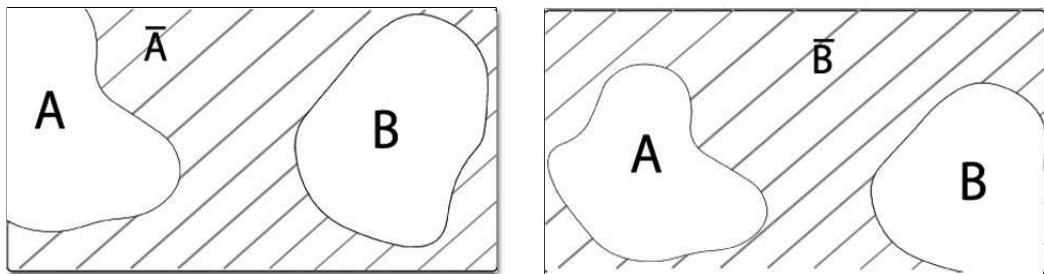


Figura 8.1. Classificazione binaria su due insiemi

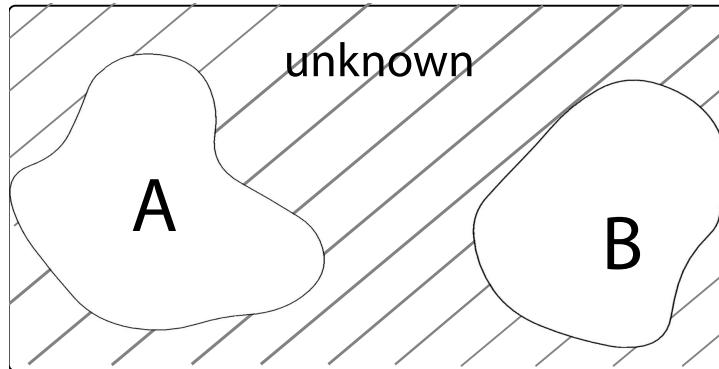


Figura 8.2. Classificazione multi-classe su due insiemi

Si è aumentato il numero di messaggi per categoria di apprendimento e di conseguenza si è testata la corretta classificazione delle mail che rappresentavano lo stesso dominio, come effettuato nella simulazione con solo due categorie. Si è poi aumentato il numero di categorie di apprendimento, lasciando invariata la quantità di messaggi di addestramento. Successivamente si sono effettuati delle simulazioni

adattative, ossia si è cercato di simulare la scelta dell’utente di “spostare” una mail classificata non correttamente nell’etichetta che realmente la rappresenta. Infine, si è cercato di fare degli esempi di addestramento bilanciato, soprattutto per il filtro di “Vapnik”.

Nella tab. 8.1 sono riportati i nomi delle cartelle utilizzate e l’indicazione di quante mail sono contenute nella cartella¹ stessa. Inoltre, data una categoria, è rappresentato il possibile tipo di addestramento.

Category	Total Mail	Learning							
		10	20	30	40	50	60	70	80
debian	263	✓	✓	✓	✓	✓	✓	✓	✓
dmt	99	✓	✓	✓	✓	✓	✓	✓	✓
eva	150	✓	✓	✓	✓	✓	✓	✓	✓
games	42	✓	✓	✓					
psycopg	229	✓	✓	✓	✓	✓	✓	✓	✓
python	156	✓	✓	✓	✓	✓	✓	✓	✓
softwarelibero	151	✓	✓	✓	✓	✓	✓	✓	✓
webbit	481	✓	✓	✓	✓	✓	✓	✓	✓
X-misc	113	✓	✓	✓	✓	✓	✓	✓	✓

Tabella 8.1. Distribuzione delle mail tra le classi

Nella tab. 8.2 si sono riportati il numero di occorrenze all’interno degli insiemi di addestramento.

Occurrency	10	20	30	40	50	60	70	80
Total	26885	73373	88309	86778	116820	148438	161305	160155
Average	2987	8152	9812	10847	14602	18554	20163	22879

Tabella 8.2. Distribuzione delle parole negli insiemi di addestramento

¹cartella: in questo contesto indica l’insieme di documenti o mail che rappresentano lo stesso concetto, ossia categoria.

8.2 Risultati acquisiti

I risultati, riportati in seguito, sono stati suddivisi per tipo di filtro utilizzato nella classificazione. Sono stati riportati, inizialmente, gli esempi di classificazione su due categorie e successivamente si è esteso l’ambito di simulazione su più categorie. Sono state anche illustrate, mediante grafici, le differenze di prestazione dovute alla pulizia del corpo del messaggio, rimuovendo la presenza eventuale di firme o di “stop words”. Infine si è riportata la differenza di prestazione dei tre filtri implementati, in termini di efficacia di classificazione e in termini di tempi di risposta. Le simulazioni sono state effettuate su una macchina, architettura “laptop”, con processore a 1.7GHz, con 512MB di memoria RAM, 256KB di cache, bus a 133MHz, HD a 3200 rpm con il sistema operativo Debian Lenny, utilizzando il “framework” Mono 1.2.6.

8.3 Confronto tra le simulazioni

La problematica maggiore, incontrata nell’implementazione di filtri nello scenario di classificazione di messaggi di posta elettronica e nell’esposizione dei risultati in tempo reale all’utilizzatore, è stata l’esigenza di contenere i tempi di calcolo nell’ordine dei [ms]. Nonostante le simulazioni siano state condotte su una macchina laptop, che sicuramente non rappresenta una soluzione server con la quale implementare questa architettura, i risultati ottenuti sono, comunque, contenuti nell’intervallo prefissato. Nella tab. 8.3 sono riportati i tempi medi di classificazione, sull’insieme di cartelle indicate sopra, con circa mille e cinquecento messaggi.

Da essa si possono notare le differenze di prestazioni dei tre filtri implementati. Il filtro del Naive Bayes ha dei tempi di calcolo nell’ordine del millisecondo, questo poichè i calcoli che svolge sono di costruzione del proprio vocabolario (par. 6.4.1) ed esegue un calcolo statistico sulle occorrenze delle parole contenute nell’insieme da classificare. Per quanto riguarda il Naive Bayes, con l’implementazione delle catene di Markov, riscontriamo un costo computazionale che è di un ordine di grandezza maggiore. Ciò è sostanzialmente dovuto alla creazione degli “ngram” e alla base di conoscenza che cresce di un fattore proporzionale alla grandezza dell’“ngram” stesso. La classificazione di un documento, pertanto, richiederà la suddivisione dello stesso in tanti “ngram” e la successiva analisi statistica del numero di occorrenze comuni. Il filtro di “Vapnik”, nonostante l’individuazione di una funzione kernel che mappi l’insieme delle feature in zone linearmente separabili, ha un costo computazionale di due ordini di grandezza maggiore rispetto al bayesiano puro ed uno rispetto all’implementazione chiamata NBayes, ma al limite di quanto prefissato.

L’implementazione del metodo multi-classe (rif. 6.2) presenta un risparmio di tempo di elaborazione del 5% per i filtri Bayes ed NBayes, mentre quasi il 50% per le support vector machine. Per quest’ultimo, il risultato ottenuto è dovuto al fatto

<i>Filtro</i>	<i>Pulizia</i>	<i>Binary</i> [ms]	<i>Multiclass</i> [ms]
Bayes	nessuna	1,3	1,09
	firma	1,29	1,08
	stopwords	1,27	1,07
	entrambe	1,26	1,05
NBayes	nessuna	28,15	27,31
	firma	27,79	26,73
	stopwords	22,07	21,75
	entrambe	22,24	21,93
SVM	nessuna	892,81	428,5
	firma	853,76	427,63
	stopwords	762,69	382,19
	entrambe	750,5	384,92

Tabella 8.3. Tempi di classificazione su una KB di due insiemi, Debian e Webbit

che nel caso binario si utilizzano n classificatori binari, in quanto esso raggruppa lo spazio delle possibili soluzioni in due insiemi, e ognuno di questi impiegherà un intervallo di tempo a svolgere l’operazione. Quindi il tempo totale sarà pari alla somma di ogni filtro. Diverso il caso multi-classe, in cui si effettua una sola classificazione, in quanto il filtro esprime un giudizio sul singolo insieme di dati. In questo caso i tempi, come già detto, sono circa la metà rispetto al caso binario utilizzando due categorie di addestramento. Sembra essere ragionevole che la differenza tra i tempi dei due casi sia proporzionale al numero di categorie di classificazione. Come avviene per SVM, anche per i bayesiani il costo computazionale è direttamente proporzionale al numero di categorie presenti. Ma il rapporto di proporzionalità non è in funzione delle n categorie, ma è dipendente, in senso lato, dal numero di parole presenti nell’insieme.

Nella tabella si è fatto, inoltre, riferimento al costo di elaborazione dovuto alla cancellazione dal corpo del messaggio di firma e stopwords. Contrariamente a quanto ci si può aspettare, la pulizia produce una riduzione nei tempi di classificazione del 20% nel caso combinato di cancellazione della firma e delle stopwords, poiché il numero di occorrenze diminuisce, riducendo così il numero di elementi del vocabolario.

Il contesto di simulazione, come accennato sopra, ha riguardato un numero di messaggi di apprendimento con alto rumore interno, questo dovuto alla difficoltà di creare degli insiemi che non presentassero delle intersezioni concettuali o a cause dell’uso di più lingue in una categoria. Per una macchina, quest’ultima considerazione, crea un diverso contesto all’interno della categoria stessa e quindi suddivide

lo spazio in n parti. Si è cercato, attraverso l'utilizzo di tutti i messaggi a disposizione, di verificare la correttezza di classificazione dei tre filtri. Nello scenario di test si sono quindi create nove categorie di apprendimento. A causa del numero differente di messaggi nelle cartelle, si sono potuti utilizzare, ai fini dell'apprendimento, al massimo trenta messaggi per ogni categoria. A priori si sa che la limitata grandezza dei domini di conoscenza porterà ad ottenere dei bassi risultati nel caso del filtro SVM. Ciò nonostante, si dovrebbe verificare un incremento della correttezza di classificazione dovuto all'aumento delle dimensioni delle cartelle utilizzate per l'apprendimento. In generale si dovrebbero ottenere dei valori anche molto diversi tra una categoria e l'altra a causa della diversa importanza dei messaggi scelti per creare gli insiemi di addestramento. Per la creazione di tali "training set", infatti, si sono presi casualmente n messaggi dalle cartelle principali, in un numero pari a quanto si voleva grande l'insieme di addestramento. I messaggi da classificare saranno quindi $m - n$ dove m indica la grandezza della cartella. Classificare dei messaggi che sono stati utilizzati per l'addestramento avrebbe portato ad ottenere dei risultati sicuramente corrette e, quindi, si sarebbe introdotto un errore nei risultati finali. In tabella 8.4 sono riportati i risultati conseguiti.

I risultati ottenuti mostrano quanto si è assunto in precedenza. Occorre porre subito l'attenzione sulla composizione delle categorie. In questo test si è voluto verificare il funzionamento dei filtri su tutti i messaggi a disposizione. Come si può intuire dal nome delle varie cartelle, si ha un insieme di email che sono fortemente correlate tra loro. Infatti tutte le categorie riguardano, in linea di massima, argomenti informatici e diventa molto difficile riuscire a rappresentare in modo univoco i singoli insiemi, soprattutto visto che i "training set" sono stati creati pescando casualmente all'interno delle varie cartelle. Nei risultati si possono riscontrare tali considerazioni. In prima analisi si osservino i risultati prodotti con filtri di tipo binario con un training pari a dieci messaggi per ogni cartella di classificazione. Le categorie *debian*, *software-libero* e *X-misc*, al loro interno, contengono contenuti molto simili tra loro e il basso dominio di conoscenza non porta ad avere una netta distinzione tra una categoria e l'altra. Questo si ripercuote soprattutto sui risultati dei filtri puramente bayesiani e sugli SVM. Si nota, infatti, un elevato grado di errore di classificazione in questi casi. In SVM, addirittura, non viene riconosciuto correttamente nessun messaggio della cartella *X-misc*. Con il filtro NBayes, invece, si hanno comportamenti nettamente migliori e, nonostante l'esiguo numero di elementi utilizzati per addestrare il filtro, si ottengono dei risultati già accettabili, intorno al 60% di correttezza. Il problema descritto in precedenza, in questo caso, incide molto meno anche se comunque si possono notare classificazioni peggiori in questi insiemi. Infatti, mentre si hanno dei risultati già buoni nella maggioranza dei casi, nelle categorie fortemente correlate si ottiene un livello di correttezza comunque inferiore. Mediamente si può concludere che il filtro NBayes, nella sua forma binaria, assume già con poca esperienza un comportamento nettamente migliore rispetto agli altri filtri presi in considerazione.

<i>Filtro</i>	<i>Cartella</i>	Binario		Multi-classe	
		10	30	10	30
Bayes	debian	11,6	37,15	6,32	6,87
	dtm	66,29	100	52,81	42,28
	eva	7,85	26,17	78,57	55
	games	8,3	18,75	81,25	91,67
	psycopg	30,59	34,17	10,96	11,55
	python	21,23	84,92	15,07	16,17
	swlibero	7,09	61,16	71,63	60,33
	webbit	96,68	98,94	29,94	66,97
	X-misc	3,61	11,82	11,65	28,48
NBayes	debian	60,08	87,98	21,74	54,51
	dtm	71,91	100	71,91	94,2
	eva	62,14	93,33	95	97,17
	games	68,75	66,67	87,5	100
	psycopg	46,19	86,94	42,47	69,35
	python	81,51	93,66	74,66	82,54
	swlibero	42,55	84,3	70,92	77,69
	webbit	78,13	98,89	30,4	87,81
	X-misc	20,39	43,38	24,27	43,37
SVM	debian	5,53	32,19	35,57	50,64
	dtm	58,42	81,16	61,8	85,51
	eva	33,57	68,33	67,14	80
	games	37,5	66,67	56,25	83,33
	psycopg	19,18	17,08	29,68	52,76
	python	39,04	65,08	63,01	74,6
	swlibero	4,25	8,26	49,66	47,11
	webbit	40,55	49,22	48,62	58,53
	X-misc	0	0	16,47	30,12

Tabella 8.4. Risultati percentuali della correttezza di classificazione sull'intero insieme di messaggi a disposizione

Per quanto riguarda il filtro di “Vapnik”, i risultati confermano quanto affermato in precedenza: occorre una base di conoscenza più significativa per ottenere gradi elevati di affidabilità.

Con l'utilizzo di un filtro multi-classe, mediamente e con dieci messaggi di addestramento, si ottengono risultati migliori nel filtro di Bayes ed in SVM, rispettivamente

si ha un incremento circa dell'11% e del 21%. Nel caso del filtro SVM si è praticamente raddoppiata l'affidabilità del classificatore. Infatti si ha una conoscenza migliore sulla composizione delle varie categorie. Mentre nel caso binario si solamente le informazioni riguardanti la singola categoria, nel multi-classe si ha una visione totale dei vari insiemi e quindi, soprattutto con domini di conoscenza di piccole dimensioni, come in questo caso, si ottengono prestazioni migliori. Nel caso di NBayes, invece, si hanno prestazioni leggermente inferiori al caso binario ma comunque paragonabili (decremento intorno all'1%).

Aumentando la dimensione del “training set” si presenta quanto era nelle aspettative. L'incremento delle prestazioni è notevole e si ottengono ottimi risultati soprattutto in alcune categorie. Infatti iniziano a presentarsi dei valori superiori al 90%. Si prenda ad esempio la categoria *dtm*. Nei filtri bayesiani arriva ad ottenere il 100% di corretta classificazione. Comportamento simile si osserva anche nel filtro SVM che passa da un 58% all'81% triplicando il dominio di conoscenza. Rimangono tuttavia, anche se con peso minore, i problemi di correlazione tra le categorie citate in precedenza. Il filtro NBayes raggiunge con questo insieme di conoscenza dei risultati ottimi, visto che si ha mediamente una correttezza prossima all'85%. Il filtro SVM raddoppia le sue prestazioni precedenti, ma tuttavia rimane al di sotto del 50% di corretta classificazione.

Utilizzando il multi-classe, invece, il filtro puramente bayesiano migliora di poco le proprie prestazioni ma, in questo caso, assume dei valori inferiori rispetto al duale binario. NBayes, invece, mantiene un comportamento simile rispetto al dominio di conoscenza più ristretto. Si ha infatti un netto miglioramento rispetto ai dieci messaggi per categoria, ma, comunque, le prestazioni del filtro multi-classe rimangono, di pochi punti percentuali, al di sotto del caso binario. Nettamente migliore, invece, è il multi-classe SVM. Le prestazioni totali infatti superano il 62% e si cominciano ad intravedere dei netti miglioramenti all'interno delle singole categorie. In base a quanto riscontrato si possono esprimere le seguenti considerazioni:

1. il filtro puramente bayesiano soffre l'elevato numero di categorie con limitati insiemi di conoscenza;
2. il filtro NBayes ottiene ottimi risultati nonostante l'esiguo numero di messaggi utilizzati per allenare il filtro soprattutto con il metodo binario, mentre ha bisogno di una base di conoscenza più grande per il metodo multiclasse;
3. il filtro SVM ha risultati molto bassi con pochi messaggi, ma le sue prestazioni tendono ad aumentare considerevolmente all'aumentare del dominio di conoscenza.

La crescita della base di conoscenza di un filtro con l'aumentare dell'esperienza conosciuta, in generale, migliora le prestazioni di classificazione, come dimostrato

nella tab. 8.4. Si è testato il funzionamento dei tre filtri con la classificazione delle mail della cartella di prova *amici*. Nonostante questa simulazione possa non avere un notevole valore statistico, in quanto l’insieme delle mail è ridotto, ha, però, la funzione di simulare l’efficacia di classificazione in un esempio reale di funzionamento dei filtri. L’apprendimento scelto è stato l’insieme delle basi di conoscenza delle nove categorie sopra elencate. Si è inoltre testato un apprendimento crescente, partendo da venti mail per categoria, in modo da mostrare gli eventuali adattamenti del filtro al crescere della descrizione degli insiemi. La tabella è suddivisa per filtro utilizzato, metodo di classificazione e mail. Queste sono numerate in base alla sequenza in cui sono state elaborate e il nome generico di Mail. Per stabilire l’esatta classificazione dei messaggi si è effettuata un’analisi preventiva degli insiemi e si è espresso un giudizio possibile di classificazione. I risultati ottenuti, pertanto, sono stati confrontati con il giudizio espresso a priori. Il simbolo \checkmark indica la corretta classificazione della mail.

#Mail	Tipo	Bayes				NBayes				SVM			
		20	40	60	80	20	40	60	80	20	40	60	80
Mail 1	Binario	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
	multi-classe	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
Mail 2	Binario	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
	multi-classe	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
Mail 3	Binario	-	-	-	-	-	-	\checkmark	\checkmark	-	-	-	-
	multi-classe	-	-	-	-	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
Mail 4	Binario	\checkmark	-	-	-	-							
	multi-classe	\checkmark	\checkmark	\checkmark	\checkmark	-	-	-	\checkmark	\checkmark	-	-	\checkmark
Mail 5	Binario	\checkmark											
	multi-classe	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark
Mail 6	Binario	-	-	\checkmark	-	-	\checkmark	\checkmark	\checkmark	-	-	\checkmark	\checkmark
	multi-classe	\checkmark	-	-	-	\checkmark	\checkmark	\checkmark	\checkmark	-	-	-	\checkmark
Mail 7	Binario	-	-	-	\checkmark	\checkmark	-	-	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark
	multi-classe	\checkmark	-	\checkmark	\checkmark								
Mail 8	Binario	-	\checkmark	-	-	\checkmark	-	-	-	-	-	\checkmark	\checkmark
	multi-classe	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark
Mail 9	Binario	\checkmark	-	\checkmark	\checkmark	\checkmark							
	multi-classe	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark	-	\checkmark	\checkmark	\checkmark

Tabella 8.5. Risultati di classificazione sull’analisi della cartella Amici

È interessante notare come il metodo multi-classe ha dei comportamenti medi migliori

del metodo binario e l'utilizzo di un addestramento più completo fornisce dei risultati di classificazione più attendibili. Inoltre, si è testato il funzionamento dei classificatori in presenza di mail che in apparenza non appartengono a nessuna delle categorie sopra indicate, questo è il caso della mail contrassegnata con l'etichetta "Mail 6". Se una mail in un processo di classificazione non viene rappresentata da nessuna delle categorie di addestramento, essa crea un nuovo insieme virtuale chiamato "Unknown", ossia non conosciuto. Tutti i filtri implementati tendono ad esprimere un loro apprezzamento di classificazione di un documento su una data categoria. L'implementazione costruita determina, in modo intrinseco, l'insieme degli elementi non classificabili. Questa è garantita dall'utilizzo di una soglia decisionale con la quale si tende a selezionare lo spazio delle possibili classificazioni. Pertanto, la mail in questione rappresenta un ottimo banco di prova per il corretto funzionamento dei filtri. Si può riscontrare un'ottima risposta ottenuta dal filtro NBayes, mediamente sia con il metodo binario che quello multi-classe. Inoltre, come espresso in precedenza, il filtro SVM è fortemente condizionato dalla base di conoscenza. Il filtro del Naive Bayes, invece, ha un andamento meno costante in tempi prestazionali rispetto ai due precedenti, esso infatti risente molto dei problemi di sbilanciamento. Il filtro, comunque, tende ad esprimere una sua preferenza a causa della soglia più bassa e all'assenza di una categoria in fase di addestramento etichettata come "Unknown". L'istogramma, inoltre, mostra i miglioramenti di classificazione al crescere del "training set".

Un'elaborazione sul contenuto del messaggio e una eventuale pulizia del numero di elementi presenti in esso porta, come visto in precedenza, a una diminuzione del tempo utile per la classificazione del singolo messaggio. La pulizia delle parti ritenute inutili ai fini della classificazione dovrebbe anche portare ad un lieve miglioramento della correttezza di classificazione, in quanto si eliminano i termini che hanno un basso valore entropico per la comunicazione, rif. 6. Nelle tab. 8.6 e 8.7 sono rappresentati i risultati di classificazione dei filtri data una base di apprendimento di due categorie, *Debian* e *Webbit*. Come riscontrato nella tab. 8.4, la categoria *Debian* presenta un alto fattore di correlazione con tutti gli altri insiemi, poiché contiene gran parte dei domini concettuali espressi nelle altre categorie. Pertanto si può immaginare uno scenario con alto rumore sui risultati.

Il filtro puramente bayesiano si adatta benissimo allo scenario con poche categorie, molto meglio di come avveniva nello scenario multi-categorie, dove le simulazioni presentavano una KB di nove domini. Esso presenta, con un dominio di apprendimento di dieci mail, valori di classificazione che si attestano intorno all'80% e tende a rimanere costante a tale valore anche con una base di apprendimento crescente. Nonostante il filtro che estende la funzionalità del Naive Bayes con l'uso delle catene di Markov sia anch'esso un filtro bayesiano, i due hanno comportamenti differenti. Ciò a dimostrazione del fatto che, con una ridotta base di conoscenza, sia difficile riuscire a rappresentare l'insieme delle possibili conversazioni e questo giustifica un

<i>Filtro</i>	<i>Pulizia</i>	10	20	30	40	50	60	70	80
Bayes	nessuna	79,14	77,27	82,46	77,71	81,68	83,01	81,95	82,71
	firma	79,14	77,27	82,46	78,31	81,68	83,17	82,12	82,71
	stopwords	80,39	75,43	82,31	78,31	81,52	82,53	81,95	82,19
	entrambe	80,39	75,43	82,31	78,61	81,68	82,85	81,79	82,36
NBayes	nessuna	55,80	78,27	77,19	76,51	77,17	79,81	77,81	77,74
	firma	55,80	78,27	77,49	76,36	73,29	80,45	75,33	77,91
	stopwords	59,12	77,98	76,46	78,01	72,67	78,37	75,99	77,05
	entrambe	54,83	65,91	72,37	73,04	70,19	72,92	70,03	71,40
SVM	nessuna	48,48	66,76	73,54	71,23	78,73	86,06	81,46	83,39
	firma	47,24	67,47	78,22	80,57	81,83	84,29	81,46	83,73
	stopwords	45,99	64,49	74,42	83,13	85,87	86,86	86,26	85,45
	entrambe	46,27	59,23	78,95	79,07	85,29	86,06	85,46	84,39

Tabella 8.6. Risultati di classificazione con filtri di tipo binario su due categorie di addestramento Debian, Webbit

alto errore di classificazione su un apprendimento di dieci mail. La situazione migliora sensibilmente con il crescere del dominio di conoscenza fino a raggiungere una media dell’78%, fig. 8.3.

L’errore ottenuto con l’utilizzo del filtro di Bayes è comunque inferiore di qualche punto percentuale rispetto al NBayes, a dimostrazione del fatto che Bayes si adatta molto meglio ad insiemi bilanciati. Il filtro di “Vapnik”, come avviene per NBayes, decresce con il crescere della base di conoscenza l’errore di classificazione. Nella tab. 8.6 si è riportato il comportamento del filtro alla pulizia del messaggio dalle parti con basso valore entropico, quali “stopword” e firma. L’andamento riscontrato per i tre filtri descrive un comportamento comune, di miglioramenti poco percepibili del risultato di classificazione. Le firme, come espresso nel par. 6.5.4, presentano un alto livello entropico nella descrizione delle mail, in quanto tendono a costruire il concetto di persona. Infatti, modellando questa informazione, si crea un arco di relazione tra la mail in questione e la persona che l’ha inviata. Si può affermare che, talvolta, quando la firma è presente in forma comprensibile alla macchina, aggiunge entropia alla classificazione ed un filtro tende ad astrarre anche dalla stessa le “feature” che possono rappresentare meglio l’insieme. Le “stopword”, invece, non aggiungono informazione, anzi il caso limite è quello di ridurre l’informazione trasportata, pertanto, la loro cancellazione si rende necessaria soprattutto in presenza di testi lunghi, rif. 6.5.2. Generalmente un messaggio di posta ha un carattere telegрафico e, per tale motivo, le “stopword” rivestono un ruolo marginale nello stesso.

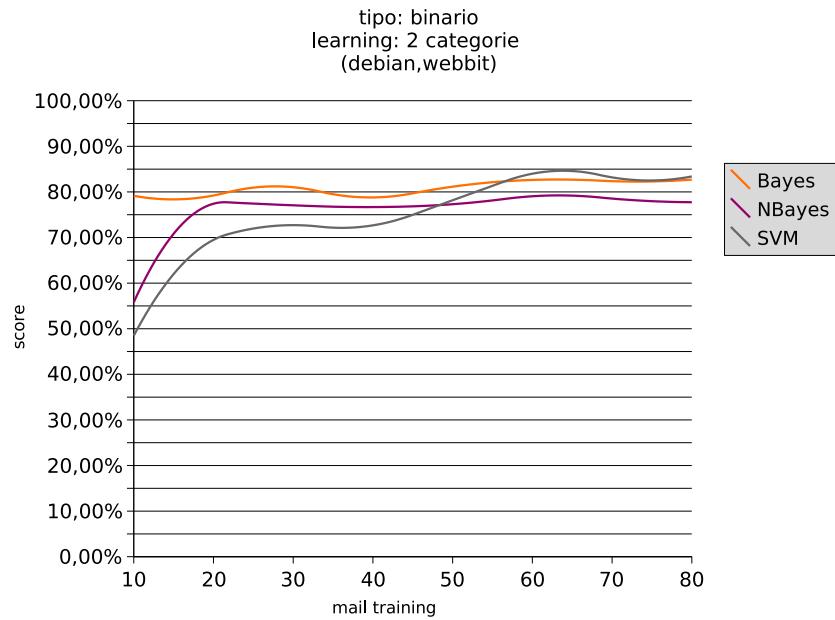


Figura 8.3. Grafico rappresentativo dell’andamento della classificazione multi-classe dei tre filtri sulle mail con addestramento dato dalle mail di Webbit e Debian

Filtro	Pulizia	10	20	30	40	50	60	70	80
Bayes	nessuna	72,10	66,48	62,13	57,83	52,48	50,64	44,04	42,62
	firma	72,10	66,48	62,28	57,68	52,80	50,96	44,21	42,81
	stopwords	69,75	64,35	62,43	58,73	54,97	53,69	48,18	47,09
	entrambe	70,17	65,06	62,72	59,04	55,28	53,85	48,51	47,09
NBayes	nessuna	53,18	66,48	71,93	73,95	70,03	72,28	71,85	71,06
	firma	53,18	66,34	71,93	73,64	70,50	72,60	72,85	71,40
	stopwords	55,25	66,05	72,37	73,34	70,03	72,92	70,20	71,23
	entrambe	54,83	65,91	72,37	73,04	70,19	72,92	70,03	71,40
SVM	nessuna	70,17	79,83	84,06	81,93	86,65	89,58	87,91	88,53
	firma	69,89	76,70	84,06	82,23	88,82	90,87	87,91	88,18
	stopwords	63,81	69,46	87,87	87,65	90,37	91,67	88,91	90,07
	entrambe	66,06	72,43	88,01	82,53	89,60	91,19	86,09	90,75

Tabella 8.7. Risultati di classificazione con filtri di tipo multi-classe su due categorie di addestramento Debian, Webbit

Il filtro puramente bayesiano, tenendo come riferimento lo stesso insieme di apprendimento, ha dei valori di classificazione inferiori rispetto al metodo binario di circa sei punti percentuali. Questo comportamento si riscontra anche per il NBayes, mentre SVM tende a diminuire sensibilmente l'errore di classificazione di circa il 20%. Solo per il filtro puramente bayesiano si riscontra fenomeni di “overfitting”, fig. 8.4, in quanto all'aumentare della base di conoscenza l'errore di classificazione aumenta. I restanti due filtri, invece, migliorano le loro prestazioni all'aumentare della base di conoscenza. Si è riscontrato lo stesso comportamento del metodo binario con l'utilizzo della pulizia del “body”.

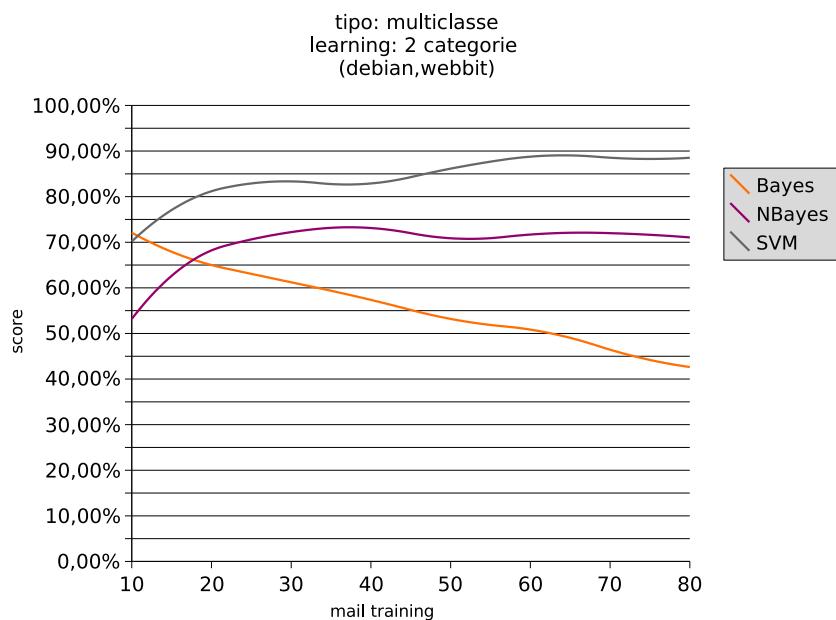


Figura 8.4. Grafico rappresentativo dell'andamento della classificazione multi-classe dei tre filtri sulle mail con addestramento dato dalle mail di Webbit e Debian

Lo scenario di simulazione ha, successivamente, riguardato prove multiple su tre categorie. In base alle verifiche effettuate sulle nove categorie si è evidenziato il comportamento limite del filtro puramente bayesiano, mentre nell'implementazione con due categorie si è riscontrato un comportamento ottimale. Pertanto si è verificato il limite massimo di funzionamento ottimale del filtro puramente bayesiano, inoltre si è osservato il comportamento dei filtri NBayes e SVM nello stesso scenario.

Si sono riportati i valori di classificazione dei tre filtri con i metodi binario e multi-classe nella tab. 8.8. Si evince che il filtro di Bayes ha delle prestazioni sbilanciate, con le cartelle *Python* e *Webbit* classificate con percentuali costantemente sopra

<i>Filtro</i>	<i>Cartella</i>	Binario				Multi-classe			
		20	40	60	80	20	40	60	80
Bayes	python	94,85	99,14	96,88	97,37	97,06	98,28	96,88	97,37
	swlibero	10,69	33,33	23,08	33,80	82,44	71,17	90,11	84,15
	webbit	100	92,06	98,81	98,5	50,54	75,96	72,21	73,32
NBayes	python	88,24	97,41	97,92	97,37	94,85	98,28	97,92	98,68
	swlibero	74,81	73,87	87,91	88,73	75,57	73,87	91,21	88,73
	webbit	98,48	85,71	95,49	98	50,76	75,51	76,72	80,05
SVM	python	91,18	81,03	78,13	81,58	92,65	95,69	91,67	90,79
	swlibero	16,79	40,54	53,85	59,44	57,25	61,26	73,63	77,46
	webbit	55,31	62,13	67,93	62,84	73,1	75,28	77,91	80,05

Tabella 8.8. Risultati di classificazione con filtri di tipo binario e multi-classe su tre categorie di addestramento Python, SoftwareLibero, Webbit

il 90% in tutti gli scenari di “learning” previsti per il metodo binario. La cartella *SoftwareLibero* presenta una percentuale mediamente bassa, con un fenomeno di “overfitting” tra i quaranta ed i sessanta di “learning”. Il filtro in questione tende a concentrare le “feature” sui due insiemi sopra citati, sbilanciando l’insieme decisionale. Questo può essere motivato dalla presenza di elementi, occorrenze dell’insieme *SoftwareLibero*, poco caratterizzanti per lo stesso. Ciò è dimostrato dai risultati del filtro SVM, in quanto la cartella in questione presenta mediamente i risultati più bassi rispetto alle altre due, con errore di classificazione che supera anche il 40% con uno scenario di apprendimento di ottanta mail. Il filtro NBayes ha, però, dei comportamenti più equilibrati rispetto agli altri due filtri con il metodo binario. La dimostrazione del risultato in questione può ricercarsi dentro al costruzione del vocabolario di “ngram”, i quali tendono a rappresentare meglio le relazioni che occorrono tra le parole stesse di una conversazione. La figura 8.5 indica l’andamento dei filtri binari.

Lo scenario multi-classe non presenta i casi limite del fenomeno di classificazione delle due categorie, pertanto, il filtro bayesiano puro presenta dei valori di classificazione che risultano costanti intorno alla mediana del 85%, con un apprendimento che supera le quaranta unità. Pertanto, il fenomeno riscontrato con la categoria *SoftwareLibero* non si ripresenta in questo scenario, a dimostrazione di quanto detto sopra. Si nota inoltre un comportamento di eccellenza del filtro NBayes con un addestramento crescente per tutte le categorie, mentre si verifica un netto incremento delle prestazioni ottenute con il filtro SVM rispetto al caso binario, questo come dimostrazione di una maggiore efficienza del metodo multi-classe. Si faccia riferimento alla figura 8.6 per la verifica degli andamenti.

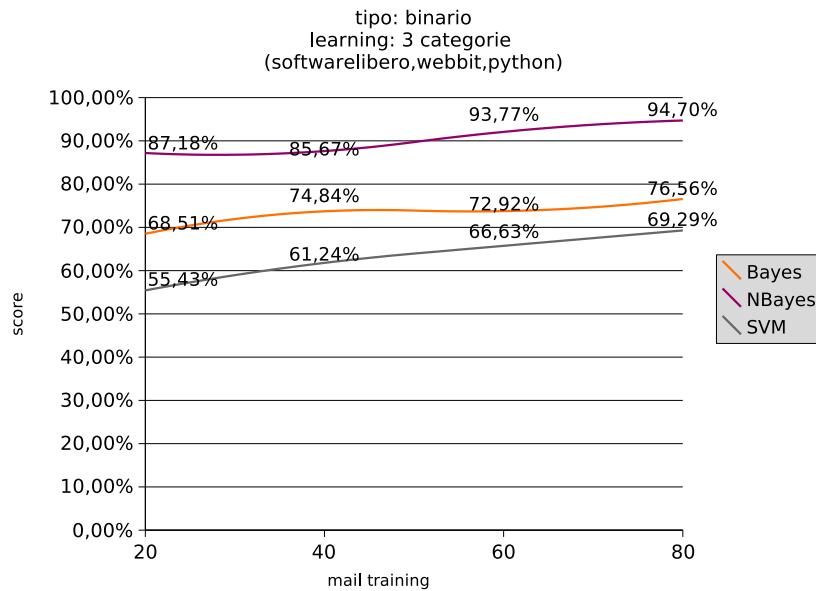


Figura 8.5. Grafico rappresentativo dell'andamento della classificazione dei tre filtri con il metodo binario, con addestramento dato dalle mail di Python, SoftwareLibero, Webbit

L'utilizzo delle tre categorie di addestramento offre dei risultati ancora soddisfacenti per la classificazione su tutti i filtri. Lo scenario con quattro categorie, invece, rappresentato nella tab. 8.9 presenta bassi valori di classificazione per il filtro Bayes. Pertanto, a livello sperimentale, possiamo affermare che in scenari complessi di insiemi non totalmente disgiunti e con rumore molto alto, ha percentuali di errore troppo alte. Questo fenomeno si presenta sia con il metodo binario che quello multi-classe. L'andamento dei filtri di NBayes ed SVM segue il dato sperimentale ottenuto con la classificazione su tre categorie. Infatti, i valori ottenuti con l'utilizzo del filtro di NBayes sono, anche con un basso addestramento, superiori al 90% di correttezza. Mentre con il filtro di “Vapnik” si ottiene un rumore molto alto con bassi valori di apprendimento, come verificato nelle precedenti simulazioni, si ottiene l'80% di correttezza di classificazione con un addestramento dato da trenta unità. La figura 8.7 mostra l'andamento ottenuto.

Lo scenario multi-classe presenta per il filtro puramente bayesiano gli stessi sintomi del metodo binario, infatti si ha un errore medio sulle quattro categorie superiore al 50%. Inoltre si riscontra solo una classificazione accettabile per la categoria *SoftwareLibero*, la quale presenta un maggior grado di occorrenze di caratterizzazione dell'insieme stesso nello scenario di simulazione descritto. Le altre, infatti, presentano un rumore molto alto che limita le prestazioni del filtro stesso. Invece i

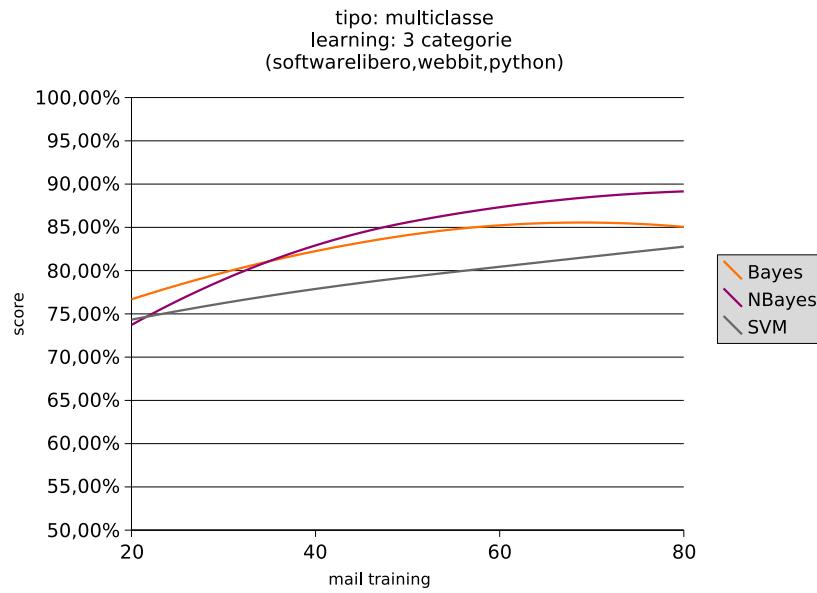


Figura 8.6. Grafico rappresentativo dell'andamento della classificazione dei tre filtri con il metodo multi-classe con addestramento dato dalle mail di Python, SoftwareLibero, Webbit

Filtro	Cartella	Binario				multi-classe			
		10	30	50	70	10	30	50	70
Bayes	dtm	89,88	100	97,96	44,23	46,06	55,07	53,06	58,62
	eva	10,71	7,5	8	28,75	76,28	65	26	25
	python	56,16	65,07	35,85	58,14	14,38	20,63	14,15	15,12
	swlibero	77,3	71,07	69,3	75,31	81,56	76,33	79,2	80,25
NBayes	dtm	93,25	100	100	100	77,52	98,55	100	100
	eva	84,28	92,5	94	92,5	96,43	100	100	100
	python	93,83	95,23	95,28	96,51	84,93	88,88	95,28	96,51
	swlibero	86,52	85,12	93,07	93,83	83,69	81,81	91,09	91,36
SVM	dtm	42,7	88,41	87,76	100	70,79	73,91	91,84	100
	eva	65,71	81,67	88	85	71,43	84,17	87	85
	python	67,12	80,95	67,92	77,91	73,99	84,13	83,96	87,21
	swlibero	0	47,93	68,32	82,72	92,91	91,74	90,1	92,59

Tabella 8.9. Risultati di classificazione con filtri di tipo binario e multi-classe su tre categorie di addestramento Dtm, Eva, Python, SoftwareLibero

filtri NBayes e SVM seguono la strada tracciata dal metodo binario, fornendo delle prestazioni ancora migliori con un addestramento sempre crescente. L’andamento è rappresentato nella figura 8.8. Con un addestramento di dieci messaggi il filtro di “Vapnik” ha un risultato di classificazione in media di dieci punti percentuali, a dimostrazione di quanto NBayes sia anche efficace nelle classificazioni con una ridotta conoscenza degli insiemi, mentre la crescita della KB tende a schiacciare la distanza fino a pochi punti percentuali.

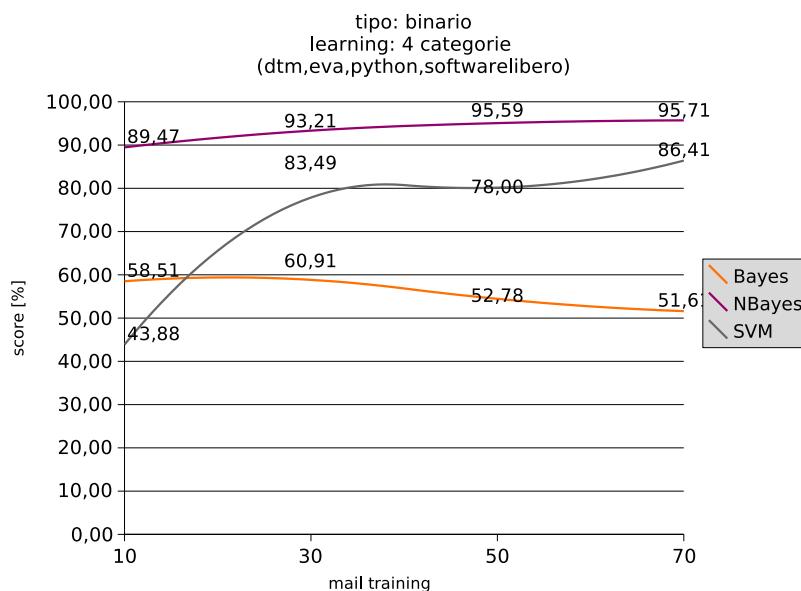


Figura 8.7. Grafico rappresentativo dell’andamento della classificazione dei tre filtri con metodo binario sulle mail con addestramento dato dalle mail di Dtm, Eva, Python, SoftwareLibero

L’ultima fase di test è dedicata all’analisi della risposta adattativa del sistema. Si è cercato di simulare il comportamento che potrebbe assumere il sistema con interazione da parte dell’utente. Ad ogni errore il sistema viene nuovamente istruito e viene inserito il messaggio, precedentemente classificato sbagliato, all’interno della base di conoscenza del dominio. Praticamente si emula l’azione che l’utente compie quando, vedendo un errore nella classificazione, sposta il messaggio nella cartella corretta. Il test è stato inoltre ripetuto aumentando il numero di categorie in esame. Si è scelto di concentrare la classificazione sulla cartella *debian* poiché, nei test precedenti, ha presentato valori di classificazione più bassi rispetto alle altre categorie. Inoltre, in questa fase, l’attenzione si è concentrata esclusivamente sui filtri NBayes ed SVM. In generale si aspetta un deciso miglioramento nelle prestazioni in quanto i “training set” vengono migliorati in corso di simulazione. Rimane tuttavia presente

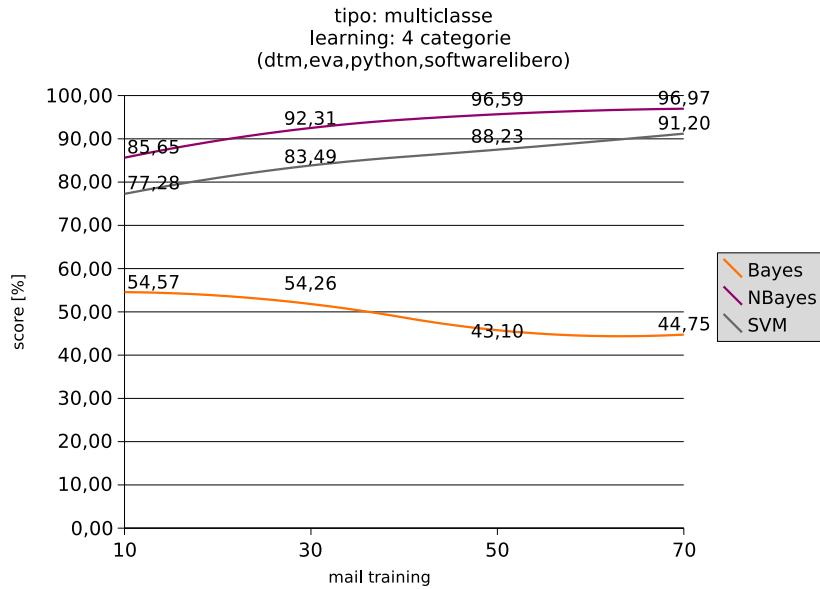


Figura 8.8. Grafico rappresentativo dell’andamento della classificazione dei tre filtri con metodo multi-classe sulle mail con addestramento dato dalle mail di Dtm, Eva, Python, SoftwareLibero

un insieme di addestramento iniziale, utile per poter inizializzare il sistema.

Filtro	#Addestramento	Non adattativo		Adattativo	
		10	60	10	60
NBayes	3	83,4	85,71	74,31	83,25
	4	85,38	86,2	66,8	68,47
	5	83,25	85,77	60,08	66,25
	6	88,24	91,7	48,22	68,96
SVM	3	11,97	70,94	86,56	76,35
	4	15,41	64,53	81,81	73,4
	5	5,53	54,19	74,31	72,28
	6	4,74	47,92	73,81	75,34

Tabella 8.10. Confronto test adattativi e non attraverso l’uso di filtri binari

Si prendano in esame i risultati prodotti nel filtro NBayes nella sua forma binaria. Nei test non adattativi, ossia quelli effettuati senza riclassificare i messaggi

Filtro	#Addestramento	Non adattativo		Adattativo	
		10	60	10	60
NBayes	3	47,36	67,98	88,14	86,21
	4	42,69	67,98	89,72	87,19
	5	39,53	58,62	90,19	85,71
	6	39,92	61,08	89,72	83,73
SVM	3	69,17	78,33	93,28	87,69
	4	57,61	79,31	90,91	86,21
	5	43,87	72,91	89,32	85,71
	6	49,41	70,44	88,93	87,19

Tabella 8.11. Confronto test adattativi e non attraverso l’uso di filtri multiclass

etichettati erroneamente, i risultati sono simili ai precedenti e si attestano generalmente intorno all’85%, indipendentemente dal numero di categorie presenti e con un insieme di conoscenza pari a dieci messaggi. Nel test adattativo, tuttavia, le prestazioni peggiorano poichè il dominio di conoscenza cresce solamente per la categoria *debian*, essendo la categoria di cui si vuole analizzare il comportamento. Infatti risultano essere presenti dei domini di conoscenza molto sbilanciati tra loro. Ossia una categoria conterrà al suo interno un numero maggiore di esempi rispetto alle altre. Questo provoca il decremento dei risultati per quanto riguarda il filtro binario NBayes. Si possono infatti notare gli effetti che l’aumento di categorie provoca nel test adattativo e in quello non adattativo. Mentre in quest’ultimo caso all’aumentare del numero dei domini di conoscenza non si verifica una diminuzione sulla correttezza di classificazione, nel caso adattativo le prestazioni scendono di molto. Anche aumentando il dominio di conoscenza iniziale a sessanta messaggi si nota un comportamento simile, tuttavia il peggioramento è meno marcato. Infatti, in quest’ultimo caso, i domini di conoscenza risultano essere meno sbilanciati rispetto al test precedente.

Nel caso multiclass, invece, si osserva un comportamento opposto rispetto al binario. Si passa da risultati intorno al 40% per il caso non adattivo a circa il 90% del caso multiclass. Anche qui valgono le stesse considerazioni espresse in precedenza. Visto il numero di categorie nel caso non adattativo si ha un insieme molto sbilanciato, problema che invece si risolve nel caso adattivo. Infatti i messaggi che vengono riclassificati vanno a bilanciare il sistema fino ad ottenere una situazione più bilanciata per la categoria *debian*.

Si passi ora a considerare le prestazioni del filtro di “Vapnik”. Nei test non adattivi possiamo notare il comportamento già verificato nei test precedenti. Con insiemi di addestramento composti da pochi elementi, soprattutto nel caso binario, il filtro

soffre la scarsa conoscenza del dominio. All'aumento dell'esperienza acquisita, le prestazioni assumono valori decisamente migliori. Tuttavia la cartella *debian*, visti i problemi già riscontati in precedenza, soffre la scarsa consistenza dell'insieme di conoscenza. Molto interessanti sono i risultati che si presentano con dieci messaggi di addestramento. La percentuale di correttezza in presenza di sei categorie passa da un 4,74% nel caso non adattivo a un 73,81% nel caso adattivo. Occorre comunque tener presente che tale valore indica i messaggi che sono stati classificati correttamente senza la necessità di essere riclassificati. Quindi nella parte iniziale del test adattativo si avranno molte riclassificazioni, mentre negli ultimi messaggi tale numero si ridurrà drasticamente. Si noti a riguardo la tabella 8.12. I risultati mostrano l'elevato grado di affidabilità del filtro SVM con insiemi di classificazione ben formati. Si è scelto in questo caso di porre all'attenzione solamente il filtro multiclasse poiché assume prestazioni migliori rispetto al duale binario. Nell'ultimo test sono stati classificati i primi centocinquanta messaggi, ossia circa la metà dei messaggi a disposizione per la cartella *debian*. Dopo questa fase sono riclassificati, inserendoli come esempi positivi, i messaggi che hanno presentato errori di classificazione. Come nei casi precedenti, i primi messaggi classificati ottengono basse percentuali di correttezza di classificazione, pari mediamente al 45%. Si osserva che, dopo la fase di riapprendimento, i nuovi messaggi vengono classificati con una percentuale molto alta rispetto alla prima fase, che indica la presenza di un buon insieme di addestramento. Nel filtro SVM ci si attesta a valori sempre superiori al 90%. I valori NBayes sono stati leggermente inferiori in quanto il maggior numero di messaggi erroneamente etichettati nel test puramente adattativo si trovano in numero maggiore rispetto ad SVM nell'ultima parte della cartella. Rimangono tuttavia ottimi i risultati di classificazione visto soprattutto l'incremento di prestazioni ottenuto su categorie strettamente correlate tra loro.

<i>Filtro</i>	<i>Riclassificazione</i>	# categorie			
		3	4	5	6
Nbayes	prima	54,66	54,66	44	45,33
	dopo	81,55	80,59	79,61	75,72
Svm	prima	50,67	52,00	42,67	46,67
	dopo	93,21	93,20	91,26	94,17

Tabella 8.12. Risultati percentuali di classificazione con riclassificazione dei messaggi dopo i primi centocinquanta messaggi con filtri multiclasse

Ritornando ai risultati riportati in tabella 8.11 si possono notare risultati migliori nel caso in cui il gruppo di messaggi iniziali di classificazione è pari a dieci messaggi rispetto a sessanta nel caso adattativo. Questo si spiega con il fatto che il dominio

di conoscenza viene popolato in maniera mirata, non presentando messaggi all'interno della base di conoscenza che verrebbero comunque classificati correttamente. In tal modo la dimensione del “traininig set” rimarrà più contenuta e al tempo stesso più rappresentativa. Dai risultati emersi si conferma che le SVM non soffrono la presenza di domini di conoscenza sbilanciati, ma al tempo stesso lavorano ottimamente se il gruppo di messaggi, utilizzati per rappresentare le categorie, viene scelto accuratamente.

Capitolo 9

Conclusioni

Dal punto di vista applicativo, questa architettura presenta un nuovo approccio alla gestione delle mail. Si è presentata quasi come una sfida ai moderni fornitori dei servizi di posta, che espongono i loro servizi evoluti mediante l'uso dei Web Service, cap. 2. Non ha la pretesa, però, di segnare un cambiamento in termini di gestione della posta lato fornitore o lato utilizzatore. Si è cercato comunque di proporre una metodologia nuova per risolvere il problema della classificazione della posta, utilizzando le tecniche attuali di classificazione che lavorano bene su documenti con un'altissima cardinalità di parole ma che, a priori, presentavano dei problemi nella classificazione di documenti molto corti e di diversa natura linguistica. Inoltre la volontà di utilizzare un base dati ontologica propone un cambiamento di tendenza nelle archiviazioni delle informazioni e apre la possibilità di effettuare dei ragionamenti direttamente su questi insiemi.

Lo studio dell'arte degli attuali strumenti di posta ha portato ad analizzare i metodi di gestione che hanno più successo ed ad individuare la necessità di utilizzare delle macchine pensanti che siano in grado di risolvere il problema di grandi quantità di informazione che crescono di giorno in giorno. Basti notare come in questi ultimi anni l'uso della posta sia diventato una peculiarità dell'utenza professionale ma anche dei comuni utilizzatori domestici, quasi da sostituire il mezzo di comunicazione più utilizzato attualmente: il cellulare.

Lo studio sulle "machine learning" e l'analisi degli strumenti di classificazione nel settore dell'intelligenza artificiale ha portato alla scelta di due strade: quella basata su analisi prettamente statistica e quella che prevede l'utilizzo di vettori di supporto. Queste rappresentavano gli antipodi in termini di velocità di esecuzione, si è quindi effettuato anche uno studio di fattibilità. Ciò, infatti, non rappresenta un particolare problema nell'analisi di documenti il cui risultato non deve essere fornito in tempo reale, ma nel contesto in cui si trova questa architettura costituisce una base decisionale fondamentale, tanto da privilegiare la prestazione alla correttezza della classificazione. Si è pertanto scelto di implementare due filtri bayesiani, che fondano

le loro radici nello studio statistico, il Naive Bayes e il NBayes, una derivazione del precedente, ma che utilizza al suo interno la teoria d'inferenza di Markov. Allo stesso tempo si è deciso di utilizzare l'SVM, un metodo molto robusto e testato che presenta la più alta percentuale di classificazione tra le attuali ricerche di classificazione effettuate.

Il grosso problema è stato fornire una base dati in cui salvare le informazioni. Si è deciso di utilizzare uno “storage RDF”, basato sul concetto dell’ontologia. Esso bene si adatta agli attuali cambiamenti del mondo del Web, che sempre più migra verso questa soluzione per permettere una migliore indicizzazione e una migliore esposizione dei suoi contenuti.

Le simulazioni condotte sono state svolte su di un insieme di mail, organizzate in nove cartelle, le quali al loro interno avevano un numero variabile di mail di apprendimento. Nonostante si sia cercato, soprattutto grazie a Federico Di Gregorio, di utilizzare un insieme quanto più generico e grande possibile, la disponibilità di un numero limitato di mail può aver condizionato i risultati ottenuti. Questo a causa di un’assenza nel mondo della ricerca di gruppi di mail organizzati gerarchicamente, come avviene invece per i testi, a tal fine riportiamo i nomi di alcuni “dataset” come *DMoz*, *TaxE* e *ACM*. Pertanto non si esclude un miglioramento dell’efficacia di classificazione con l’utilizzo di insiemi più ampi.

Nonostante queste difficoltà si è potuto verificare che l’utilizzo di questi strumenti possono fornire all’utilizzatore finale un reale aiuto nella classificazione delle proprie mail.

In base ai risultati ottenuti nei test si possono dedurre le seguenti considerazioni:

1. con un esiguo numero di categorie il filtro puramente bayesiano rappresenta un ottimo compromesso tra risultati e tempo di calcolo;
2. il filtro NBayes assume comportamenti ottimi già con pochi messaggi di apprendimento indipendentemente dal numero di categorie utilizzate;
3. il filtro SVM offre migliori risultati in situazioni con insiemi molto sbilanciati tra loro. La velocità di calcolo, tuttavia accettabile, è di gran lunga superiore ai filtri bayesiani. Può essere utilizzato soprattutto su applicativi residenti direttamente sulla macchina dell’utente finale vista l’elevata elaborazione di calcolo.

9.1 Sviluppi futuri

Nonostante il settore di ricerca sulla classificazione dei documenti sia abbastanza evoluto, esiste una concreta possibilità di estendere o migliorare quanto già sviluppato, soprattutto alla luce del numero di argomenti trattati. Si faccia riferimento alla struttura a blocchi interoperabili creata:

1. recupero informazioni;
2. filtri di classificazione, che rappresentano il cuore dell'applicazione;
3. presentazione dei dati all'utente.

Per il primo punto, possiamo suddividere il problema in due sotto parti: il recupero delle mail da un server di posta e l'interazione con la base dati ontologica. Il protocollo di comunicazione verso il server di posta implementato è l'IMAP, in quanto è più flessibile e dinamico, in uno scenario caratterizzato da un assiduo utilizzo della rete. In aggiunta, il protocollo IMAP, prevede la possibilità di utilizzare il server contenitore delle mail come “storage” di informazioni, mediante la creazione di “flag” o cartelle. Pertanto un possibile utilizzo di questa caratteristica potrebbe diversificare la scelta di quale base dati utilizzare. Vi è, comunque, la possibilità di implementare il protocollo POP3, con l'aggiunta di una libreria, per dare continuità agli utilizzatori meno propensi all'innovazione (si faccia ad esempio riferimento al settore lavorativo, nel quale si è molto legati al prodotto che si è utilizzato per anni). Vi è inoltre la possibilità di estendere l'infrastruttura con eventuali protocolli futuri, sia di tipo testuale che non.

La scelta di utilizzare una base dati ontologica è stata dettata sia da motivi di rappresentazione ed esposizione dei dati in domini accessibili all'interno di una rete di ontologie, sia per la possibilità di sfruttare le caratteristiche intrinseche di queste ultime, come ad esempio la possibilità di effettuare ragionamenti di classificazione o di ricerca all'interno con l'utilizzo di “reasoner” che siano in grado di sfruttare la struttura ad albero dell'ontologia.

Una scelta effettuata in fase di progettazione è stata quella di creare una struttura gerarchica, nella quale ogni livello comunica con i livelli strettamente adiacenti. Un possibile sviluppo è sicuramente quello di implementare nuovi filtri di classificazione. Per quanto riguarda quest'ultimo punto, considerato lo scenario in cui deve trovarsi l'infrastruttura, vi è l'impossibilità di sapere con certezza se si lavora con insiemi bilanciati o meno, rif. 7.3. Infatti, in un classico utilizzo della posta, generalmente la suddivisione delle stesse in cartelle è alquanto casuale e lo spazio delle “feature”, che rappresentano, è anch'esso casuale, pertanto si può avere uno scenario sbilanciato. In questi contesti i filtri di classificazione dei documenti hanno problemi di convergenza e nonostante si sia utilizzato una struttura pesata, che potesse bilanciare mediante funzioni lineari lo spazio dimensionale, essa comunque può essere migliorata. Un'altra possibile implementazione può prevedere la cooperazione dei filtri. Infatti, ci sono tre essenziali componenti di pesi che possono avere un impatto sulle prestazioni di classificazione(Cfr. [14]). I componenti sono:

1. l'efficacia globale, che misura l'efficacia di un classificatore mentre elabora documenti non etichettati prima;

2. l'efficacia locale, che misura l'efficacia di un classificatore mentre elabora documenti di un generico dominio o categoria;
3. decisione di confidenza, esprime un giudizio sulla confidenza del classificatore sul quel particolare insieme di documenti elaborati.

Una probabile strada da seguire potrebbe essere quella di mettere insieme le tre componenti sopra citate.

La quantità di informazione (intesa come capacità di descrivere contenuti non noti a priori) contenuta in una mail non è distribuita uniformemente. Si è parlato nei capitoli precedenti di livelli entropici. Un possibile sviluppo di questa infrastruttura potrebbe prevedere l'utilizzo di altri livelli entropici all'interno del classificatore. Differenemente da come affermato in (Cfr. [5]), è possibile usare altre informazioni contenute negli "header" della mail per migliorare le prestazioni del classificatore. Ma diversamente dall'esperimento condotto in (Cfr. [19]), nel quale si ricorreva solo all'uso del "subject" (in questa infrastruttura esso determina la creazione di un flusso di conversazione, pertanto fornisce un aiuto al classificatore), è possibile inserire altri livelli, creando così un albero dei pesi di scelta. Da prevedere anche l'utilizzo di interpreti di allegati, anch'essi fanno parte del contenuto informativo di una mail. Le prestazioni di un generico classificatore tendono a migliorare quando viene effettuato lo "stemming". Esso infatti, in contesti di classificazioni di testi con un dominio bilanciato, fornisce delle prestazioni che migliorano il risultato di qualche punto percentuale, soprattutto nel caso del filtro di Vapnik. Esso, pertanto, rappresenta un possibile sviluppo futuro. Nel riconoscimento dello "stemming" si incontra il problema del riconoscimento lingua. Infatti, ogni lingua è caratterizzata dall'avere un suo vocabolario le cui parole sono create da una radice. In questo scenario lo "stemming" può presentare dei problemi di funzionamento, in quanto si trova ad avere due parole che hanno lo stesso significato ma espresse con due o più lingue differenti e con radici di conseguenza diverse. Nelle simulazioni effettuate si è citato il problema di avere più forme linguistiche all'interno di uno stesso insieme di apprendimento. Un possibile sviluppo futuro può prevedere, in questi contesti, la traduzione delle diverse lingue della categoria in una di riferimento.

Nella fase di apprendimento si è deciso di creare delle categorie ed etichettarle come esclusive o non. Sfruttando le potenzialità del motore RDF, si potrebbe costruire un albero delle categorie esclusive in modo da creare cammini, la cui visita potrebbe determinare la relazione tra esse e quindi una più efficace gestione delle etichette create.

Ultimo in ordine di citazione, ma non meno importante del resto, i maggiori miglioramenti possono essere operati nella presentazione dei dati all'utente, gestendo la formattazione utilizzando sempre il WSDL, ed offrendo la possibilità di utilizzo di un WebService o di un client per la raccolta e la gestione dei dati personali.

Appendice A

Libsvm

Esistono vari progetti che implementano la teoria sulle macchine a vettori di supporto descritta nel capitolo 4. Queste varie implementazioni differiscono tra di loro per:

- il tipo di licenza con cui sono distribuite,
- i linguaggi di programmazione utilizzati,
- il supporto alla classificazione multiclass,
- l'output fornito all'utente.

Per l'architettura implementata si è scelto tra le varie opzioni a disposizione la libreria LibSvm (Cfr. [6]). LibSvm è una libreria che offre:

- classificazione dei support vector mettendo a disposizione i kernel di tipo C-SVC e nu-SVC,
- regressione con i kernel epsilon-SVR e nu-SVR,
- stima di distribuzione con one-class SVM.
- classificazione multi-classe.

LibSvm accetta i dati in entrata secondo il seguente formato:

label : feature1 : value feature2 : value..... featureN : value

“Label”, utilizzando un filtro di tipo binario, può assumere il valore +1 o -1 a seconda che il vettore rappresenti un esempio positivo (+1) o negativo (-1). Nel caso della classificazione multiclass questo termine indica la classe di appartenenza del

vettore quindi può assumere qualsiasi valore intero, cioè considerando un’insieme di n categorie il “label” assumerà un valore numerico che indicizza in modo univoco la categoria. Ovviamente vettori della stessa classe di categorizzazione avranno label identica. Il termine feature, nel nostro caso, rappresenta la parola contenuta nel testo del messaggio di posta elettronica in analisi e tale termine sarà l’indice che identifica tale parola all’interno del vocabolario (si veda la struttura della tabella riprodotta nella figura D.5). “Value” infine sarà il peso della parola all’interno della frase. Per semplicità “value” assumerà come valore l’occorrenza della parola in questione. Nella libreria SVM .NET è anche presente la possibilità di effettuare uno scaling sui dati. Questa operazione sui dati permette di ottenere delle prestazioni migliori, visto che viene effettuata per semplificare sia la complessità computazionale, che per evitare che “feature” con valori elevati contribuiscano troppo pesantemente rispetto agli attributi che presentano bassi valori. Per fare questo occorre, però, conoscere a priori la totalità delle “feature” che si presentano all’interno dei messaggi di posta elettronica. Come già detto in precedenza, il dizionario utilizzato cresce col presentarsi di nuovi elementi, ossia quando un termine si presenta per la prima volta all’interno del contenuto di un messaggio da analizzare esso viene inserito all’interno della base dati e indicizzato con un identificativo numerico univoco. Quanto appena descritto non permette di effettuare una normalizzazione sui dati in quanto si dovrebbe conoscere già a priori quale sarà lo spazio delle “feature” dei messaggi di posta elettronica che dovranno essere classificati, visto che lo stesso criterio di normalizzazione deve essere effettuato su tutti i messaggi in esame. Questo non è quindi possibile a meno di non avere dizionari per ogni lingua che contengono al loro interno tutte le parole che possono comparire all’interno della lingua. L’utilizzo di questi dizionari implica però un eccessivo spreco di memoria, ma soprattutto implica il fatto di riconoscere la lingua nel 100% dei casi visto che lingue differenti saranno contenute in tabelle differenti. Si potrebbe anche pensare di utilizzare un’unica tabella in cui inserire tutte le parole di ogni lingua, ma sembra una soluzione non vincente in partenza soprattutto per motivi di gestione. Ogni lingua nel tempo evolve e compaiono sempre nuovi termini per cui andare a gestire l’inserimento di nuovi elementi risulterebbe un’operazione molto scomoda. L’utilizzo di questi dizionari può presentare anche problemi nel caso in cui compaiono errori di battitura. In questi casi si dovrebbe cercare una soluzione nel caso in cui una parola sia stata scritta in modo errato all’interno del messaggio. Si può pensare di scartare la parola, oppure, tramite gli algoritmi 6.6.1, cercare il termine che potrebbe essere causa dell’errore. Naturalmente qui si presenta un notevole peso computazionale visto che si dovrebbero effettuare tanti confronti tra parole appartenenti alla stessa lingua. Avendo scelto una tecnica diversa ,che permette di non legarsi a dei dizionari “statici”, la normalizzazione dei dati non può quindi essere effettuata.

Appendice B

Tabelle

C:	telnet pop3.example.com 110
S:	+OK <22593.1129980067@example.com>
C:	USER pippo
S:	+OK
C:	PASS pluto
S:	+OK
C:	LIST
S:	+Ok 1 817 2 124 .
C:	RETR 1
S:	+OK Return-Path: <pippo@example.org> Delivered-To: pippo@example.org Date: Sat, 22 Oct 2005 13:24:54 +0200 From: Mario Rossi <mario@rossi.org> Subject: xxxx Content-Type: text/plain; charset=ISO-8859-1 testo messaggio .
C:	DELE 1
S:	+OK
C:	QUIT
S:	+OK

Tabella B.1. Esempio di comunicazione con server POP3

C:	telnet imap.example.com 143
S:	* OK Dovecot ready.
C:	* 001 login biagio@patrocloo.it password
S:	001 OK Logged in.
C:	002 select inbox
S:	* FLAGS (\Answered \Flagged \Deleted \Seen \Draft) * OK [PERMANENTFLAGS (\Answered \Flagged \Deleted \Seen \Draft *)] Flags permitted. * 36 EXISTS * 0 RECENT * OK [UIDVALIDITY 1180965894] UIDs valid * OK [UIDNEXT 54] Predicted next UID 002 OK [READ-WRITE] Select completed.
C:	003 fetch 1 rfc822
S:	* 1 FETCH (RFC822 476 Return-Path: <giuseppe@patrocloo.it> X-Original-To: biagio@patrocloo.it Delivered-To: biagio@patrocloo.it Received: from localhost (localhost [127.0.0.1]) by patrocloo (Postfix) with SMTP id 4735E38201 for <biagio@patrocloo.it>; Mon, 4 Jun 2007 21:59:48 +0200 (CEST) from: biagio@patrocloo to: biagio@patrocloo Subject: prova Message-Id: <20070604195955.4735E38201@patrocloo> Date: Mon, 4 Jun 2007 21:59:48 +0200 (CEST) ciao biagio, ti aspetto domani mattina alle 9.00 come d'accordo.) 003 OK Fetch completed.
C:	004 logout
S:	* BYE Logging out 004 OK Logout completed. +OK

Tabella B.2. Esempio di comunicazione con server IMAP

S:	220 smtp.example.com ESMTP Postfix
C:	HELO mydomain.com
S:	250 Hello mydomain.com
C:	MAIL FROM: <sender@mydomain.com>
S:	250 Ok
C:	RCPT TO: <friend@example.com>
S:	250 Ok
C:	DATA
S:	354 End data with <CR> <LF> .<CR> <LF>
C:	Subject: messaggio di prova From: sender@mydomain.com To: friend@example.com Ciao, prova prova .
S:	250 Ok: queued as 12345
C:	QUIT
S:	2 21 Bye

Tabella B.3. Esempio completo è di una comunicazione con un SMTP server

word	occurrency
ciao	1
oggi	1
partita	1
alle	1
11	1
ho	1
organizzato	1
tutto	1
e	2
ci	1
troviamo	1
al	1
solito	1
campo	1
comunicalo	1
anche	1
a	2
giuseppe	1
marco	1
berry	1

Tabella B.4. Un messaggio della categoria Amici

<i>word</i>	<i>occurrency</i>
ciao	2
berry	2
ha	1
organizzato	2
la	1
partita	2
alle	2
11	2
ci	2
troviamo	2
al	2
campo	2
saluti	1
biagio	1
oggi	1
ho	1
tutto	1
e	2
solito	1
comunicalo	1
anche	1
a	2
giuseppe	1
marco	1
<i>Totale</i>	36

Tabella B.5. Categoria Amici

<i>word</i>	<i>occurrency</i>
ciao	2
biagio	2
ti	1
invio	1
le	2
ultime	1
modifiche	2
al	2
progetto	2
di	2
bayes	2
inoltro	1
una	1
copia	2
anche	1
a	2
federico	2
presto	1
giuseppe	2
fatto	1
commit	1
di	1
svm	1
ho	2
e	2
visto	1
sentito	1
mi	1
ha	2
detto	1
che	1
non	1
ricevuto	1
la	1
<i>Totale</i>	48

Tabella B.6. CATEGORIA TESI

word	occurrency	$peso_{Amici}$	$peso_{Tesi}$
ciao	1	2	2
biagio	1	0	2
le	1	0	2
ultime	1	0	1
modifiche	1	0	2
al	1	2	2
progetto	1	0	2
di	1	0	2
bayes	1	0	2
sono	1	0	0
perfette	1	0	0
mi	1	0	1
sento	1	0	0
con	1	0	0
federico	1	0	2
così	1	0	0
gli	1	0	0
invio	1	0	1
la	1	1	1
copia	1	0	2
che	1	0	0
non	1	0	1
ha	1	1	2
ricevuto	1	0	1
intanto	1	0	0
continuo	1	0	0
su	1	0	0
svm	1	0	1
finisco	1	0	0
e	1	2	2
mando	1	0	0
un	1	0	0
commit	1	0	1
giuseppe	1	1	2

Tabella B.7. Simulazione di arrivo di nuova mail

Appendice C

Grafici delle simulazioni

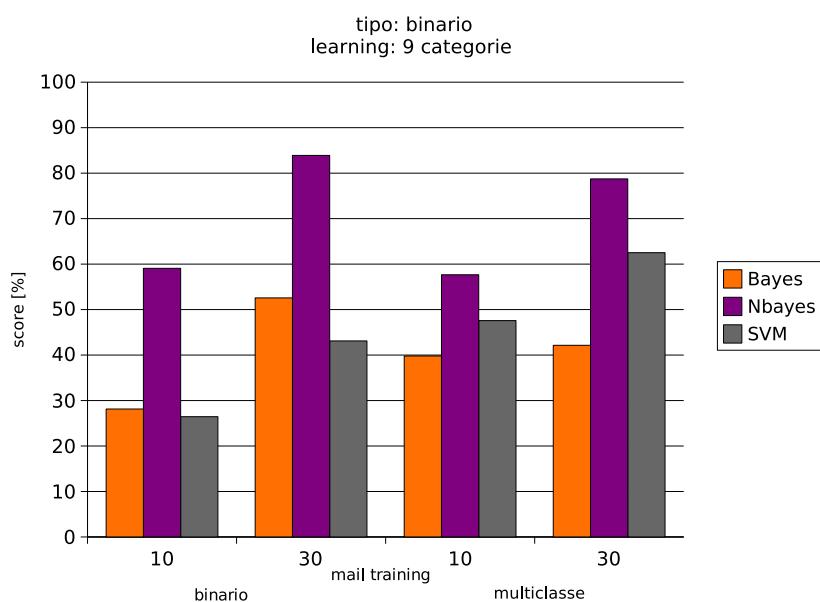


Figura C.1. Istogramma esplicativo della correttezza dei filtri nella clasificazione su tutti i messaggi a disposizione

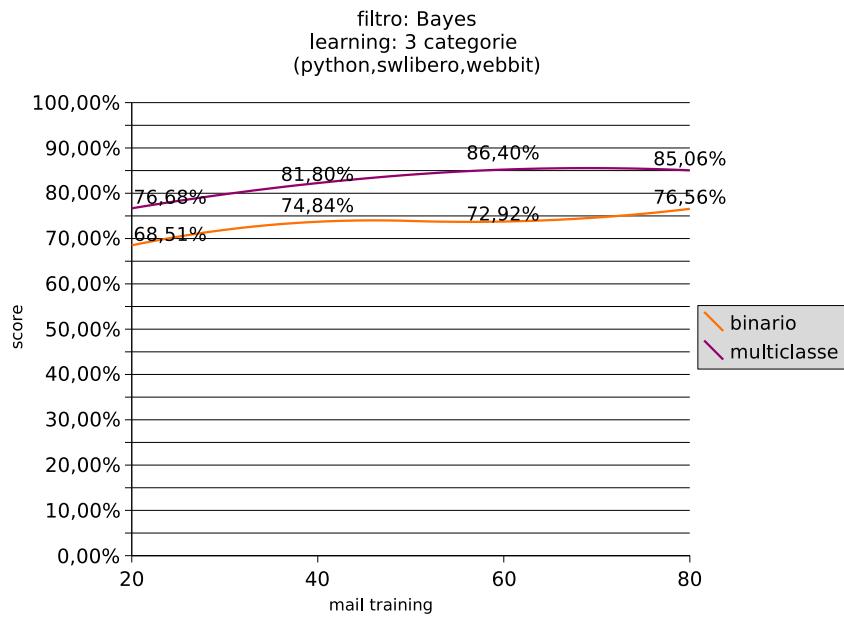


Figura C.2. Andamento della classificazione del filtro Bayes con Python, SoftwareLibero, Webbit

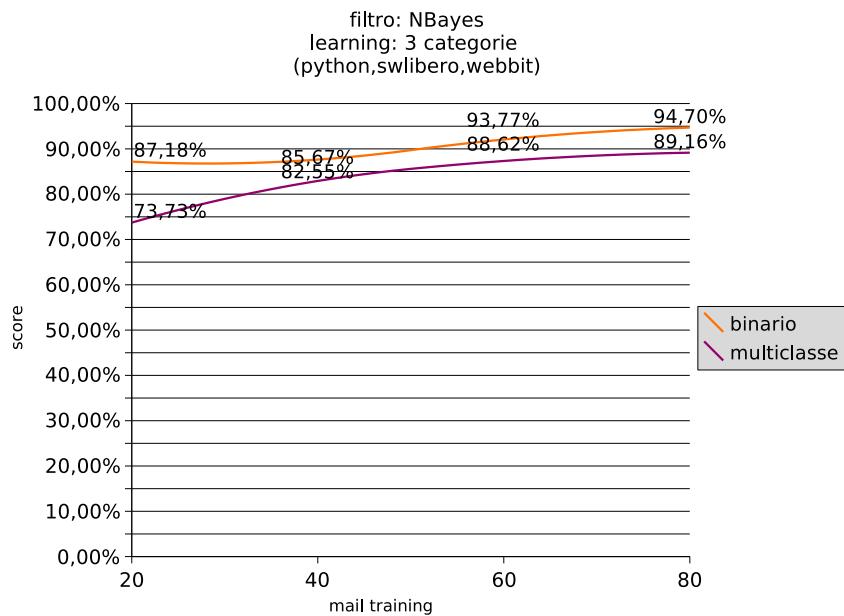


Figura C.3. Andamento della classificazione del filtro NBayes con Python, SoftwareLibero, Webbit

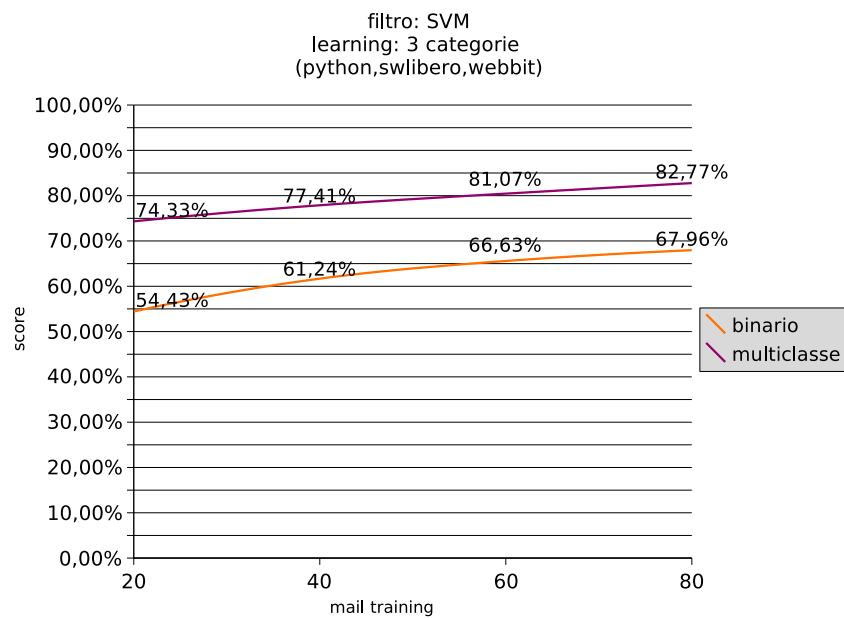


Figura C.4. Andamento della classificazione del filtro SVM con Python, SoftwareLibero, Webbit

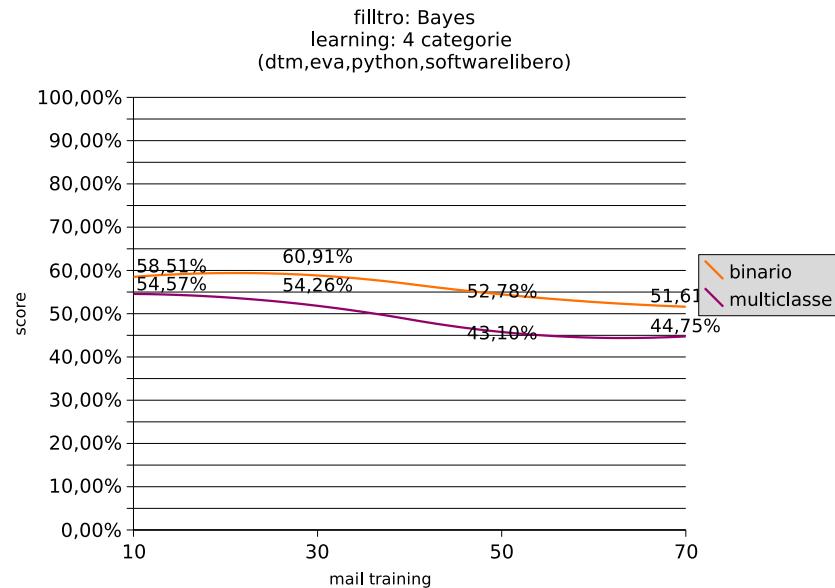


Figura C.5. Andamento della classificazione del filtro Bayes con Dtm, Eva, Python, SoftwareLibero

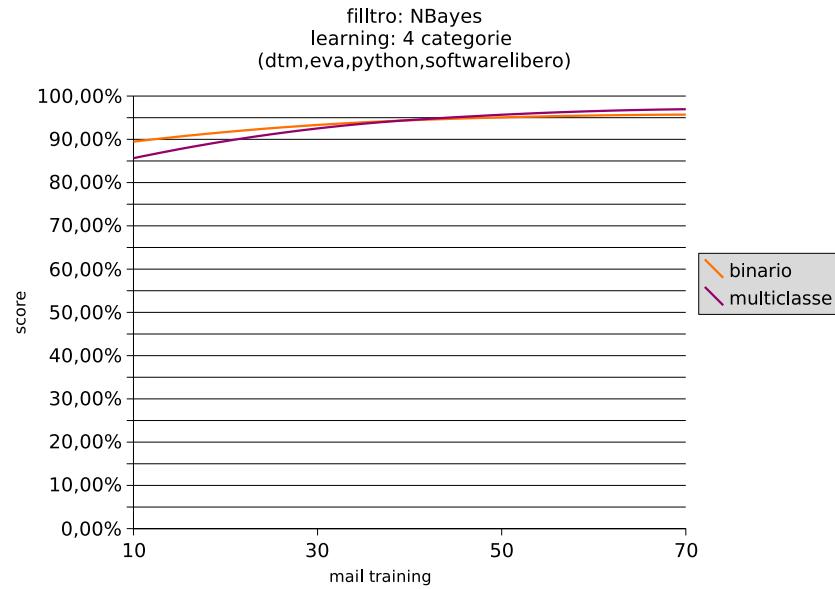


Figura C.6. Andamento della classificazione del filtro NBayes con Dtm, Eva, Python, SoftwareLibero

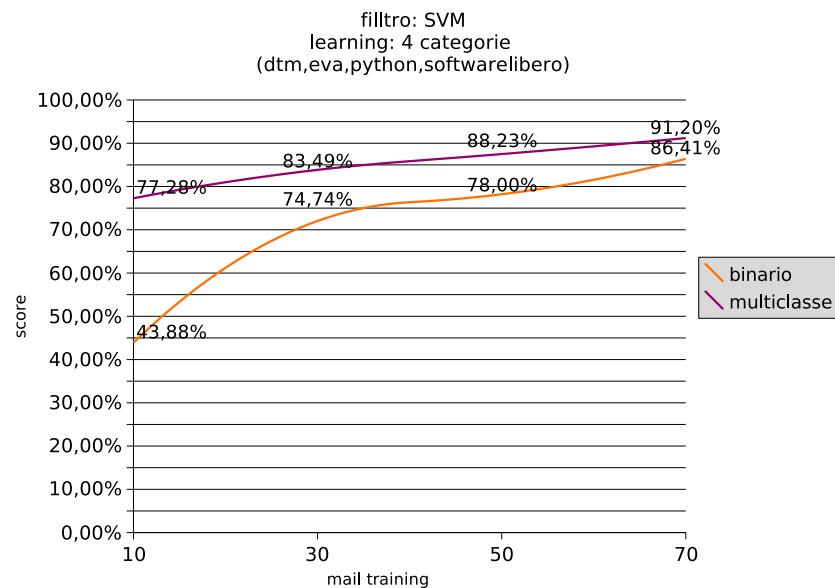


Figura C.7. Andamento della classificazione del filtro SVM con Dtm, Eva, Python, SoftwareLibero

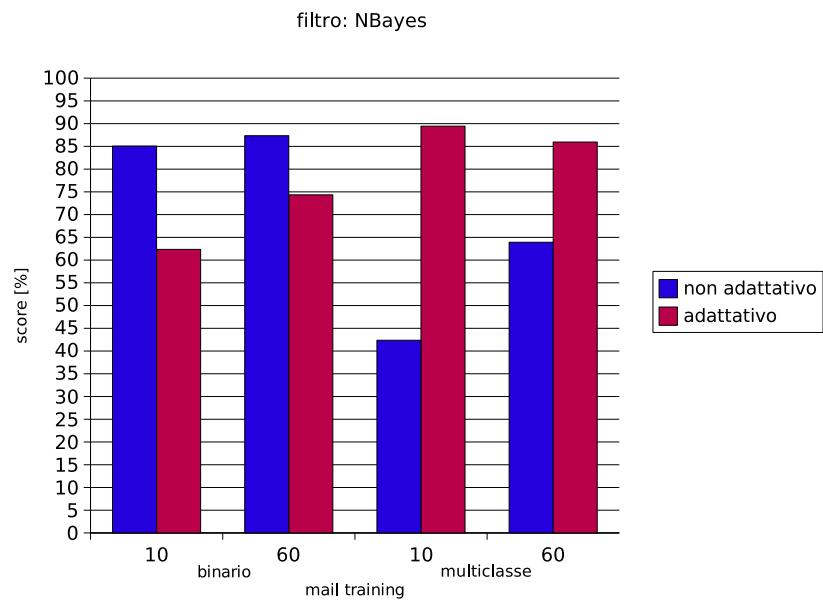


Figura C.8. Andamento della classificazione del filtro NBayes nel test adattativo

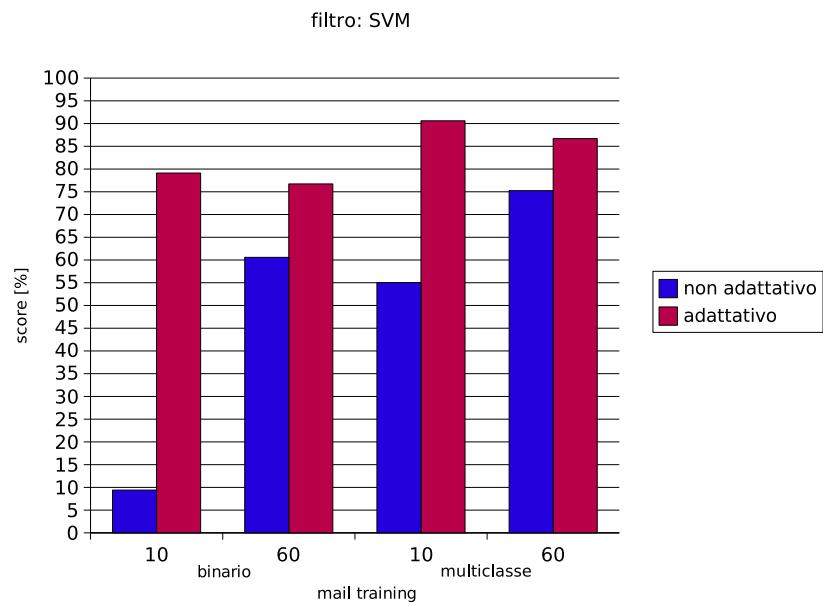


Figura C.9. Andamento della classificazione del filtro SVM nel test adattativo

Appendice D

Schemi relazionali

D.1 Icetricky

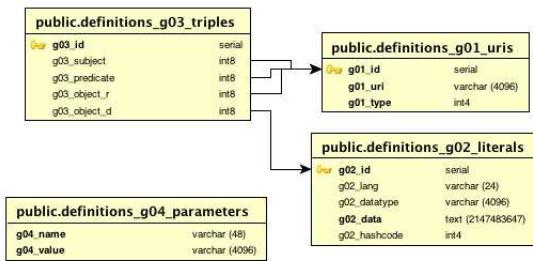


Figura D.1. Schema relazionale delle “definitions”

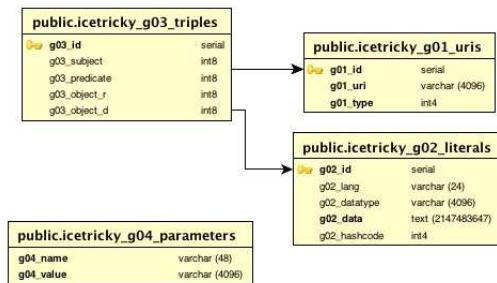


Figura D.2. Schema relazionale delle “triples”

D.2 Icetricky-Configuration

public.authentication	
account	text (2147483647)
passwd	text (2147483647)
enabled	bool
cookie	text (2147483647)

Figura D.3. Schema della tabella di autenticazione

public.smtp	
account	text (2147483647)
password	text (2147483647)
hostname	text (2147483647)
port	int4
secure	bool
userid	text (2147483647)

Figura D.4. Schema della tabella che contiene le informazioni associate agli Smtp Server per i client

public.bow	
id	int8
word	text (2147483647)

Figura D.5. Schema della tabella che contiene un vocabolario indicizzato di parole utili per Svm

public.ngrams	
language	text (2147483647)
ngram	text (2147483647)
occurrency	int4

Figura D.6. Schema della tabella che contiene i dati di training per il riconoscimento della lingua

public.stopwords	
language	text (2147483647)
stop	text (2147483647)

Figura D.7. Schema della tabella che contiene le stop word utili per la cancellazione delle parole “poco utili” per la classificazione del testo

public imaplog	
userid	text (2147483647)
date	text (2147483647)
tracetype	text (2147483647)
msg	text (2147483647)

Figura D.8. Schema della tabella che contiene i log del client imap

public ui log	
userid	text (2147483647)
date	text (2147483647)
tracetype	text (2147483647)
msg	text (2147483647)

Figura D.9. Schema della tabella che contiene i log della user interface

public.category	
account	text (2147483647)
namecategory	text (2147483647)
type	text (2147483647)
catindex	int4
exclusive	bool
totalwords	int4
rf_positive_dictionary	text (2147483647)
rf_negative_dictionary	text (2147483647)
rf_model	text (2147483647)

Figura D.10. Schema della tabella che contiene le categorie di classificazione

public.dictionary	
rif_dictionary	text (2147483647)
messagelid	text (2147483647)
countword	int4
phrase	text (2147483647)
positive	bool

Figura D.11. Schema della tabella che contiene i messaggi di addestramento

public.model	
rf_model	text (2147483647)
classlabel	_int4 (2147483647)
supportvectorcoefficient	_float8 (2147483647)
numberofclasses	int4
numberoftsvperclass	_int4 (2147483647)
pairwiseprobability_a	_float8 (2147483647)
pairwiseprobability_b	_float8 (2147483647)
rho	_float8 (2147483647)
supportvectorcount	int4
rf_parameter	text (2147483647)
rf_node	_text (2147483647)

Figura D.12. Schema della tabella che contiene le informazioni riguardanti la classe Model

public.node	
rif_node	text (2147483647)
index	int4
value	float8 (17,17)

Figura D.13. Schema della tabella che contiene le informazioni riguardanti la classe Node

public.parameter	
rf_parameter	text (2147483647)
c	float8 (17,17)
cachesize	float8 (17,17)
coef0	float8 (17,17)
degree	float8 (17,17)
eps	float8 (17,17)
gamma	float8 (17,17)
kerneltypes	int4
nu	float8 (17,17)
p	float8 (17,17)
probability	bool
shrinking	bool
svmtypes	int4
weightcount	int4
weightlabels	_int4 (2147483647)
weights	_float8 (2147483647)

Figura D.14. Schema della tabella che contiene le informazioni riguardanti la classe Dictionary

Appendice E

Diagrammi delle classi

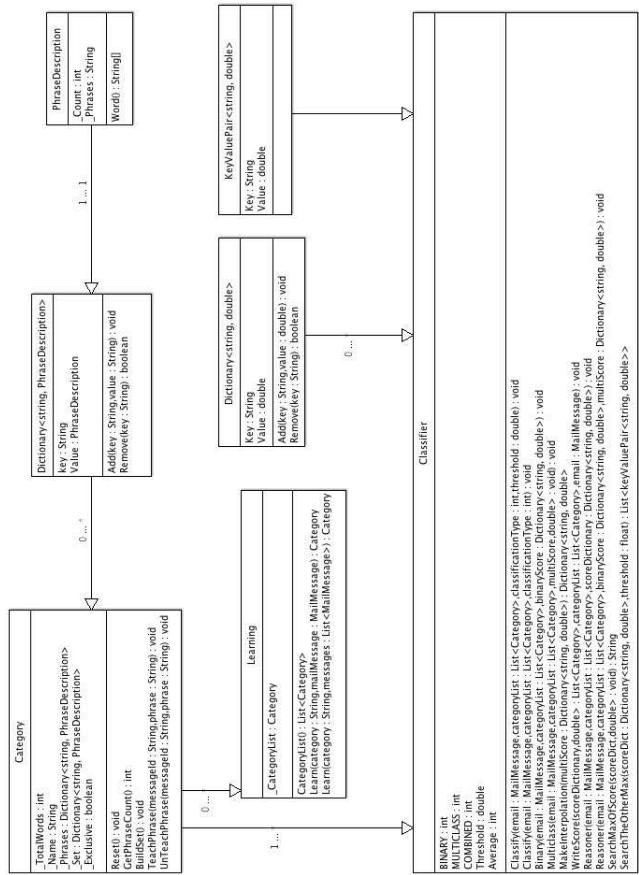


Figura E.1. Diagramma delle classi del progetto Icetricky.Core.Engine.Bayes

E – Diagrammi delle classi

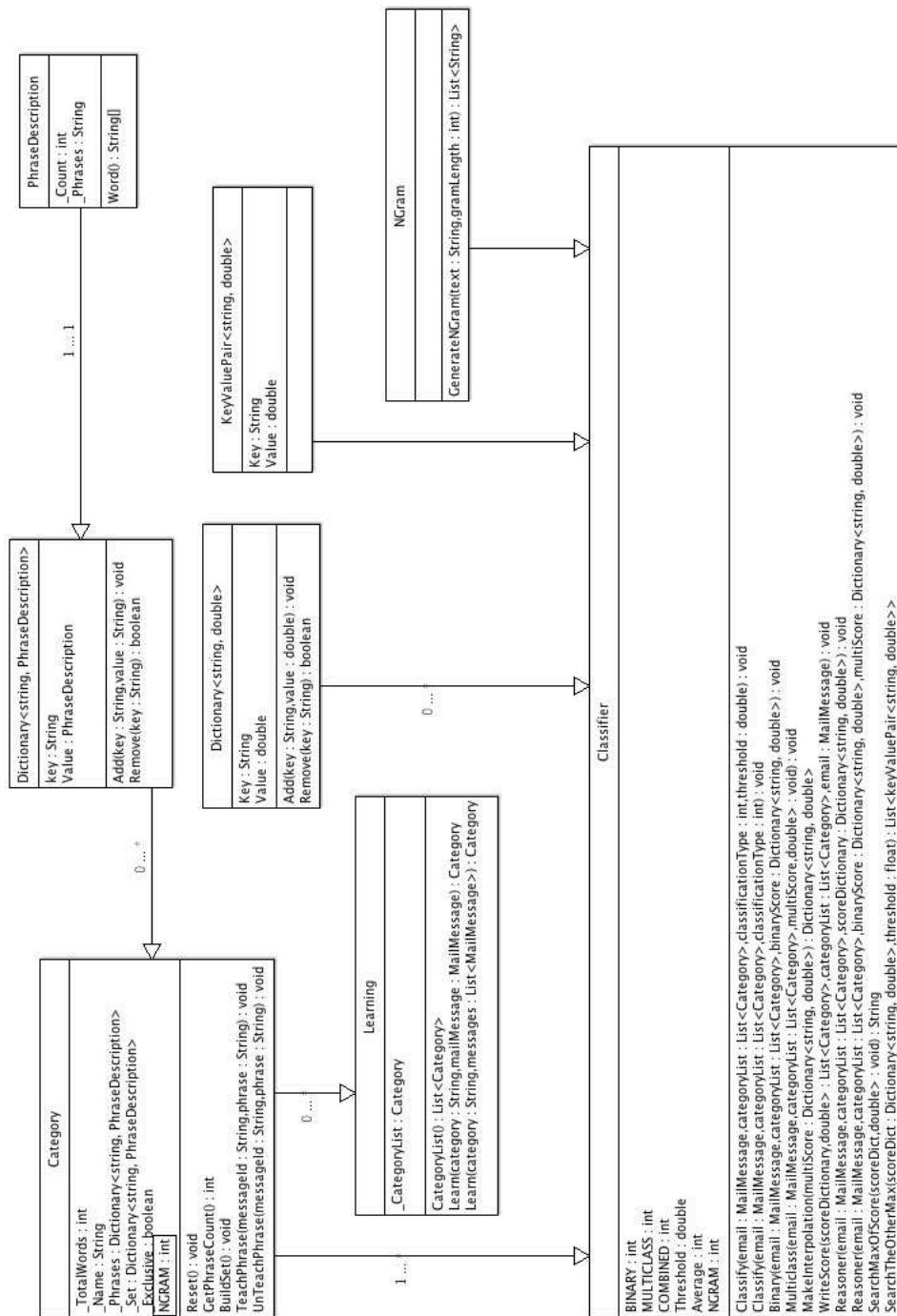


Figura E.2. Diagramma delle classi del progetto Icetricky.Core.Engine.NBayes

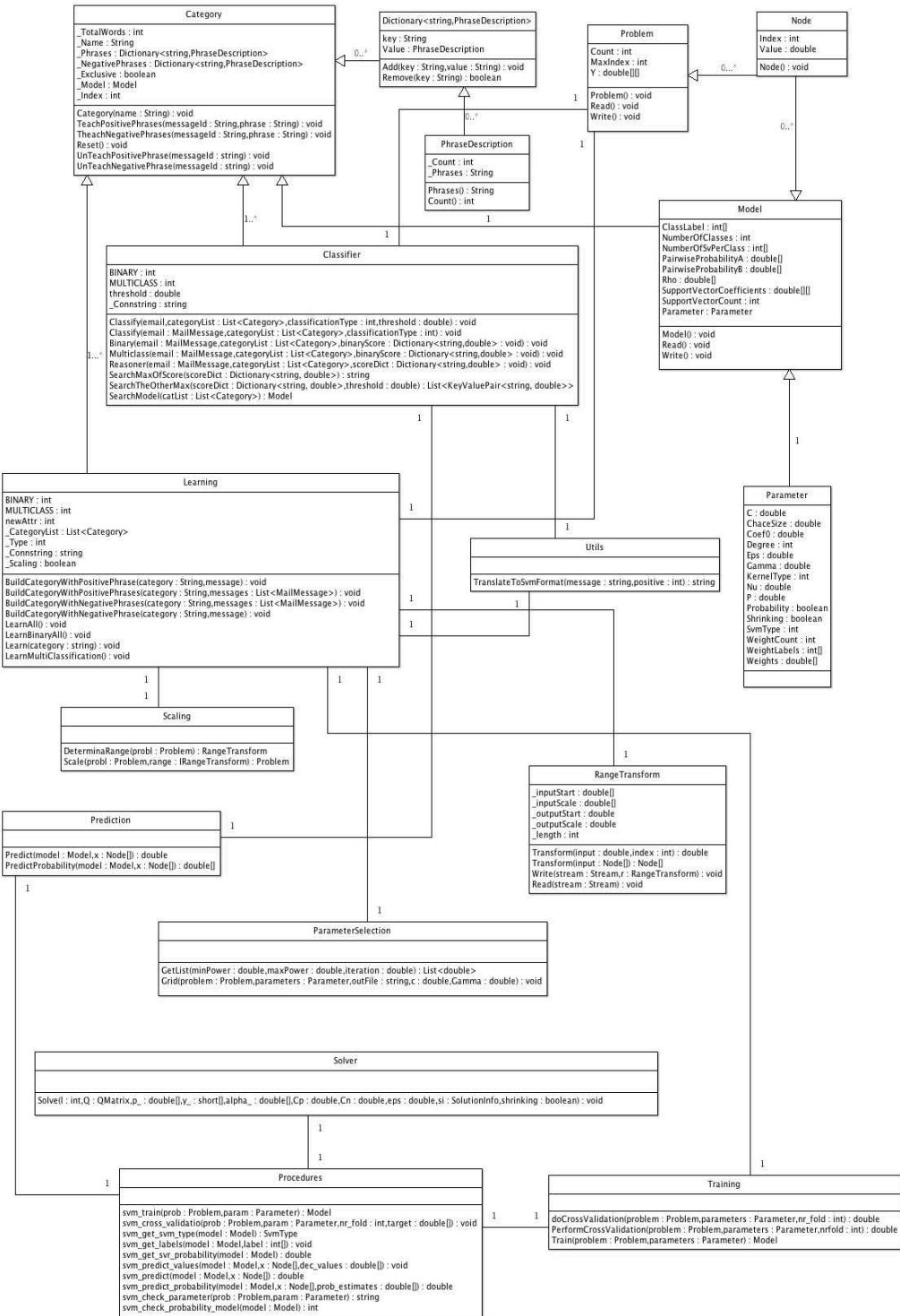


Figura E.3. Diagramma delle classi di Icetricky.Core.Engine.Svm

Appendice F

Schema dell'ontologia

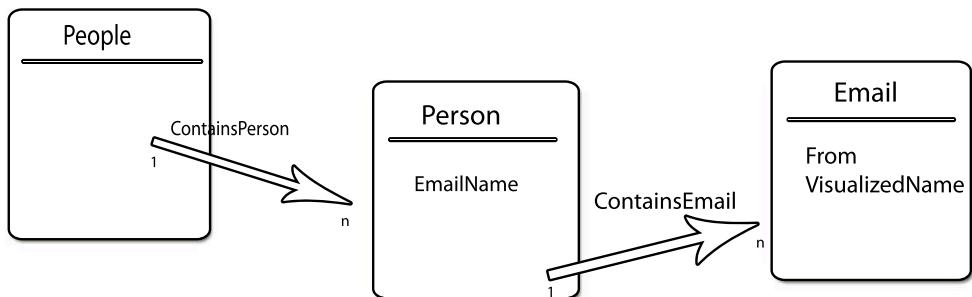


Figura F.1. Diagramma dell'ontologia del progetto PeopleFactory

□

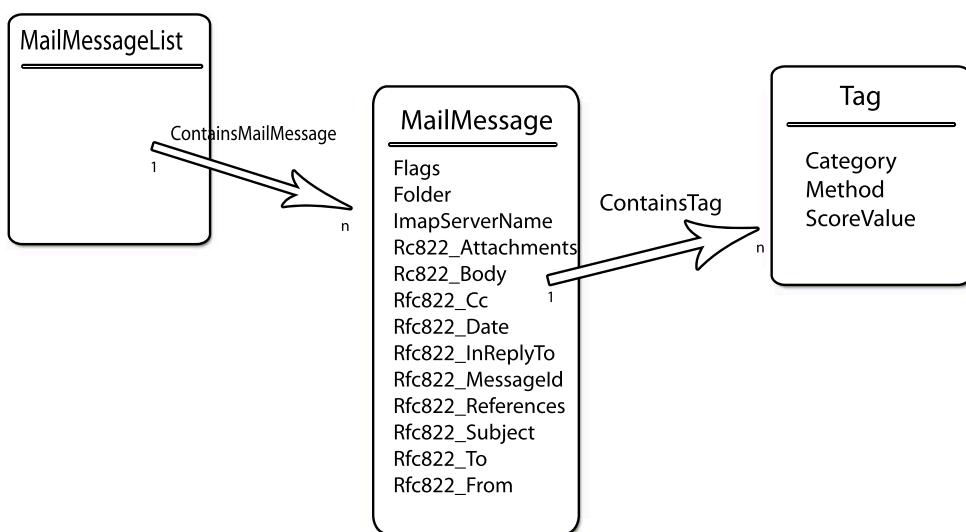


Figura F.2. Diagramma dell’ontologia del progetto MessageFactory

Bibliografia

- [1] Inferenza bayesiana. In http://it.wikipedia.org/wiki/Inferenza_bayesiana.
- [2] *RFC 3501 - Internet Message Access Protocol - Version 4rev1*, 2003.
- [3] Korinna Bade and Andreas Nurnberger. Personalized hierarchical clustering. pages 181–187, 2006.
- [4] Helmut Berger, Michael Dittenbach, and Dieter Merkl. Analyzing the effect of document representation on machine learning approaches in multi-class e-mail filtering. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 297–300, Washington, DC, USA, 2006. IEEE Computer Society.
- [5] Jake D. Brutlang and Christofer Meek. Challenges of the email domain for text classification. pages 1003–110, 2000.
- [6] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001.
- [7] W.W. Cohen. Learning to classify English text with ILP methods. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 124–143. IOS Press, 1996.
- [8] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
- [9] Shuang Deng and Hong Peng. Document classification based on support vector machine using a concept vector model. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 473–476, Washington, DC, USA, 2006. IEEE Computer Society.
- [10] Johan Hovold Department. Naive bayes spam filtering using word position attributes.
- [11] Lipika Dey, Ashish Chandra Rastogi, and Sachin Kumar. Generating concept ontologies through text mining. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 23–32, Washington, DC, USA, 2006. IEEE Computer Society.
- [12] Gamma E., Helm R., Johnson R., and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1995.

- [13] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
- [14] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Haixun Wang, David W. Cheung, and Huan Liu. A balanced ensemble approach to weighting classifiers for text classification. pages 869–873, 2006.
- [15] Graham-Cumming. John graham-cumming. In <http://www.jgc.org/>. Dec 2007.
- [16] Tom Gruber. What is an ontology? In <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>.
- [17] Tom Gruber. What is an ontology? In <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>.
- [18] Vojislav Kecman. *Learning and Soft Computing, Support Vector Machine, Neural Networks and Fuzzy Logic Systems*. The MIT press, Cambridge, 2001.
- [19] Svetlana Kiritchenko and Stan Matwin. Email classification with co-training. page 8, 2001.
- [20] Sauro Menchetti. *Estensione del Classificatore Naive Bayes per la Categorizzazione del Testo con Dati Parzialmente Etichettati*. PhD thesis, Università degli studi di Firenze, 2000.
- [21] T.M. Mitchell. *Machine Learning. Series in Computer Science*. 1997.
- [22] Fuchun Peng and Dale Schuurmans. Combining naive bayes and n-gram language models for text classification, 2003.
- [23] Elio Piccolo. Intelligenza artificiale. In http://ulisse.polito.it/matdid/3ing_inf_N3000_TO_0/lucidi/Lucidi_Corso_2006.
- [24] Stuart J. Russel and Peter Norvig. *Artificial Intelligence. A modern Approach*. Prentice Hall, 1995.
- [25] Paolo Sacconier and Francesco Tamagni. *Classificazione gerarchica per l'indicizzazione semantica del testo*. PhD thesis, Politecnico di Torino, 2006.
- [26] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
- [27] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [28] W3C. Owl web ontology language semantics and abstract syntax. In <http://www.w3.org/TR/owl-semantics/>.
- [29] W3C. Owl web ontology language use cases and requirements. In <http://www.w3.org/TR/webont-req/>.
- [30] W3C. Rdf prime. In <http://www.w3.org/TR/REC-rdf-syntax/>.
- [31] Wikipedia. Hidden markov model. In http://en.wikipedia.org/wiki/Hidden_Markov_model.
- [32] Wikipedia. Web service. In <http://en.wikipedia.org/wiki/Webservice>. Dec 2007.