Studio di tecniche di Apprendimento Automatico per l'analisi e la classificazione di messaggi di posta elettronica.

Martina Astegno

14 aprile 2011

Indice

Elenco delle figure

Elenco delle tabelle

Parte I

Introduzione e basi teoriche

Capitolo 1

Introduzione

I messaggi di posta elettronica sono diventati un mezzo veloce, economico e facilmente accessibile per la comunicazione di informazioni. La crescita esplosiva del traffico di e-mail ha quindi reso indispensabile l'adozione di meccanismi in grado di classificare e di organizzare i messaggi in ingresso sulla base di specifici criteri. Questo riguarda soprattutto le aziende organizzate in vari dipartimenti, differenti per tipologia di attività e servizio offerto. L'idea di base è quella di riuscire ad analizzare in modo automatico il traffico di e-mail in ingresso, e, sulla base del loro contenuto incanalare i messaggi verso il settore di riferimento per l'argomento trattato. Così facendo il singolo operatore aziendale dovrà analizzare solamente le e-mail di sua competenza, senza dover eseguire una selezione preventiva su tutti i messaggi in ingresso.

Presso l'Università di Padova era stato realizzato un sistema di classificazione automatica integrato all'interno di un'azienda operante nel settore dell'ICT. Una delle aree principali di *business* dell'azienda è l'erogazione di servizi di *call center* integrati (servizi informativi clienti, servizi di CRM, servizi di *help desk*, servizi avanzati di *booking* per eventi culturali). Il sistema smistava le e-mail in arrivo presso una particolare casella di posta elettronica all'operatore più adatto per la lettura ed eventuale risposta, interfacciandosi con il sistema di *ticketing* OTRS utilizzato nel *call center* aziendale.

Questo sistema è stato utilizzato come base di partenza per il progetto di tesi. Nel corso del progetto, il sistema di base è stato esteso ed adattato alle esigenze di un nuovo utente, ossia un'azienda operante nel settore dell'editoria. L'azienda si occupa della gestione e vendita di una rivista a diffusione mondiale; offre inoltre diversi servizi, tra cui la vendita di libri e di altri prodotti, e svolge attività di pubbliche relazioni, che le impongono la gestione di un elevato numero di e-mail al giorno.

Il nuovo dominio applicativo ha richiesto un'intera revisione del sistema di classificazione preesistente per poterlo adattare ai nuovi requisiti

aziendali.

Prima di poter entrare nel dettaglio della procedura di classificazione, è stato necessario modificare il protocollo di comunicazione tra il sistema di classificazione e il sistema aziendale. É stata introdotta una nuova modalità di gestione delle e-mail, che prevede l'adozione di un particolare identificativo univoco, associato dall'azienda ad ogni messaggio in ingresso. Questa modifica ha coinvolto tutte le componenti del sistema, e le modalità di pubblicazione dei risultati di classificazione, nonchè la lettura del *feedback* fornito dagli operatori aziendali.

L'obiettivo dell'attività di tesi è la realizzazione di un sistema di classificazione automatica capace di organizzare il traffico di e-mail in ingresso. Le categorie in cui smistare i messaggi di posta elettronica corrispondono alle diverse attività e servizi offerti dall'azienda coinvolta. Il classificatore, per ogni nuova e-mail, fornisce una lista delle classi disponibili, inserendo in prima posizione la categoria con il valore più alto di probabilità stimato. L'azienda ha fissato un valore soglia minimo, relativo alla percentuale di classificazioni corrette, che il modello deve collocare in prima posizione nel *ranking* finale; questo valore è pari all'80% del totale delle e-mail classificate.

Per raggiungere queste prestazioni è stato necessario procedere ad uno studio e a una successiva revisione delle singole fasi di classificazione. È bene precisare che tutte le scelte e modifiche apportate derivano da un'analisi accurata dei risultati dei test che sono stati eseguiti nel corso della revisione del sistema.

La classificazione delle e-mail si basa sull'adozione di una particolare tecnica di *Machine Learning*, ossia la *Text Categorization*: dato un possibile insieme di categorie/argomenti ed una collezione di documenti, lo scopo è riuscire a individuare il *topic* corretto per ogni documento. In un primo momento potrebbe risultare un compito semplice, ma in realtà analizzando meglio le problematiche e le difficoltà derivanti anche dalla particolare natura del testo preso in esame, ci si rende conto che è necessario adottare un approccio per gradi e analizzare tutti i fattori coinvolti, per evitare di trascurare dettagli che potrebbero condurre a risultati inattesi.

Innanzitutto è indispensabile acquisire una certa conoscenza sugli aspetti e sulle peculiarità che caratterizzano i dati da classificare. Da uno studio dettagliato sugli esempi supervisionati disponibili sono emerse alcune particolarità nel testo delle e-mail che hanno richiesto una revisione della fase di *pre-processing* e di indicizzazione; questa operazione ha condotto ad un raffinamento della rappresentazione degli esempi, utilizzata nell'apprendimento del modello di classificazione.

Per ciò che concerne la fase di classificazione vera e propria invece, il sistema preesistente si basava su uno dei modelli più semplici disponibili, quello bayesiano. Durante l'attività di tesi, dopo aver perfezionato la rappresentazione degli esempi, si è deciso di valutare la possibilità di integrare

o di sostituire il modello precedentemente utilizzato, con altri classificatori che garantiscono maggiore accuratezza nella classificazione.

L'adozione di un nuovo modello ha apportato un sensibile miglioramento alle prestazioni del classificatore, comportando però un maggior sforzo per il *tuning* dei nuovi parametri richiesti.

In un primo momento il sistema di classificazione modificato è stato sperimentato in un ambiente di test. Questa fase di test ha permesso di valutare le prestazioni del classificatore al variare di determinate condizioni e parametri. Solo dopo un'accurata analisi dei risultati ottenuti, il nuovo sistema è stato messo in produzione nel sistema aziendale.

All'interno del presente documento posso essere distinte 3 parti principali descritte brevemente di seguito.

Parte 1 : introduzione al nuovo dominio applicativo e illustrazione dei fondamenti teorici su cui si basa l'attività di tesi.

Parte 2 : breve descrizione del sistema di classificazione preesistente, seguita da una analisi accurata del sistema proposto, e illustrazione dei risultati dei test eseguiti per verificare la correttezza delle scelte effettuate.

Parte 3 : considerazioni finali, soddisfacimento dei requisiti aziendali e analisi dei possibili sviluppi futuri.

Capitolo 2

Text Categorization

Uno dei temi più comuni del *Text Mining* è la *Text Categorization*: dato un insieme di categorie/argomenti ed una collezione di documenti, si richiede di trovare il *topic* corretto per ogni documento. Lo studio della *Automated Text Categorization* inizia negli anni '60, ma è solo negli anni '90 che si assiste ad un sostanziale sviluppo, dovuto anche al crescente numero di documenti in forma digitale. Oggi la *Automatic Text Categorization* ha svariati utilizzi: *text indexing*, *document sorting, spam filtering,* categorizzazione delle pagine web in cataloghi gerarchici e molti altri.

Esistono due approcci alla *Text Categorization*:

- *Knowledge Engineering approach*: si focalizza sulla creazione manuale di regole per la classificazione, che devono essere definite da un esperto del dominio. Anche se le performance raggiungibili da questo tipo di sistemi sono più alte di quelle ottenibili dall'approccio di tipo *Machine Learning*, non viene utilizzato poichè potrebbero essere necessari anni per definire un insieme di regole soddisfacente. Questo problema rende l'altro approccio molto più utilizzato.

- *Machine Learning approach*: tramite un processo induttivo viene generato un classificatore da un insieme di esempi pre-classificati.

L'approccio trattato nel presente documento è quello basato sull'apprendimento automatico.

2.1 Definizione di *Text Categorization*(TC)

In generale, la *Text Categorization* ha come obiettivo quello di approssimare una funzione, non nota, di assegnamento a classi $F : D \times C \rightarrow \{0, 1\}$, dove D è l'insieme dei documenti, mentre C è l'insieme delle categorie.
F è definita come:

$$F(d, c) = \begin{cases} 1 & \text{se il documento } d \text{ appartiene alla categoria } c \\ 0 & \text{altrimenti} \end{cases}$$

La funzione di approssimazione $M : D \times C \rightarrow \{0, 1\}$ è chiamata classificatore. Il problema è generare un classificatore che produca risultati il più possibile vicini alla reale funzione di classificazione F.

2.2 Tipologie di classificazione

È possibile individuare una serie di caratteristiche che determinano differenti tipologie di classificazione. Le principali distinzioni sono definite da:

- il numero di label associate al documento;
- la base per l'ordinamento dei risultati forniti dal classificatore (categorie o documenti);
- il tipo di decisione presa dal classificatore (*hard* o *soft*).

Single/Multi Label Categorization

A seconda delle proprietà di F possiamo distinguere due diverse modalità di classificazione.

- *Multi-Label*: le categorie si sovrappongono, quindi un documento può appartenere a più categorie.
- *Single-Label*: ogni documento appartiene esattamente ad una categoria.

La classificazione binaria è un caso speciale della classificazione *single-label*, dove il numero di categorie è due. Si tratta della classificazione più utilizzata. Dal punto di vista teorico il caso binario generalizza la classificazione *multi-label*, e di conseguenza quest'ultima può essere realizzata ricorrendo a $|C|$ classificatori binari.

Document/Category pivoted

Solitamente, i classificatori sono utilizzati nel seguente modo: dato un documento, il classificatore deve trovare tutte le categorie a cui quel documento appartiene. Questa è chiamata categorizzazione *document-pivoted*. Dualmente, potremmo aver bisogno di trovare tutti i documenti che appartengono ad una categoria fissata (*category-pivoted*). La differenza tra le due è piuttosto rilevante, specialmente nel caso in cui la categorizzazione è *online*, quando cioè, non sono sempre disponibili tutti i documenti o tutte le categorie in un determinato istante.

Hard/Soft categorization

Un sistema di categorizzazione completamente automatizzato deve prendere una decisione binaria per ogni coppia documento-categoria. Questi sistemi eseguono una *hard categorization*. Le performance raggiunte da questi sistemi possono non essere completamente soddisfacenti. In questi casi si adotta un approccio semi-automatizzato in cui il sistema presenta una lista di categorie ordinata secondo la stima di appartenenza e un umano seleziona la categoria da assegnare *(Soft Categorization)*. Molti classificatori producono per ogni coppia documento-categoria un valore reale compreso tra 0 e 1, chiamato *Categorization Status Value*, mettendo a disposizione quindi un *ranking*. È possibile prendere decisioni binarie introducendo un valore soglia, che può essere calcolato automaticamente (possibilmente massimizzando le performance del classificatore su un *Validation Set*).

2.3 Applicazioni della *Text Categorization*

La *Text Categorization* può essere usata in diverse applicazioni. Di seguito vengono proposte e descritte le principali.

Indicizzazione automatica per sistemi di *Boolean Information Retrieval*

L'obiettivo dell'indicizzazione automatica per sistemi di *Boolean IR* è di associare ad ogni documento delle *keywords* o delle *keyphrases* che ne descrivano il contenuto, selezionandole da un insieme finito, chiamato dizionario controllato (e.g. thesaurus gerarchici tematici). Di solito l'assegnamento viene svolto da un esperto umano, e chiaramente questo comporta un costo notevole. Considerando invece come categorie i termini presenti nel dizionario controllato, l'indicizzazione può essere vista come una istanziazione della TC e quindi risolvibile adottando delle tecniche automatiche. Questa applicazione è strettamente correlata con un altra attività, la generazione automatica di metadati. Nelle librerie digitali risulta utile riuscire ad individuare una serie di metadati tematici che descrivano il contenuto dei vari testi disponibili. Questo scopo può essere raggiunto ricorrendo alle tecniche di TC, dato che questa procedura può essere ricondotta al problema dell'indicizzazione automatica di testi con dizionario controllato.

Organizzazione di documenti

L'organizzazione di documenti costituisce un problema piuttosto generale che può essere istanziato in base alle esigenze. L'obiettivo primario è quello di organizzare e di strutturare un insieme di documenti in base a determinati

parametri e può essere raggiunto sfruttando le tecniche di TC. Alcuni esempi sono:

- organizzazione di brevetti in categorie;
- archiviazione automatica di articoli in sezioni;
- raggruppamento di articoli scientifici in sessioni.

Text filtering

Il *Text filtering* è l'attività di classificare una collezione dinamica di testi, ad esempio lo stream di documenti, distribuiti in modo asincrono, da un produttore di informazioni a un consumatore. Il flusso di *news* è il caso tipico, in cui il produttore è l'agenzia e il consumatore è il giornale. In questo contesto il sistema di *filtering*, posizionato lato produttore, deve bloccare l'invio delle *news* per cui il consumatore non ha interesse. Si tratta perciò di un caso di *single-label categorization*, in cui i documenti (*news*) sono suddivisi in due categorie, i rilevanti e i non rilevanti. Inizialmente deve essere creato un profilo per ogni consumatore. Questa operazione viene eseguita sfruttando il *feedback* fornito dall'utente a cui vengono sottoposte delle *query* per stabilire la rilevanza o meno di documenti specifici.

Classificazione gerarchica di pagine web

Una possibile applicazione al mondo di Internet della TC è la classificazione automatica di pagine web o siti in una o più categorie. Così facendo, l'utente non deve più eseguire una *query* attraverso un motore di ricerca web, ma può analizzare la gerarchia di categorie individuate e concentrare la propria ricerca su quella di interesse. Questa procedura ha due peculiarità.

- La natura ipertestuale dei documenti: gli *hyperlink* costituiscono un ricca fonte di informazioni, essi infatti attribuiscono rilevanza alla pagina a cui punta il collegamento.
- La struttura gerarchica dell'insieme delle categorie: potrebbe essere sfruttata per scomporre il problema di classificazione in tanti sotto-problemi, che corrispondono ai rami di scelta dei nodi interni della gerarchia.

2.4 Approccio *Machine Learning*

Nell'approccio ML, il classificatore è costruito automaticamente a partire da un insieme di *training* costituito da documenti pre-classificati. Si tratta di un istanza di apprendimento *supervisionato* poiché si conosce il vero valore

della funzione di assegnamento nel *Training Set.* La versione non supervisionata della classificazione è chiamata *clustering.* Esistono vari approcci per apprendere il classificatore, alcuni dei quali sono varianti di algoritmi più generali di ML, mentre altri sono stati creati specificatamente per la categorizzazione.

2.4.1 Alcune definizioni fondamentali

L'approccio *Machine Learning* si basa sulla esistenza di un corpus iniziale $\Omega = \{d_1, d_2, \ldots, d_{|Q|}\}$ di documenti classificati sulla base delle categorie presenti nell'insieme $C = \{c_1, c_2, \ldots, c_{|C|}\}$, su cui il sistema deve operare. Questo significa che i valori della funzione totale $\Phi : D \times C \rightarrow \{T, F\}$ sono noti per ogni coppia $(d_j, c_i) \in \Omega \times C$, dove $\Omega \subset D$. Un documento d_j è considerato un esempio positivo per la categoria c_i se il valore della funzione $\Phi(d_j, c_i) = T$, un esempio negativo se $\Phi(d_j, c_i) = F$. Vengono fornite di seguito alcune definizioni fondamentali:

- **Training Set (TR)**: insieme di esempi pre-classificati che costituiscono il corpus iniziale, su cui viene eseguito l'apprendimento del classificatore.

- **Validation Set (TV)**: insieme di esempi che viene usato per la messa a punto dei parametri di un classificatore.

- **Test Set (Te)**: $Te = \{d_{|TV|+1}, \ldots, d_\Omega\}$, viene usato per valutare l'efficacia dei classificatori. La categoria proposta dal modello, per ogni documento appartenente a Te, viene confrontata con la decisione dell'esperto. L'efficacia del classificatore si basa sul numero di errori di classificazione che vengono commessi. È importante ricordare che gli esempi presenti nel Te non devono in alcun modo essere usati per l'apprendimento del modello iniziale. Dopo aver effettuato la scelta del classificatore, è possibile rieseguire il *training* su tutto il TR, migliorandone così l'efficacia (approccio *train-and-test*). Un approccio alternativo è il *k-fold cross validation*, in cui vengono appresi k classificatori $\Phi_1, \ldots, \Phi_k$ suddividendo il corpus iniziale in k sottoinsiemi disgiunti $Te_1, \ldots, Te_k$, e applicando l'approccio *train-and-test* per ogni coppia $< TV_i = \Omega - Te_i, Te_i >$. La valutazione dell'efficacia finale viene eseguita analizzando le prestazioni dei singoli k classificatori, e calcolando poi la media dei risultati ottenuti.

2.4.2 Indicizzazione dei documenti

Gli algoritmi di apprendimento e classificazione non possono lavorare direttamente sul testo nella sua forma originale. Per questo è necessario effettuare un passaggio di *pre-processing*, in cui i documenti grezzi sono convertiti in

una forma più maneggevole. Per assolvere questo compito vengono messe a disposizione una serie di tecniche, tra cui le più note sono le seguenti:

- *Tokenization*
  Rappresenta uno dei primi passi di analisi del linguaggio. Consiste nel suddividere il testo in *token*, ognuno dei quali corrisponde ad un'istanza di un tipo; di conseguenza il numero di *token* sarà tanto più alto quanto maggiore è il numero dei tipi presenti. Per quanto riguarda il trattamento della punteggiatura, l'approccio dipende dalla necessità di mantenere o meno l'informazione relativa. In alcuni contesti si può decidere di considerare solo i confini di frase, trascurando la punteggiatura interna. Queste scelte dipendono chiaramente dall'ambiente in cui si opera.

- *Stemming*
  Dopo aver suddiviso il testo in *token* è necessario effettuare la conversione in una forma standard. Lo *stemming* rappresenta una possibile forma di standardizzazione che estrae la radice di una parola, eliminando affissi e desinenze. Questa operazione si limita però a troncare la parola mediante un'analisi morfologica semplificata che non sempre è corretta. L'effetto consiste nel ridurre il numero di tipi distinti presenti nel testo, e nell'aumentare la frequenza dell'occorrenza di alcuni tipi particolari.

- *Lemmatization*
  Costituisce un'altra forma di standardizzazione del testo. È un processo che cerca il lemma a partire da una parola con desinenza. Rispetto alla precedente tecnica, questa si occupa anche di disambiguare tra le diverse forme base a cui può corrispondere una forma flessa.

- *Finding Collocations*
  Una *collocation* è una espressione dotata di significato, formata da due o più parole, che corrisponde a un qualche uso idiomatico. È importante sottolineare che il significato delle *collocation* non può essere dedotto dal significato delle singole parti.

- *N-grams*
  Un n-gram è una sequenza generica di n parole. Si distinguono dalle *collocations* in quanto non corrispondono necessariamente ad un uso idiomatico (non sono per forza portatori di informazione semantica); molto spesso gli n-grams rappresentano costruzioni sintattiche comuni. Sono utili nel caso in cui i testi trattati siano in lingue non note al sistema, o in ricerche che coinvolgano problemi di *authorship analysis*.

- *Word Sense Disambiguation*
  Questa operazione consiste nel determinare quale dei significati di una

parola ambigua si riferisce ad una occorrenza specifica. Possono essere individuati due diversi tipi di ambiguità:

  - Polisemia: i diversi significati attribuiti alla stessa parola sono legati etimologicamente e semanticamente.
  - Omonimia: i diversi significati di un lessema si trovano a essere rappresentati da un'unica forma ortografica solo per caso.

  Esistono due classi di risoluzione del problema, una definita *dictionary-based* che ricorre a risorse lessicali, e una basata su metodi di *learning*.

- *Anaphora Resolution*
  L'anafora è una relazione tra una espressione linguistica, detta appunto anafora, e un'altra che la precede e ne determina il riferimento. Il metodo in esame ha il compito di identificare e risolvere le relazioni di anafora fra le entità all'interno di un testo; questo procedimento risulta comunque essere particolarmente difficoltoso, ed infatti le migliori tecniche a disposizione possono garantire una precisione massima del 50-60%.

- *Part of Speech Tagging*
  Questa tecnica consiste nell'attribuire ad ogni parola del testo un *tag* con la sua corretta funzione nel discorso, determinando quindi se si tratta di un articolo, nome, verbo, aggettivo e così via. Il *tagging* può essere visto anche come un'applicazione limitata di disambiguazione sintattica: nel caso in cui una parola appartenga a più categorie sintattiche, determina quale tra queste categorie è la più plausibile nel contesto dato.

A questo punto i documenti vengono trasformati in *feature vectors*. Gli approcci più comuni utilizzano come *feature* semplicemente tutte le parole nei documenti. Sono presenti vari metodi per assegnare pesi alle *feature*. Il più comune è lo schema TF-IDF, che data un termine t_k nel documento d_j calcola:

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot log \frac{|Tr|}{\#_{Tr}(t_k)}$$

in cui $\#(t_k, d_j)$ indica il numero di occorrenze del termine t_k nel documento d_j e $\#_{Tr}(t_k)$ rappresenta la frequenza del termine nei documenti del Tr (il numero di documenti in cui il termie t_k compare almeno una volta). È importante notare che in questa formula si considerano solo le occorrenze dei termini, trascurando quindi l'ordine con cui compaiono nel documento e il loro ruolo sintattico. Di solito, i pesi associati ai termini sono ricondotti nell'intervallo [0,1], e se i documenti sono rappresentati da vettori di uguale lunghezza è possibile normalizzare il valore di $tfidf$ attraverso la *cosine*

normalization. Il peso normalizzato viene quindi calcolato come segue:

$$w(t_k, d_j) = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|Tr|}(tdidf(t_s, d_j))^2}}$$

Nonostante $TF - IDF$ sia di gran lunga lo schema di associazione di pesi più noto, vengono utilizzate anche altre tecniche, soprattutto nei casi in cui il TR non è disponibile nella sua interezza, e quindi non è possibile calcolare la frequenza dei termini in tutti i documenti.

2.4.3 Riduzione dello spazio delle *feature*

Molte delle parole presenti nei documenti sono irrilevanti ai fini della categorizzazione e possono essere scartate senza influenzare le performance del classificatore. Inoltre nella maggior parte dei casi avere uno spazio di termini troppo ampio può sollevare alcune problematiche, riducendo infatti lo spazio delle *feature* si contribuisce a ridurre il cosidetto fenomeno dell'*overfitting*.

Feature selection

Nel caso in cui il nuovo spazio delle *feature* è un sottoinsieme di quello iniziale, si parla di *feature selection*. Una tecnica molto semplice consiste nell'usare liste di *stop words*, cioè elenchi precompilati di parole irrilevanti (articoli, preposizioni, congiunzioni, ...). Altre tecniche più avanzate analizzano la rilevanza di ogni *feature*. È possibile calcolare la frequenza di un termine t_k nei documenti $DocFreq(t_k)$. Esperimenti dimostrano che considerando solo il primo 10% delle parole più frequenti non si riducono le performance del classificatore. Esistono anche altre tecniche più complesse, ad esempio l'*Information Gain*:

$$IG(t_k) = \sum_{c \in C \cup \bar{C}} \sum_{f \in \{t_k, \bar{t_k}\}} P(f, c) \cdot log \frac{P(c|f)}{P(c)}$$

Le probabilità sono stimate sulla base della frequenza nel *Training Set*. Un'altra buona misura è il *chi-squared*:

$$\chi^2_{max} = \max_{c \in C} \frac{|Tr| \cdot (P(f, c) \cdot P(\bar{f}, \bar{c}) - P(f, \bar{c}) \cdot P(\bar{f}, c))^2}{P(f) \cdot P(\bar{f}) \cdot P(c) \cdot P(\bar{c})}$$

Esperimenti dimostrano che entrambe queste misure possono ridurre la dimensionalità del *feature space* di un fattore 100 senza perdita di qualità nella categorizzazione.

Feature extraction

Un altro modo per ridurre lo spazio delle *feature* è quello di creare un nuovo insieme di *feature*, non necessariamente con termini presenti nei documenti. Corrisponde a creare una trasformazione, dallo spazio delle *feature* originale ad un altro spazio a molte meno dimensioni. Questo metodo combatte i problemi legati alla polisemia, omonimia e sinonimia. Una tecnica è quella del *term clustering*, cioè unire in una sola *feature* vari termini fortemente correlati dal punto di vista semantico. Questi gruppi di parole sono poi utilizzati come *feature*, anzichè considerare i singoli termini. Un altro approccio più complesso è quello del *Latent Semantic Indexing*.

2.5 Classificatori a confronto

Classificatori probabilistici

I classificatori probabilistici vedono la *Categorization Status Value* $CSV(d, c)$ come la probabilità $P(c|d)$ che il documento d appartenga alla categoria c, e calcolano questa probabilità applicando il teorema di Bayes:(rif articolo)

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

La probabilità a priori $P(d)$ non deve essere calcolata perchè è costante per tutte le categorie. Per stimare $P(c|d)$ è necessario fare delle assunzioni riguardo la struttura del documento d. Assumiamo che i documenti siano rappresentati come vettori di *feature* $d = (w1, w2, \ldots)$. L'assunzione più comune è quella di indipendenza dei termini dei classificatori *Naïve Bayes*:

$$P(d|c) = \prod_i P(w_i|c)$$

Questi classificatori vengono chiamati *Naïve* poichè questa assunzione è palesemente falsa; si comportano tuttavia abbastanza bene, e modelli più complessi che tentano di rilassare questa assunzione non hanno prodotto consistenti miglioramenti nelle performance.

Classificatore ottimo di Naïve Bayes

Questo classificatore rappresenta una delle tecniche più semplici e popolari e può essere utilizzati per insiemi di dati di dimensione piuttosto grande. Riassumendo quanto già esposto precedentemente, la funzione target da apprendere è:

$$f : D \rightarrow C$$

con istanze d definite dagli attributi $\langle w_1, w_2, \ldots, w_n \rangle$.
Il valore più probabile di $f(d)$ è:

$$\begin{aligned} c_{MAP} &= \arg\max_{c_j \in C} P(c_j | w_1, w_2, \ldots, w_n) \\ &= \arg\max_{c_j \in C} \frac{P(w_1, w_2, \ldots, w_n | c_j) P(c_j)}{P(w_1, w_2, \ldots, w_n)} \\ &= \arg\max_{c_j} \in C P(w_1, w_2, \ldots, w_n | c_j) P(c_j) \end{aligned}$$

tenendo presente l'assunzione di Naïve Bayes:

$$P(w_1, w_2, \ldots, w_n | c_j) = \prod_i P(w_i | c_j)$$

definendo quindi il classificatore ottimo:

$$c_{NB} = \arg\max_{c_j \in V} P(c_j) \prod_i P(w_i | c_j)$$

Una considerazione importante che deve essere fatta relativa alla distribuzione degli esempi a disposizione è relativa al cosidetto fenomeno dell'*overfitting*, ossia quando per una data classe c_j esiste almeno una istanza che non possiede un attributo di valore w_i. In questa situazione si ha che:

$$\hat{P}(w_i | c_j) = 0 \text{ e quindi } \hat{P}(c_j) \prod_i \hat{P}(w_i | c_j) = 0$$

Una soluzione tipica è la stima Bayesiana per $\hat{P}(w_i | c_j)$:

$$\hat{P}(w_i | c_j) \leftarrow \frac{n_c - mp}{n - m}$$

in cui:

- n è il numero di esempi di apprendimento per cui $v = v_j$
- n_c è il numero di esempi per cui $v = v_j$ e $w = w_i$
- p è la stima a priori di $\hat{P}(w_i | v_j)$
- m è il peso dato a priori, ossia il numero di esempi "virtuali".

Vediamo ora di seguito come può essere adottato questo classificatore nell'ambito della classificazione di documenti testuali.

Fase di apprendimento: ***LearnNB****(Esempi, C)*

1. Collezionare tutte le parole ed altri token che occorrono in *Esempi*

$Vocabolario \leftarrow$ tutte le parole distinte ed altri token in $Esempi$

2. Stimare i termini $P(c_j)$ e $P(w_i|c_j)$

for valore di target $c_j \in C$ **do**
  $doc_j \leftarrow$ sottoinsieme di esempi per cui il valore di target è c_j
  $P(c_j) \leftarrow \frac{|doc_j|}{|Esempi|}$
  $Text_j \leftarrow$ concatenazione documenti in doc_j
  $n \leftarrow$ numero di parole e token totali in $Text_j$

  for $w_i \in Vocabolario$ **do**
    $n_i \leftarrow$ numero di volte che w_i occorre in $Text_j$
    $P(w_i|c_j) \leftarrow \frac{n_i - 1}{n - |Vocabolario|}$
  end for
end for

Fase di classificazione: *ClassifyNB(doc)*

1. Individuare i termini che compaiono nel $Vocabolario$

$Posizioni \leftarrow$ tutte le posizioni in cui compare un termine presente in $Vocabolario$

2. Restituire come output la classe c_j che massimizza la probabilità a posteriori

$\arg\max_{c_j \in V} P(c_j) \prod_{i \in Posizioni} P(w_i|c_j)$

Bayesian Logistic Regression

È possibile anche stimare direttamente la probabilità $P(c|d)$. La *Bayesian Logistic Regression* è un approccio statistico al problema. Assumendo la categorizzazione binaria, il modello diventa:

$$P(c|d) = \varphi(\beta \cdot d) = \varphi(\sum_i \beta_i d_i)$$

dove $c = \pm 1$ è il valore della funzione di *membership*, $d = (d_1, d_2, \ldots)$ è la rappresentazione dei documenti nel *feature space*, β sono i parametri del modello (da apprendere) e φ è la *logistic link function*

$$\varphi(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

In questo modello è importante evitare il fenomeno di *overfitting*. Si possono usare varie distribuzioni di probabilità per il vettore β, ad esempio è possibile utilizzare una distribuzione di probabilità gaussiana con media 0 e varianza τ:

$$p(\beta_i|\tau) = \frac{1}{\sqrt{2\pi\tau}} \exp(\frac{-\beta_i^2}{2\tau})$$

In alternativa si può adottare la distribuzione di Laplace, per permettere ad alcune probabilità di assumere valore zero e quindi ammettere la sparsità del modello. La probabilità a posteriori β, una volta stimata, verrà utilizzata per le previsioni. Per effettuare questa stima si sceglie solitamente l'ipotesi *Maximum a posteriori*, cioè:

$$H_{MAP} = \arg\max_{\beta} l(\beta)$$

con $l(\beta)$ definita come:

$$l(\beta) = p(\beta|D) = -\left(\sum_{(d,c)\in D} ln(exp(-c\beta \cdot d) + 1)\right) + \ln p(\beta)$$

$$\ln p(\beta) = -\left(\sum_i \left(\ln\sqrt{(\tau)} + \frac{\ln 2\pi}{2} + \frac{\beta_i^2}{\tau}\right)\right)$$

per la distribuzione gaussiana vista in precedenza.

Alberi di decisione

Molti classificatori hanno il problema che non possono essere facilmente compresi dagli esseri umani. I classificatori simbolici, di cui gli alberi di decisione sono l'esempio più diffuso, non sono affetti da questo problema. Un albero di decisione è una struttura in cui i nodi interni sono etichettati dalle *features*, gli archi uscenti dai nodi sono etichettati con test sul valore della *feature*, e le foglie con le categorie. Un esempio di albero di decisione, costruito a partire dai dati in tabella 2.1 è riportato in figura 2.2. Un albero di decisione categorizza un documento partendo dalla radice dell'albero fino alle foglie seguendo i rami in cui le condizioni sono soddisfatte dal documento; la ricerca di arresta quando viene raggiunto un nodo foglia. Al documento viene poi assegnata la categoria indicata da quella foglia. La maggioranza dei DT(*Decision Three*) usa una rappresentazione binaria dei documenti, di conseguenza gli alberi di decisione sono alberi binari. I principali algoritmi utilizzati per la costruzione degli alberi di decisione sono *ID3 C4.5* e *CART*.

Regole di decisione

I classificatori basati su regole appartengono alla famiglia dei classificatori simbolici. Le regole sono in *Disjunctive Normal Form*. Sono costruite con

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Figura 2.1: Tabella esempio di Training Set per alberi di decisione.

metodi *bottom-up*. Il classificatore iniziale (più specifico) è costruito direttamente dagli esempi nel *Training Set*, considerando ogni esempio come una clausola:

$$d_1 \wedge d_2 \wedge \ldots \wedge d_n \rightarrow c$$

Successivamente il classificatore mette in atto una serie di generalizzazioni, rimuovendo termini dalle clausole e unendo regole. Alla fine del processo viene effettuato il *pruning* similmente agli alberi di decisione, per rendere il classificatore più generale.

Regressione

La regressione è una tecnica per approssimare una funzione a valori reali conoscendo solo i suoi valori in alcuni punti. Questa tecnica può essere applicata alla *Text Categorization* approssimando la funzione di assegnamento delle categorie ai documenti. Un metodo (lineare) che si può utilizzare è il metodo dei minimi quadrati. La funzione di assegnamento alle categorie è vista come una matrice di dimensioni $|C|\text{x}|F|$, che descrive una trasformazione lineare dallo spazio delle *features* a quello di tutti i possibili assegnamenti di categorie. Per costruire il classificatore, cerchiamo la matrice che

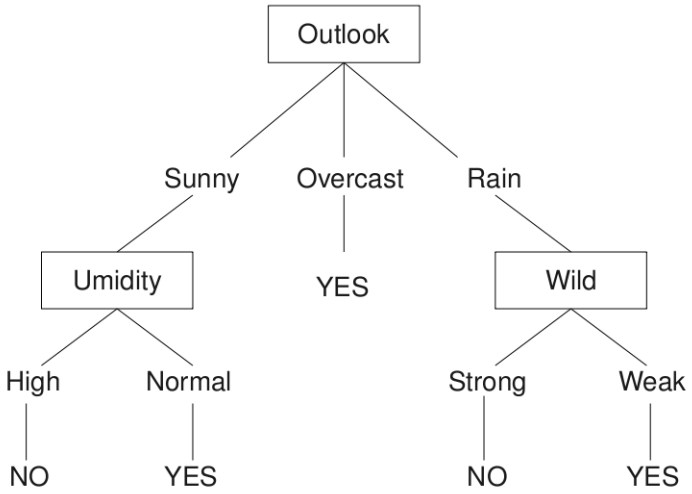

Figura 2.2: Esempio di albero di decisione.

si adatti meglio ai dati di *training*. In dettaglio, minimizziamo l'errore nel *Training Set* secondo la formula:

$$M = \arg\min_{M} \|MD - O\|_F$$

dove D è la matrice delle rappresentazioni degli esempi di dimensione $|F|\text{x}|TrainingSet|$, O è la matrice che rappresenta, per ogni documento, le categorie a cui appartiene di dimensione $|C|\text{x}|TrainingSet|$, mentre $\|\cdot\|_F$ indica la norma di Frobenius, definita come:

$$\|A\|_F = \sqrt{\sum A_{ij}^2}$$

Metodo di Rocchio

Il classificatore di Rocchio classifica i documenti calcolando la distanza dal vettore "profilo" della categoria. Il vettore profilo per una certa categoria c_i è un vettore $(w_1, w_2, \ldots)$ calcolato secondo la formula:

$$w_i = \frac{\alpha}{POS(c)} \sum_{d \in POS(c)} w_{di} - \frac{\beta}{NEG(c)} \sum_{d \in NEG(c)} w_{di}$$

dove $POS(c)$ e $NEG(c)$ sono gli insiemi dei documenti di *training* che rispettivamente appartengono o non appartengono alla categoria c, e w_{di} è il peso dell'i-esima *feature* nel documento d. Solitamente gli esempi positivi

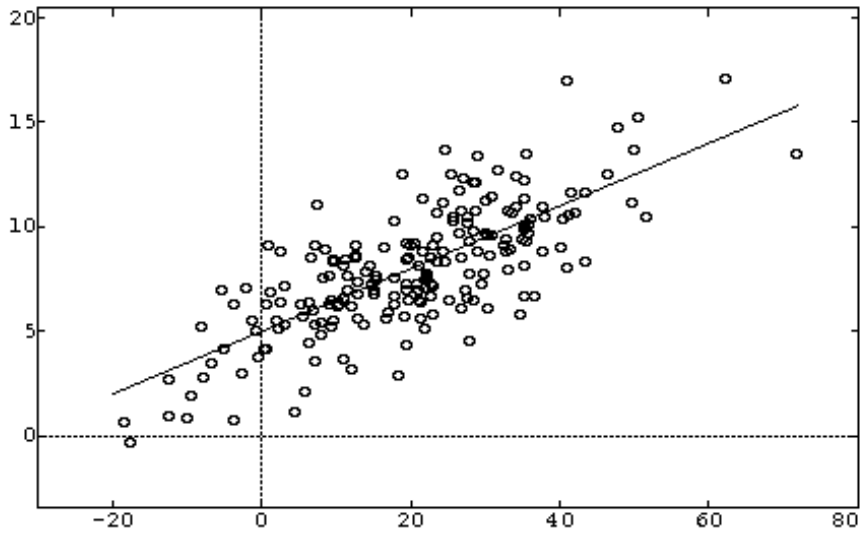

Figura 2.3: Esempio di regressione lineare.

sono considerati più importanti, quindi $\alpha >> \beta$. Il metodo di Rocchio è molto facile da implementare e computazionalmente efficiente, ma ha performance mediocri. Tuttavia esistono alcune variazioni di questo metodo che usano vari accorgimenti considerate lo stato dell'arte.

Classificatori *Example-based*

Questo tipo di classificatori calcola le differenze tra il documento da classificare e gli esempi a disposizione. Sono chiamati anche *lazy learners* poichè eseguono lavoro solo quando devono classificare un nuovo esempio. La fase di *training* è costituita semplicemente dal rappresentare (e tenere in memoria) i documenti del *Training Set*, insieme alle rispettive *label*. Il più noto classificatore appartenente a questa categoria è *K-Nearest-Neighbour*.

KNN

Per stabilire a quale categoria appartiene un documento d, kNN controlla a quale categoria appartengono i k documenti del *Training Set* più simili a d, ed assegna la categoria di maggioranza. È cruciale la scelta del numero k, il cui valore può essere appreso attraverso un *validation set*, oppure stabilendo un valore a priori. KNN è tra gli algoritmi di classificazione più performanti, perchè non assume che gli esempi siano linearmente separabili. Il problema

è l'elevato costo computazionale della classificazione, infatti deve essere calcolata la similarità con ogni esempio del *Training Set* per ogni documento da classificare.

Classifier Committees

L'idea di base dei comitati di classificatori è quella che generalmente un gruppo di esperti, combinando la loro conoscenza, può ottenere risultati migliori rispetto al singolo. Nel metodo *bagging* i singoli classificatori vengono appresi in parallelo dallo stesso *Training Set*. Affinchè questo approccio funzioni, i singoli classificatori devono essere significativamente diversi l'uno dall'altro (nella rappresentazione dei documenti, o nei metodi di apprendimento). I risultati dei singoli classificatori devono poi essere combinati. Il metodo più semplice è il voto a maggioranza, in cui una categoria è assegnata ad un documento se e solo se (k+1)/2 dei classificatori votano per quella categoria. Altra possibilità è una votazione pesata a maggioranza, dove la *Categorization Status Value* è data dalla somma pesata delle CSV dei singoli classificatori. Il valore dei pesi può essere calcolato sulla base del comportamento dei classificatori in un *validation set*.

Il *boosting* è un altro metodo per combinare classificatori. Nel *boosting* i classificatori sono appresi in sequenza. Prima di apprendere l'i-esimo classificatore, i pesi dei singoli esempi nel *Training Set* vengono ricalcolati dando maggior peso agli esempi che sono stati classificati scorrettamente dal classificatore precedente. AdaBoost è l'algoritmo più conosciuto tra quelli di *boosting*.

2.6 *Support Vector Machines* (SVM) in dettaglio

Le *Support Vector Machine* rientrano nella categoria dei cosidetti *metodi Kernel*, ossia di quegli algoritmi adottati per la *pattern analysis* che cercano di individuare e di studiare le relazioni che esistono tra dati di qualsiasi tipologia. Nello specifico i dati, rappresentati da vettori, vengono mappati in uno spazio a dimensione molto superiore, *feature space*, rispetto a quello di input. Grazie alle funzioni kernel è possibile calcolare il prodotto scalare tra esempi, che corrisponde ad una misura di distanza, dopo la trasformazione, senza prima calcolare esplicitamente i vettori nello spazio delle *feature*. L'adozione delle funzioni *kernel* comporta due vantaggi:

1. la possibilità di risolvere casi non linearmente separabili con metodi lineari;

2. l'opportunità di utilizzare il classificatore anche nel caso in cui la dimensione della rappresentazione nel *vector space* non è fissata.

Le *Support Vector Machine* sono dei classificatori lineari binari, in cui ad ogni esempio viene associata una etichetta scegliendo tra due possibili *label*, che per convenzione sono $+1$ (esempi positivi) e -1 (esempi negativi). Ogni esempio viene rappresentato tramite un vettore $\mathbf{x}$ con componenti x_i. La notazione $\mathbf{x}_i$ indica i-esimo vettore nel dataset $\{(\mathbf{x}_i, y_i)_{i=1}^n\}$, dove y_i è la label associata all'esempio $\mathbf{x}_i$.

2.6.1 Definizioni fondamentali

Un concetto fondamentale necessario per definire un classificatore lineare è il prodotto scalare tra due vettori, definito da $\mathbf{w}^T\mathbf{x} = \sum_i w_i x_i$. Data una funzione discriminante della forma $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, in cui $\mathbf{w}$ è il vettore dei pesi e b il bias, l'iperpiano

$$\{\mathbf{x} : f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0\}$$

divide lo spazio in due: il segno della funzione discriminante $f(\mathbf{x})$ stabilisce la posizione del punto rispetto l'iperpiano. Quest'ultimo rappresenta quindi il *decision boundary* del classificatore. In base alla disposizione dei dati si individua un *decision boundary* lineare (Fig. 2.4) o non lineare, e di conseguenza il relativo classificatore.

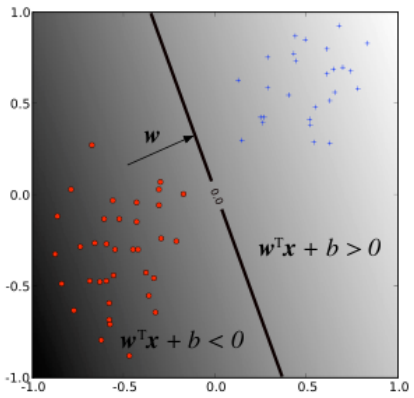

Figura 2.4: *Decision boundary* di un classificatore lineare [16].

Un'altra definizione fondamentale è quella di margine. Dato un'iperpiano indichiamo con $\mathbf{x}_+$ ($\mathbf{x}_-$) il punto più vicino all'iperpiano tra gli esempi positivi (negativi). Ricorrendo a semplici considerazioni geometriche il margine da un iperpiano f rispetto a un dataset D è definito come:

$$m_D(f) = \frac{1}{2}\hat{\mathbf{w}}^T(\mathbf{x}_+ - \mathbf{x}_-)$$

dove $\hat{\mathbf{w}}$ è un vettore di norma 1 con la stessa direzione di $\mathbf{w}$ e assumiamo che i due vettori $\mathbf{x}_+$ e $\mathbf{x}_-$ siano equidistanti dal *decision boundary*. Eseguendo una

serie di operazioni che rendono significativo il valore del margine è possibile ottenere una ulteriore formulazione:

$$m_D(f) = \frac{1}{2}\hat{\mathbf{w}}^T(\mathbf{x}_+ - \mathbf{x}_-) = \frac{1}{||\mathbf{w}||}$$

2.6.2 Definizione di SVM

La SVM rappresenta un classificatore che massimizza il margine $\frac{1}{||\mathbf{w}||}$, ossia minimizza $||\mathbf{w}||^2$. Di conseguenza si può formulare il seguente problema di ottimizzazione con vincoli:

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad i = 1, 2, \ldots, n.$$

I vincoli garantiscono la correttezza della classificazione degli esempi, assumendo che siano linearmente separabili. Molto spesso però ci si trova di fronte a distribuzioni di esempi non linearmente separabili. In questi casi vengono aggiunte, alla formulazione dei vincoli, delle variabili $\xi_i \geq 0$ (variabili di *slack*) per ammettere la presenza di eventuali *misclassification*:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \; i = 1, 2, \ldots, n.$$

Se ξ_i è compresa tra 0 e 1 (estremi inclusi) significa che l'esempio si trova nel margine (errore di margine), altrimenti se $\xi_i > 1$ siamo in presenza di un errore di classificazione. Per penalizzare quindi questi errori viene aggiunto alla definizione della funzione obiettivo un certo fattore C,

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \xi_i \geq 0$$

La variabile C stabilisce una relazione tra la necessità di massimizzare il margine e quella di minimizzare il valore si slack. In questo caso si parla di *Soft Margin SVM*. Il problema viene risolto passando alla sua forma duale usando il metodo dei moltiplicatori di Lagrange. La formulazione diventa quindi:

$$\underset{\alpha}{\text{maximize}} \quad \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_i \alpha_i = 0 \quad 0 \leq \alpha_i \leq C$$

Il vettore $\mathbf{w}$ dei pesi può essere espresso in termini degli esempi di input:

$$\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$$

Gli esempi $\mathbf{x}_i$ con $\alpha_i > 0$ vengono definiti vettori di supporto.

2.6.3 Scelta dei parametri

Durante la fase di *training* vengono individuati i due parametri fondamentali che identificano l'iperpiano, le α_i e b. Oltre a questi ci sono altri valori che devono essere selezionati e che rientrano nei cosidetti *hyperparameters*. Tra questi ritroviamo ad esempio la costante C prevista nella *Soft-Margin SVM*, e tutti gli altri parametri da cui dipende la funzione kernel scelta. Il valore di C contribuisce in modo non indifferente alla scelta dell'iperpiano separatore. Per un valore di C elevato infatti si assegna una penalità molto alta agli errori (anche errori di margine), e così facendo viene selezionato l'iperpiano che comporta il minor numero di errori possibile anche a costo di ridurre in maniera sostanziale il margine (Fig. 2.5). Nelle funzioni kernel,

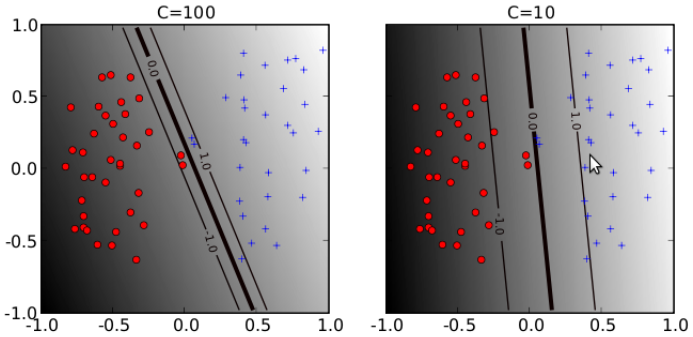

Figura 2.5: Effetto del parametro C nella *Soft-Margin SVM* [16].

la scelta dei parametri riveste un ruolo fondamentale. Di seguito vedremo alcune caratteristiche dei kernel più noti: kernel lineare, kernel polinomiale e kernel gaussiano. I kernel polinomiali di grado elevato consentono di avere un *decision boundary* molto flessibile. Quelli lineari possono essere utilizzati con risultati apprezzabili solamente nei casi in cui gli esempi a disposizione sono linearmente separabili (Fig. 2.6). Tuttavia se il grado è troppo elevato si può incorrere nel cosidetto problema dell'*overfitting*. Per quanto riguarda invece il kernel gaussiano (Fig. 2.7), definito come

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma ||\mathbf{x} - \mathbf{x}'||^2\right)$$

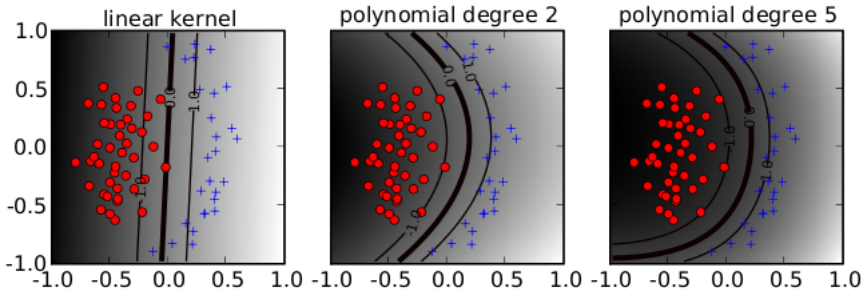

Figura 2.6: Effetto del grado in un kernel polinomiale [16].

il parametro γ influenza decisamente la flessibilità del *decision boundary*. Come possiamo vedere in figura 2.7 con valori bassi di γ si ha un *decision boundary* molto vicino a quello lineare. Man mano che il parametro aumenta, cresce anche la flessibilità del confine di separazione. Anche in questo caso se il valore di γ è troppo elevato si incorre però nell'*overfitting*. Quando si opera con un kernel polinomiale l'unico parametro da fissare è la costante C del *soft-margin*. Per i kernel polinomiali e gaussiani invece lo spazio di ricerca degli *hyperparameter* ha due dimensioni. In questo caso viene costruita una griglia con tutte le possibili combinazioni di valori e per ogni suo punto vengono valutate le performance del classificatore. Ad esempio se consideriamo la coppia (γ, C) riducendo il valore di γ diminuisce la curvatura del confine, aumentando invece il valore di C si forza il *decision boundary* ad adattarsi alla penalità associata agli errori e agli errori di margine (Fig. 2.8).

2.7 Valutazione dei classificatori di documenti testuali

Come nel caso di sistemi IR, anche per la classificazione di testi, la valutazione delle *performance* del classificatore viene condotta in modo sperimentale. Viene misurata la *effectiveness*, ossia l'abilità di prendere la decisione corretta, e in alcuni casi viene presa in considerazione anche la *efficiency*.

Una prima misura che può essere usata per la valutazione del classificatore è l'*accuracy*, ossia la frazione di documenti classificati correttamente. Viene definita come:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Tuttavia si tratta di una misura poco adatta nell'ambito della *Text Cat-*

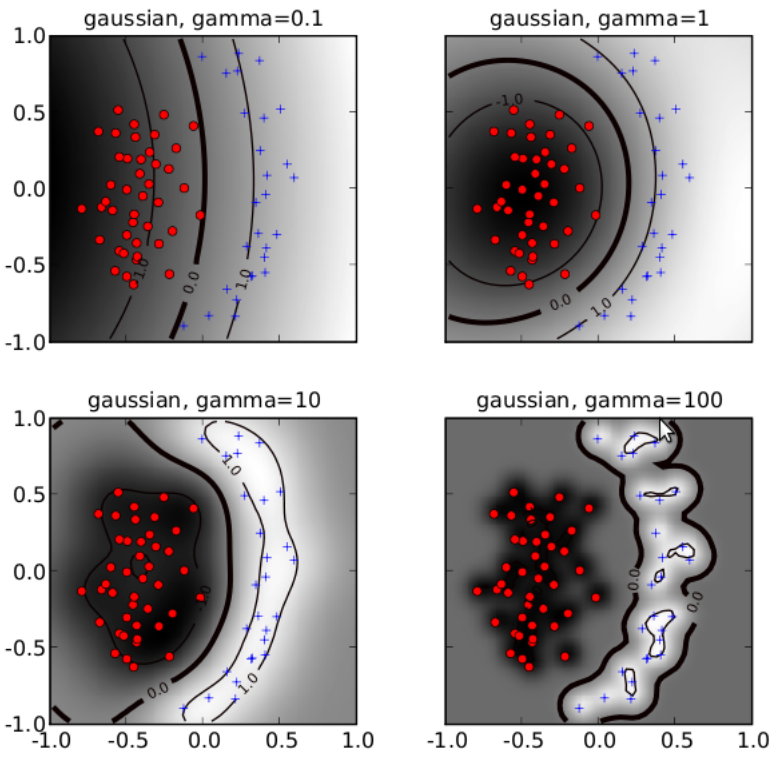

Figura 2.7: Effetto del parametro γ in un kernel Gaussiano, fissato il valore di C [16].

egorization. Tipicamente, infatti, il denominatore assume valori piuttosto grandi, rendendo minima l'influenza delle variazioni del numero di decisioni corrette sul valore finale di *accuracy*. Inoltre nel caso in cui le categorie a disposizione sono sbilanciate, il *trivial rejector*, ossia il classificatore che classifa tutte le istanze con la classe più numerosa, ottiene valori di *accuracy* elevati. Per i motivi appena esposti, si ricorre quindi ad altre due misure per stimare la *effectiveness* del classificatore: la *Precision*(π) e la *Recall*(ρ).

La *precision* ("correttezza") rispetto alla categoria c_i (π_i) è la probabilità che, se un documento selezionato in maniera casuale è classificato nella categoria c_i, la decisione è corretta, ossia:

$$P(\breve{\Phi}(d_x, c_i) = T | \Phi(d_x, c_i) = T)$$

La stima della *Precision* in relazione alla classe c_i è ottenuta come segue:

$$\hat{\pi}_i = \frac{TP_i}{TP_i + FP_i}$$

In modo analogo, la *recall* ("completezza") rispetto alla categoria c_i (ρ_i) è definita come la probabilità che se un documento d_x deve essere classificato

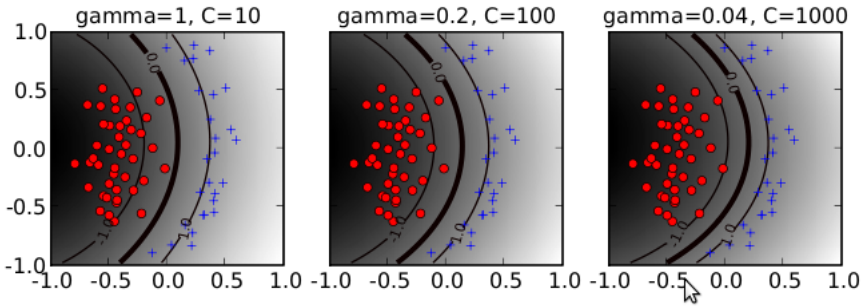

Figura 2.8: Effetto di differenti conbinazioni degli *hyperparameter* [16].

con la classe c_i, questa decisione viene presa dal classificatore.

$$P(\Phi(d_x, c_i) = T | \breve{\Phi}(d_x, c_i) = T)$$

La stima della *Recall* in relazione alla classe c_i è ottenuta come segue:

$$\hat{\rho}_i = \frac{TP_i}{TP_i + FN_i}$$

Per valutare i valori di *Precision* e *Recall* globali sarà necessario calcolare la media dei singoli valori ottenuti in relazione alle diverse classi c_i. È possibile distinguere due diverse modalità per calcolare i valori di π e ρ:

- *microaveraging* (μ) - *Precision* e *Recall* sono ottenute sommando tutte le singole decisioni:

$$\hat{\pi}^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)}$$

$$\hat{\rho}^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)}$$

- *macroaveraging* (M) - *Precision* e *Recall* vengono prima valutate singolarmente per ogni categoria, e successivamente viene calcolata la media dei singoli valori a disposizione:

$$\hat{\pi}^{M} = \frac{\sum_{i=1}^{|C|} \hat{\pi}_i}{|C|}$$

$$\hat{\rho}^{M} = \frac{\sum_{i=1}^{|C|} \hat{\rho}_i}{|C|}$$

I due metodi appena descritti possono dare risultati decisamente differenti a seconda della distribuzione degli esempi per categorie. Ad esempio, se all'interno delle diverse classi ci sono pochi esempi positivi, i valori di *Precision* e *Recall* calcolati saranno più alti con la *macroaveraging* rispetto al calcolo ottenuto applicando la *microaveraging*. Esistono diversi modi per combinare i valori di π e ρ. Il più noto è la funzione F_β, con $0 \leq \beta \leq +\infty$ definita come:

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}$$

In questa formula β può essere considerato come il grado di importanza attribuito a π e ρ. Se $\beta = 0$ allora F_β coincide con π, altrimenti se $\beta = +\infty$ allora F_β è uguale a ρ. Di solito viene usato il valore $\beta = 1$, in modo tale da attribuire la stessa importanza alla *Precision* e alla *Recall*.

2.8 *Text Classification* applicata ai messaggi di posta elettronica

Negli ultimi anni la *Text Categorization* si sta rivelando un'area di ricerca particolamente attiva e in continua evoluzione. Molti approcci esistenti vengono applicati alla moltitudine di documenti resi disponibili attraverso Internet. I metodi di *Text Categorization* sono stati applicati ampiamente al problema dello *spam filtering*. Ad esempio, Drucker et al. (1999) [4], analizzano l'uso delle Support Vector Machine, per identificare le email come spam o come non spam. Gli autori inoltre mettono a confronto le prestazioni di altri algoritmi, tra cui Ripper, Rocchio e gli alberi di decisione con *boosting*; le SVM hanno condotto ai migliori risultati sia in termini di *accuracy* che di efficienza.

Vi sono chiaramente altri problemi legati alla gestione delle email, come ad esempio, la necessità di identificare correttamente i diversi *thread* all'interno del flusso globale. Lewis et al. (1997) [5] dimostrano che riconoscere i *thread* corrispondenti alle conversazioni tra due o più persone, che avvengono tramite lo scambio di diversi messaggi, può essere trattato come un compito di *language processing*. I loro primi test hanno condotto a buoni risultati in termini di efficacia, ricorrendo ai metodi di matching testuale standard di *Information Retrieval*.

Altro problema che sta alla base del lavoro di tesi e che coinvolge i metodi di *Text Categorization*, è l'organizzazione delle email in *folder*, con l'eventuale possibilità di generare queste ultime in modo automatico. Giacoletto et al. [6], evidenziano il fatto che le attività aziendali subiscono un continuo cambiamento nel corso del tempo, e quindi una struttura fissa di *directory* in cui organizzarle può risultare inadatta e condurre a risultati inaccurati. Viene proposto quindi un nuovo approccio che consente di integrare alla gerarchia fissa in cui ripartire e organizzare le attività, definita a priori dal-

l'utente, ulteriori schemi di classificazione, sulla base di un'accurata analisi del contenuto delle nuove email.

Payne e Edwards (1997) [7] usano due diversi paradigmi di apprendimento, CN2 basato sull'induzione di regole e k-NN (k Nearest Neighbour), per suddividere e organizzare le loro email personali. Mettendo a confronto i risultati conseguiti, variando alcuni parametri e considerando o meno alcune feature legate alla struttura delle email, le prestazioni migliori sono state ottenute con l'algoritmo CN2 basato sulla generazione di un set ordinato di regole (Clark and Niblett, 1989 [8]). Cohen (1996)[9] considera invece un certo numero di problemi di classificazione binaria con una *directory* fissata e tutte le altre presenti nella gerarchia, e confronta le prestazioni del suo algoritmo RIPPER con quelle del classificatore basato sul calcolo dei pesi tramite *tfidf*. Con un numero limitato di esempi preclassificati, Cohen riesce a mostrare che l'approccio RIPPER, consistente nella generazione di un insieme di regole basato sull'individuazione di parole chiave, considerando però solo la presenza o meno dei termini, e non la frequenza, fornisce delle prestazioni superiori rispetto al classificatore basato sul calcolo di *tfidf*.

Successivamente Provost (1999) [10] dimostra che l'adozione del modello di Naïve Bayes, nell'ambito della *Text Categorization* applicata ai messaggi di posta elettronica, comporta dei risultati migliori rispetto all'approccio di Cohen basato su RIPPER. Nel 2000 Rennie [11] usa Naïve Bayes per realizzare un sistema di classificazione che è in grado di suggerire per ogni email in ingresso 3 possibili classi di appartenenza, selezionate dalle *directory* disponibili. I risultati sono decisamente incoraggianti, infatti con un alta probabilità, la classe corretta è presente tra le tre etichette proposte dal classificatore e questo comporta una riduzione notevole del lavoro di selezione a carico degli utenti finali. Kiritchenko e Matwin (2001) [12] sono i primi ad impiegare le famose *Support Vector Machine* per la classificazione di email in *folder* e a mostrare i miglioramenti rispetto all'approccio bayesiano.

La classificazione di email in *folder* costituisce un particolare problema sotto diversi aspetti e comporta una serie di difficoltà che lo distinguono dal tradizionale approccio della TC. Nel corso del tempo, gli utenti possono inserire nella gerarchia di *directory* fissata inizialmente nuove categorie, oppure eliminarne altre già presenti. Segal e Kephart (2000) [13] applicano un classificatore con schema *tfidf* in un sistema adattato all'apprendimento incrementale *time-based* per la classificazione di email in *folder*. Inoltre è opportuno evidenziare il fatto che le email sono inserite all'interno di un flusso temporale e quindi alcuni messaggi assumono un certo significato se considerati in sequenza ad altri. Nel 2004, Kiritchenko et al. [14] propongono una soluzione che prende in considerazione le relazioni temporali in una sequenza di email, individuando dei pattern temporali da integrare alle informazioni utilizzate dai metodi di apprendimento *content-based*. Per apportare ulteriori miglioramenti all'identificazione delle classi di appartenenza delle email in ingresso, è possibile introdurre nuovi approcci che non

si basano solamente sull'analisi di tecniche statistiche, ma che considerano anche altri aspetti esterni, tra cui per esempio l'analisi del traffico in uscita dalla casella dell'utente. A tal proposito, Martin et al. (2005), introducono un nuovo metodo che si basa sull'individuazione di *behavioral feature* per distinguere le email di spam dal resto del traffico in entrata.

Un aspetto che sicuramente deve essere analizzato riguarda la struttura e i campi in cui si articola il testo di una email. È necessario fare delle scelte per stabilire quali sono le informazioni da considerare per la classificazione, e quali invece possono essere trascurate. All'interno del *body* delle email è possibile identificare delle zone con particolari caratteristiche, e a quel punto selezionare quelle che contribuiscono maggiormente a migliorare la classificazione. Lampert et al. (2009) [15] propongono un nuovo metodo basato sulla SVM per individuare all'interno delle email 9 segmenti, distinti in base a specifici criteri, tra cui quelli grafici, ortografici e lessicali. I test hanno fornito dei risultati incoraggiando pari al 87,01% di *accuracy*, valore che sale ulteriormente se le zone astratte da indentificare vengono ridotte a due (93,60%).

Tuttavia è bene precisare che la classificazione automatica di email in *folder* rappresenta un'area di ricerca non ancora affermata e consolidata. Questo è dovuto anche all'assenza di un dataset standard e pubblico di email su cui poter valutare i vari metodi di classificazione e poter comparare i risultati derivanti dal lavoro dei vari ricercatori. È comunque disponibile un corpus pubblico di email messo a disposizione dalla Enron Corporation, costituito da 500000 email provenienti dagli account di 150 persone.

Sulla base di queste premesse l'attività di tesi si concentra sull'analisi e lo studio delle principali tecniche di classificazione testuali applicabili al contesto dei messaggi di posta elettronica, ponendo particolare attenzione ai dettagli delle singole fasi che manipolano e modificano i dati a disposizione per la rappresentazione finale, usata dal modello di classificazione.

Parte II

Sistema di classificazione

Capitolo 3

Sistema di classificazione preesistente

Prima di analizzare in dettaglio le modifiche e i miglioramenti che sono stati realizzati nel corso dell'attività di tesi, è indispensabile fornire una panoramica del sistema di classificazione iniziale. Nelle sezioni successive verrà messo in evidenza lo scenario d'uso e il relativo flusso dei dati, nonchè la struttura dell'applicazione con un'analisi dettagliata delle singole componenti.

3.1 Scenario d'uso

Il sistema di classificazione è stato sviluppato per soddisfare i requisiti di una particolare azienda di Padova che opera nel settore della progettazione e realizzazione di impianti per l'integrazione di tecnologie per la città "digitale", dell'info-mobilità, delle tecnologie di connettività e di rete *multi-layer*. Una delle principali aree in cui opera l'azienda è l'erogazione di servizi di *call center* integrati, con eventuale estensione a *contact center*. Proprio in relazione a questa attività, l'azienda ha richiesto lo sviluppo di un'applicazione che fosse in grado di instradare le email, in arrivo presso una particolare casella di posta elettronica, all'operatore più adatto per la lettura e l'invio dell'eventuale risposta, interfacciandosi con il sistema di *ticketing* (OTRS) adottato nel *call center* aziendale. La tassonomia di classi in base alla quale viene mappata una email corrisponde all'insieme di servizi di *call center* forniti dall'azienda.

3.1.1 Flusso dei dati

Il sistema di interfacciamento tra il classificatore e i sistemi di *ticketing* aziendali hanno portato all'individuazione di un particolare flusso di dati. Periodicamente il classificatore preleva dalla casella di posta indicata dall'azienda le email da classificare e dopo aver eseguito la fase di *training*

iniziale su dati preclassificati, con il modello appreso procede alla classificazione. I risultati proposti dal classificatore vengono resi disponibili al sistema di *ticketing* adottato dall'azienda sottoforma di file in formato XML. In figura 3.1 viene illustrato il flusso dei dati durante la classificazione.

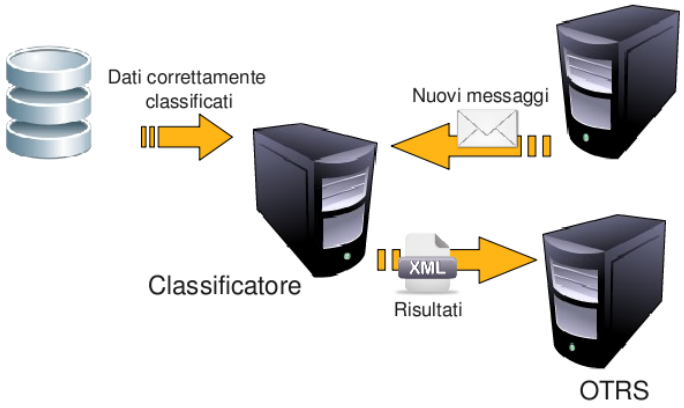

Figura 3.1: Flusso dei dati durante la classificazione.

Una volta resi disponibili i risultati della classificazione, per completare il ciclo, è necessario procedere alla lettura del *feedback* fornito dall'azienda. Gli operatori aziendali, forniscono la classificazione corretta per le email che sono state processate, inmodo tale che questi dati possano essere incorporati nella base di conoscenza iniziale del sistema. L'aggiunta di nuova conoscenza consente poi un raffinamento ulteriore del modello appreso in fase di *training* (Fig. 3.2).

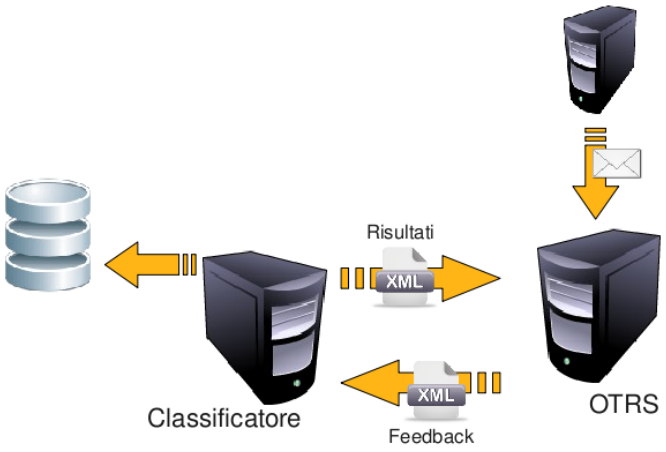

Figura 3.2: Flusso dei dati per la gestione del *feedback*.

3.1.2 Comunicazione tra classificatore e sistema aziendale

La comunicazione tra il classificatore e il sistema OTRS avviene attraverso un protocollo di scambio di file XML via HTTP. I file vengono prodotti da un lato dell'applicazione e letti in modo asincrono tramite una HTTP *get*. Viene adottata la stessa struttura del file XML sia per i risultati della classificazione che per la lettura del *feedback*. Di seguito viene presentato il formato del file XML, che corrisponde a un certo lotto prodotto in modo progressivo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<classificatore>
   <lotto>
     <data>DD/MM/YYYY HH:MM:SS</data>
     <numero>numero identificativo del lotto</numero>
   </lotto>
   <email>
     <uid>uid assegnato da protocollo IMAP</uid>
     <mid>message ID assegnato da protocollo IMAP</mid>
     <coda>
       <id1> id classificazione automatica 1</id1>
       <id2> id classificazione automatica 2</id2>
       <id3> id classificazione automatica 3</id3>
       <id4> id classificazione automatica 4</id4>
       <id5> id classificazione automatica 5</id5>
     </coda>
   </email>
   <email>
     <uid>uid assegnato da protocollo IMAP</uid>
     <mid>message ID assegnato da protocollo IMAP</mid>
     <coda>
       <id1> id classificazione automatica 1</id1>
       <id2> id classificazione automatica 2</id2>
       <id3> id classificazione automatica 3</id3>
       <id4> id classificazione automatica 4</id4>
       <id5> id classificazione automatica 5</id5>
     </coda>
   </email>
</classificatore>
```

All'interno del file, che identifica un certo lotto, le email mantengono un doppio identificativo:

- l'**UID** è l'identificatore universale IMAP. In realtà il campo contiene il numero progressivo del messaggio nella casella di posta elettronica, essendo il valore di UID vero e proprio inaccessibile alle classi Java che trattano le email.
- l'**MID** è l'ID assegnato al messaggio di posta elettronica, unico all'interno del mail server. Indicando in modo univoco il messaggio di posta elettronica, è considerato sufficiente per l'identificazione da parte del sistema di *ticketing* (OTRS).

Per ogni email viene indicata la classificazione suggerita dal classificatore specificando i codici numerici corrispondenti alle categorie presenti nella tassonomia. Si è convenuto ad inserire fino ad un massimo di cinque label per ogni email disponendole in ordine decrescente di confidenza. Per quanto riguarda invece il file XML di *feedback*, viene indicata una sola categoria, che corrisponde alla classificazione corretta.

Oltre alla struttura del file XML è stato anche definito il protocollo di scambio dei risultati. Nello specifico, i file devono essere pubblicati via HTTP ad un indirizzo pubblico, del tipo:

http://<indirizzo servizio>

Nel momento in cui si invia una richiesta a tale indirizzo, come risposta si ottiene l'ultimo lotto prodotto sia nel caso di classificazione che di *feedback*; il codice HTTP 404:*Not Found* verrà invece ricevuto come risposta se non ci sono lotti disponibili. Per richiedere un lotto specifico, all'interno della richiesta dovrà essere indicato il numero corrispondente a quello desiderato:

http://<indirizzo servizio>?lotto=<numero>

Per avere la garanzia di leggere tutti i lotti che sono stati prodotti, si può richiedere l'ultimo lotto disponibile, e successivamente leggere, richiedendoli in modo specifico, tutti i lotti compresi tra l'ultimo letto e l'ultimo disponibile.

3.2 Architettura generale

All'interno del sistema di classificazione è possibile individuare quattro elementi principali (Fig 3.3):

- un **DBMS** per la memorizzazione dei dati;
- un **Server email** che rappresenta la fonte di email da classificare;

- il **Classificatore di email**, con le varie componenti di ML, per la classificazione delle email;
- il **Sistema aziendale** che consente l'integrazione dei risultati (ad esempio OTRS) di classificazione all'interno dell'azienda.

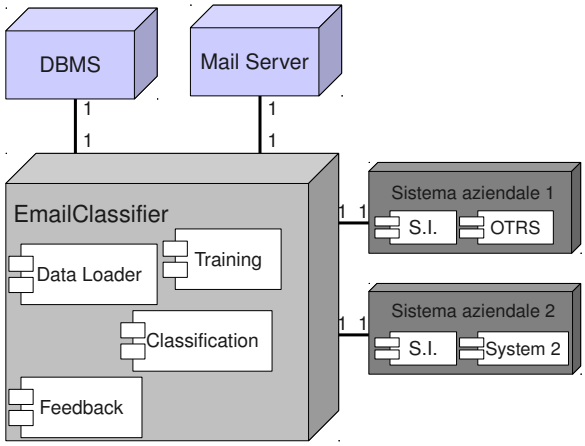

Figura 3.3: Architettura generale.

All'interno del modulo di *Machine Learning* si distinguono tre componenti distinte per il *training*, il calcolo della classificazione dei nuovi messaggi e la lettura del *feedback*. Un quarto modulo si occupa di effettuare il caricamento iniziale della base di dati per l'apprendimento di un primo modello di classificazione. Inoltre, come appare evidente in figura 3.3 l'interfaccia utente non risiede necessariamente sul server che si occupa della classificazione, ma di solito viene spostata presso i sistemi interni alle aziende.

Il sistema ricorre al supporto di un database MySQL per la memorizzazione dei dati. Il database comprende una tabella *Confirmed* contenente tutti e soli i dati di cui è nota la classificazione corretta, e una tabella *ToBeConfirmed* in cui vengono memorizzati i dati prodotti dal classificatore e in attesa di supervisione.

La lettura delle email da classificare viene effettuata tramite accesso IMAP alla casella di posta specificata in fase di configurazione.

3.2.1 Componenti di *Machine Learning*

All'interno del sistema di classificazione i thread associati alle diverse operazioni di classificazione accedono a dati condivisi, ed è pertanto necessaria una gestione esplicita dei lock per garantire il corretto accesso alla base di dati, e al file system. Sono stati implementati in modo tale da consentire

un " *soft stop*" del sistema, ossia l'interruzione del ciclo di funzionamento; per ognuno di essi viene messa a disposizione una modalità di testing che rappresenta una singola esecuzione del thread che verifica il corretto funzionamento della singola componente.

Thread di *training*

L'operazione di *training* è particolarmente complessa e può richiedere un certo periodo di tempo non trascurabile. Pertanto il thread associato viene eseguito ogni 24 ore durante le ore notturne. Il thread verifica se vi sono nuovi dati supervisionati nella tabella *Confirmed* della base di dati, e in caso affermativo viene rieseguito il *training*. I dati per cui è nota la classificazione corretta sono letti dal database e passati al *trainer* selezionato in fase di configurazione del sistema. Al termine della fase di *training* il modello appreso viene memorizzato in un file temporaneo. In figura 3.4 viene riportato il ciclo di esecuzione del thread, in cui è possibile individuare una variabile booleana (*setActive*) che consente l'arresto e il riavvio controllato del thread (la variabile è presente in tutte le componenti).

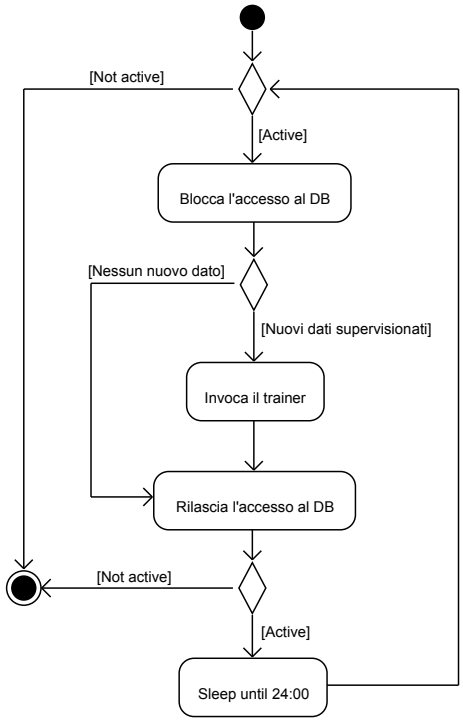

Figura 3.4: Flusso esecuzione thread di *training*.

Thread di *classifying*

La classificazione delle email viene organizzata tramite un sistema a lotti, in cui i risultati sono resi disponibili al sistema aziendale, in formato XML e in modo asincrono, attraverso una *servlet*. Per far sì che la classificazione si avvici il più possibile ad un servizio *on-demand* si è deciso di ridurre l'intervallo di tempo tra due esecuzioni successive del thread, in modo tale che venga eseguito ogni 5 minuti. Il thread preleva le nuove email dalla casella selezionata in fase di configurazione e, basandosi sul modello appreso e memorizzato in fase di *training*, produce il file XML contenente la classificazione proposta che verrà messa a disposizione dell'interfaccia utente. All'interno del nome del file XML viene indicato il numero di lotto corrispondente; per mantenere l'ordine progressivo dei lotti prodotti, il valore numerico dell'ultimo lotto generato viene salvato in un file temporaneo, in modo tale che ad ogni riavvio successivo del sistema si possa ripartire con l'indice corretto.

Servlet di presentazione dei risultati

Come detto in precedenza, per la presentazione dei risultati della classificazione si ricorre ad una *servlet*. Quest'ultima ritorna l'ultimo lotto generato (di default) oppure un lotto specifico corrispondente al valore del parametro *lotto* inserito nel *request*. La *servlet* legge i lotti con i risultati della classificazione direttamente dalla sottodirectory *classifiedEmails* della *Home* del classificatore. Il path di questa cartella è indicato come parametro di inizializzazione nel file *web.xml* della *servlet*. Il *Web Application Server* adottato è Apache Tomcat che consente la gestione delle *servlet* utilizzate per la pubblicazione dei risultati di classificazione e per la lettura dei file di *feedback*.

Thread di *feedback*

L'algoritmo per la lettura del *feedback* sfrutta la definizione del protocollo di comunicazione descritto in precedenza. Si possono distinguere le seguenti fasi:

1. lettura dell'ultimo lotto disponibile e decodifica del numero di lotto corrispondente;
2. lettura in sequenza di tutti i lotti compresi tra l'ultimo letto e il lotto appena ricevuto e salvataggio del *feedback* per ogni lotto;
3. memorizzazione dell'indice dell'ultimo lotto letto in un file temporaneo, per gestire correttamente la lettura in caso di riavvio.

Il *feedback* viene letto tramite protocollo HTTP a partire da un URL specificato in fase di configurazione del sistema, e successivamente viene eseguito il parsing del file per estrarre le informazioni contenute in esso.

Libreria di ML: *Mallet*

Per garantire il corretto funzionamento del sistema di classificazione è stata implementata una coppia di classi per il *trainer* e il classificatore. Si è deciso di adottare **Mallet**, una suite Java dedicata all'apprendimento automatico, focalizzata sulla manipolazione di documenti testuali; Mallet comprende una serie di funzionalità utili per il trattamento del testo durante l'indicizzazione (Es. Ricerca termini con massimo IG), una completa suite di classificatori implementati con diversi algoritmi e strumenti per misurarne le prestazioni. Nello specifico, il sistema utilizza per la classificazione l'algoritmo di Naïve Bayes, adottando la coppia di classi messe a disposizione da Mallet, `NaiveBayesTrainer` e `NaiveBayesClassifier` con i relativi metodi.

3.2.2 Configurazione e avvio del sistema

Per avviare il sistema è necessario configurare i parametri principali di progetto, tra cui la base di dati, il mail server e il web server per la lettura del *feedback*. All'interno del file di configurazione sono presenti inoltre altri parametri che semplificano la gestione dei path di installazione. Come mostrato in figura 3.3, all'interno del classificatore si distingue un particolare modulo, *DataLoader*, che consente il caricamento dei dati iniziali nella base di dati. Le email preclassificate che vengono utilizzate per eseguire un primo *training*, vengono prelevate da una casella di mail apposita (distinta da quella usata a regime) configurabile come parametro. All'interno delle email presenti in questa cartella la prima riga del *body* deve contenere l'indicazione della classificazione corretta, secondo la convenzione:

Coda: Id_coda

in cui *Id_coda* corrisponde al valore numerico associato alla classe di appartenenza. L'avvio del sistema risulta essere piuttosto intuitivo: vengono messi a disposizione diversi script che consentono differenti modalità di *startup* del sistema. Inizialmente sarà necessario avviare il classificatore procedendo con il caricamento iniziale dei dati (`load_and_startup.sh`), mentre per ogni avvio successivo verrà eseguito lo script `startup.sh`. È prevista anche una modalità di testing che consente di eseguire un ciclo completo dell'intera applicazione (`load_and_singlerun.sh` oppure `singlerun.sh`). L'arresto del sistema è realizzato tramite lo script `shutdown.sh`.

Capitolo 4

Nuovo scenario operativo

Dopo uno studio preliminare del sistema di classificazione preesistente, rivolto con particolare attenzione alle funzionalità delle singole componenti e al modello di classificazione adottato, è stato possibile procedere ad un'analisi dei requisiti evidenziati dal nuovo sistema aziendale coinvolto nel progetto.

Nei capitoli successivi verranno descritte in modo dettagliato le scelte e le soluzioni che sono state adottate per soddisfare le esigenze aziendali, giustificandole illustrando i risultati dei test che sono stati svolti durante l'attività di tesi.

4.1 Azienda coinvolta

L'azienda coinvolta nell'attività di tesi opera nel settore dell'editoria, e in particolare si occupa di gestire una rivista diffusa in tutto il mondo. Offre inoltre diversi servizi, tra cui la vendita di libri e di altri prodotti, e svolge attività di pubbliche relazioni, che le impongono la gestione di un elevato numero di e-mail al giorno. Il sistema aziendale si appoggia su diversi siti web che consentono agli utenti di usufruire dei servizi offerti. Ciò comporta l'individuazione di un certo numero di caselle di posta elettronica a cui i clienti possono fare riferimento. Si tratta di 15 caselle ripartite in base alla lingua utilizzata e al servizio messo a disposizione. Le lingue supportate sono l'inglese, l'italiano, lo spagnolo, il portoghese, il francese e il tedesco.

I servizi proposti si distinguono in 3 temi principali che consistono in: relazioni con il pubblico (A), prodotti(B) e gestione profili (C), simile ai *Social Network*. Chiaramente le informazioni relative alla provenienza delle e-mail potrebbero essere utili ai fini della classificazione, come verrà descritto al termine del documento nella sezione riservata agli sviluppi futuri.

Nello specifico l'azienda ha a disposizione 2 caselle generiche, indipendenti da lingua e tema trattato, che in base alla statistiche aziendali rilevano un flusso di circa 27000 e-mail all'anno. Una casella è dedicata esclusiva-

mente al servizio di tipo C e raccoglie e-mail provenienti da ogni parte del mondo, per un totale all'anno, di 16500 e-mail. Per ognuno degli altri due temi, A e B, sono previste 6 caselle di posta, una per ogni lingua supportata. L'ammontare delle e-mail che coinvolgono la vendita di prodotti (B) è di circa 5000, mentre per il tema A, il totale sale a 16500 e-mail annuali.

Sommando questi valori si ottiene un ammontare medio di 65000 e-mail ricevute dall'azienda nell'arco di un anno. Chiaramente il flusso delle e-mail non è regolare nel tempo, ma varia in base a determinati periodi. Sarà decisamente basso durante il periodo estivo e avrà dei picchi in occasione di festività o altro. In media tuttavia, è stato possibile rilevare un flusso giornaliero che va da un minimo di 50 e-mail, fino ad un massimo di 600. Da questi dati emerge quindi la necessità di introdurre all'interno del sistema aziendale una componente che consenta di agevolare la gestione e il controllo del flusso di e-mail. L'idea è di riuscire a instradare le e-mail in ingresso, provenienti da varie caselle di posta, ai diversi operatori aziendali, a seconda delle loro competenze e abilità.

4.2 Tassonomia adottata

Il sistema di classificazione necessita di un set iniziale di dati, preclassificati sulla base di una specifica tassonomia. A tal proposito è stato richiesto all'azienda di definire una tassonomia delle categorie in cui suddividere i messaggi di posta elettronica. Le classi, identificate da valori numerici, corrispondono ai diversi servizi messi a disposizione, che all'interno del sistema aziendale vengono gestiti e controllati da operatori con competenze specifiche. In figura 4.1 viene riportata la tassonomia che è stata definita in una prima fase iniziale. La tassonomia è organizzata in modo gerarchico, la radice corrisponde all'identificativo ⓪⓪, ed per ogni classe figlia diretta della radice si ha un numero variabile di sottocategorie. Tra le classi di secondo livello è prevista una label comune a tutte le categorie di primo, in cui all'identificativo della macro classe viene aggiunto il suffisso 00; la presenza di questa etichetta consente di attribuire alle e-mail soltanto la categoria di primo livello, senza dover raffinare ulteriormente la classificazione, selezionando una specifica categoria di livello due. L'azienda ha fornito anche un'ulteriore suddivisione per comprendere meglio la semantica legata ad alcune categorie e per fissare diverse priorità. Le classi sono state suddivise in due aree, gestionale e relazionale, nel seguente modo:

- Gestionale:
  - classe ⓪④
  - classe ⓪⑤
  - classe ⓪③

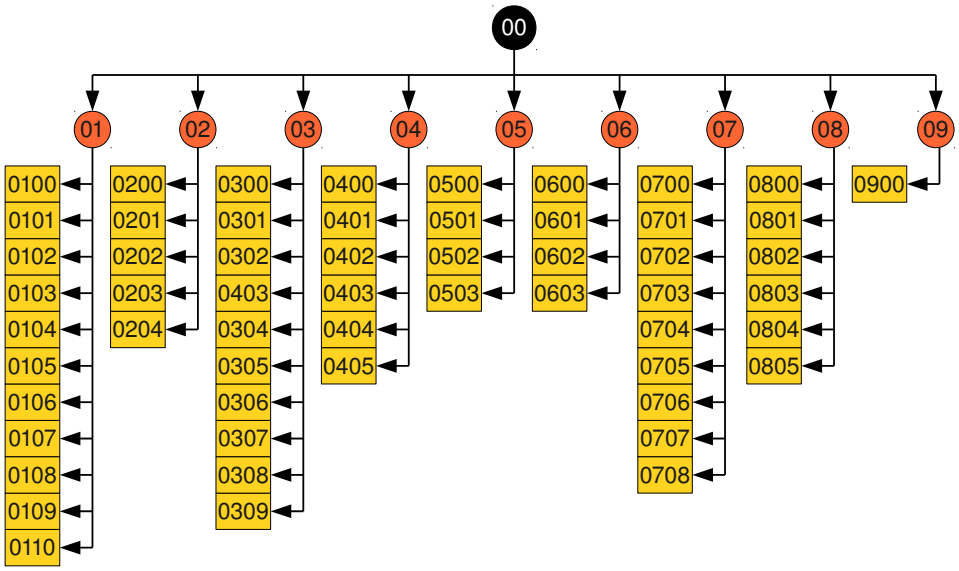

Figura 4.1: Tassonomia iniziale usata per la classificazione.

- Relazionale:
  - classe (06)
  - classe (02)
  - classe (08)
  - classe (07)
  - classe (01)

Analizzando in dettaglio il contenuto dei messaggi di posta preclassificati, e considerando anche le priorità evidenziate, sono emerse alcune problematiche legate all'ambiguità e a possibili sovrapposizioni tra alcune classi. Dal set di e-mail di esempio, fornite dall'azienda, sono stati selezionati alcuni casi ambigui, per cui non era evidente l'appartenenza ad una determinata categoria, e sono stati sottoposti ad ulteriori valutazioni. In particolare questo accade per le classi identificate dal valore (03) e (08). L'individuazione di queste due categorie risulta essere infatti particolarmente difficoltosa, tanto che, anche la decisione presa al riguardo da due esperti operatori aziendali, si è rivelata essere non concorde. Questo è dovuto al fatto che le due classi sono molto simili dal punto di vista del tema trattato, e si distinguono solamente per piccole sfumature, quasi soggettive, presenti nella composizione delle frasi; inoltre la categoria (08) può essere vista anche come una specializzazione della classe (03).

Sulla base di queste osservazioni si è deciso, in accordo con l'azienda, di procedere ad una modifica della tassonomia iniziale mirata a eliminare situazioni ambigue e a fissare delle divisioni nette tra i diversi temi/servizi associati alle varie categorie. La variazione principale riguarda l'unione delle due classi di primo livello, (03) e (08), in un'unica classe (03); in realtà l'identificativo (08) ha assunto il ruolo di sottoclasse nella categoria (03). Altra piccola modifica ha coinvolto la sottoclasse [0304], che avendo in realtà la stessa semantica della categoria di secondo livello [0502] è stata inglobata in quest'ultima. La versione definitiva della tassonomia viene riportata in figura 4.2.

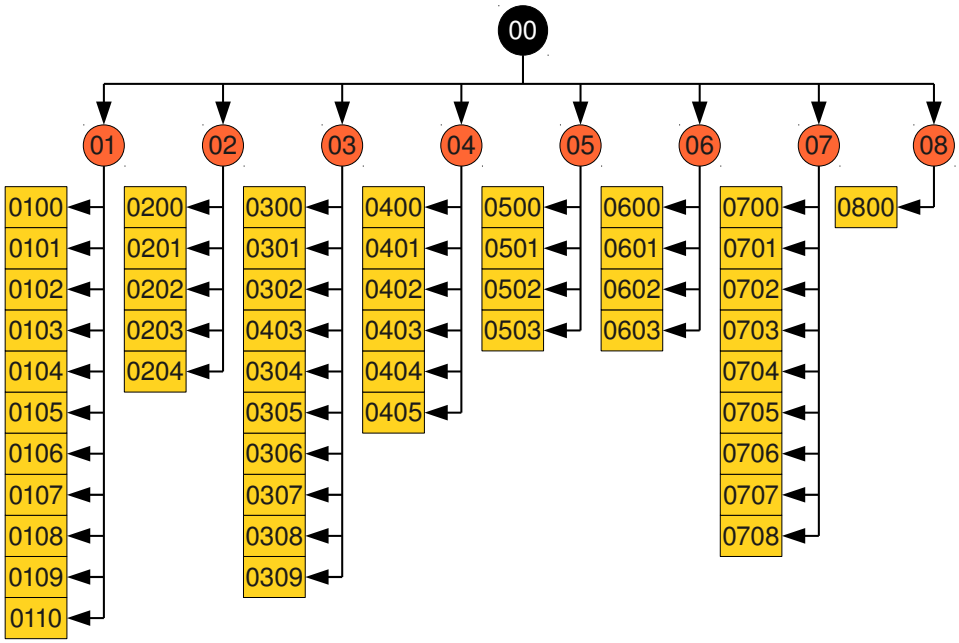

Figura 4.2: Tassonomia definitiva usata per la classificazione.

Sulla base di questa tassonomia definitiva è possibile ora riportare una stima della distribuzione dei messaggi di posta elettronica del TR all'interno del set delle categorie di primo livello presenti. Dalla tabella 4.1 appare subito evidente che le e-mail non sono distribuite in modo uniforme tra le classi, e infatti alcune risultano essere molto più popolose rispetto ad altre. La presenza di classe sblilanciate comporta alcuni problemi in fase di classificazione.

ID della classe	% e-mail
01	39,7
02	2,7
03	32,8
04	6,3
05	10,4
06	2,3
07	6,1
08	0,1

Tabella 4.1: Stima della distribuzione delle e-mail del TR nelle diverse categorie di primo livello.

Fin da subito è possibile notare che la classe ⑭, a cui l'azienda ha associato il valore di priorità più alto, non è così numerosa, e questo può condurre a serie difficoltà nel distinguerla dalle altre. Nel capitolo 6 verranno esposte in dettaglio le soluzioni che sono state adottate all'interno del modulo di classificazione per risolvere questo problema che coinvolge classi sbilanciate.

4.3 Descrizione requisiti

All'inizio dell'attività di tesi è stato necessario definire i nuovi requisiti che il sistema di classificazione doveva garantire per soddisfare le esigenze del nuovo sistema aziendale. Fin da subito l'azienda ha richiesto la possibilità di gestire e controllare le e-mail manipolate dal sistema di classificazione, associando ad ognuna un particolare identificativo utilizzato all'interno del flusso aziendale. Tutte le e-mail trattate, oltre ai due campi (UID e MID) già presenti, dovranno essere identificate attraverso un *id_activity*, che corrisponde ad un codice alfanumerico fornito direttamente dall'azienda. Sebbene la tassonomia sia organizzata in livelli gerarchici, il sistema di classificazione dovrà in un primo momento individuare soltanto le categorie di primo livello, e solo in una fase successiva la classificazione verrà estesa alle sottocategorie. L'obiettivo principale è chiaramente quello di garantire le migliori prestazioni del sistema di classificazione, ossia, come richiesto dall'azienda, fare in modo che almeno l'80% delle e-mail venga classificato correttamente. Una e-mail è classificata in modo corretto se all'interno del *ranking* proposto come risultato di classificazione, la *label*, corrispondente alla sua classe di appartenenza, compare in posizione 0. In base alle esigenze del sistema aziendale coinvolto è emersa anche la necessità di procedere ad una modifica della gestione dei risultati di classificazione e della lettura del *feedback*. Il sistema deve avvicinarsi i più possibile al ruolo di un *web service* in grado di fornire i risultati di classificazione per le e-mail *on-demand*.

Riassumendo di seguito vengono riportati i requisiti individuati, distinguendoli per categoria e tra obbligatori [O] e desiderabili [D].

Requisiti funzionali [F]

- **Obbligatori**

  [FO-01] Per utilizzare il sistema di classificazione è necessario fornire una tassonomia e un set di dati preclassificati.

  [FO-02] Il sistema classifica i documenti considerando le classi di primo livello della tassonomia fornita.

  [FO-03] Un documento è associato a una sola classe.

  [FO-04] La relazione tra le classi della tassonomia è di tipo inclusivo tra una classe padre e le proprie classi figlie.

  [FO-05] I documenti classificati dal sistema sono documenti testuali.

  [FO-06] I documenti devono essere trasformati nel formato richiesto dall'algoritmo di apprendimento.

  [FO-07] Il sistema apprende il modello eseguendo il *training* sui dati supervisionati.

  [FO-08] Il sistema salva il modello appreso per la successiva fase di classificazione

  [FO-09] L'utente sottomette al sistema un documento da classificare.

  [FO-10] Il sistema è in grado di classificare i documenti sottomessi dall'utente.

  [FO-11] Il sistema fornisce un ranking rappresentante il risultato di classificazione.

  [FO-12] Il *ranking* fornito per un e-mail presenta le categorie in ordine descrescente di probabilità.

  [FO-13] Il *ranking* contiene solo le prime 5 categorie con valore di confidenza più alto.

  [FO-14] L'utente deve fornire un *feedback* al sistema per le e-mail classificate.

  [FO-15] Il sistema preleva il *feedback* di ogni singola e-mail.

  [FO-16] Le nuove informazioni supervisionate acquisite vengono integrate nel sistema.

  [FO-17] Il sistema apprende un nuovo modello aggiungendo all'insieme di dati iniziali le nuove informazioni.

- **Desiderabili**

  [FD-01] Il sistema può estendere la classificazione alle categorie di secondo livello.

  [FD-02] Il *ranking* dovrebbe contenere tutte le categorie e le sottocategorie della tassonomia ordinate in modo decrescente per probabilità.

  [FD-03] Il sistema dovrebbe eseguire l'apprendimento in modo incrementale.

  [FD-04] L'utente può aggiornare la tassonomia in modo dinamico.

  [FD-05] Il sistema può gestire l'introduzione di regole associative fissate a priori.

  [FD-06] Il sistema di classificazione può supportare molteplici lingue.

  [FD-07] Il sistema può essere in grado di associare ad un documento un insieme di categorie con cardinalità maggiore di 1.

  [FD-08] Nel processo di classificazione potrebbero essere incluse informazioni relative alle fonti originali delle e-mail.

Requisiti di qualità [Q]

Obbligatori

[QO-01] Il *ranking* deve presentare almeno l'80% delle e-mail classificate correttamente in prima posizione.

[QO-02] Il sistema deve attribuire maggiore importanza all'individuazione corretta delle classi con alta priorità.

Desiderabili

[QD-01] Il sistema può prevedere un modulo per eliminare le informazioni superflue dal testo del documento.

[QD-02] Il sistema dovrebbe adottare un modulo per la correzione automatica degli errori ortografici presenti nel testo.

[QD-03] Il sistema dovrebbe prevedere una componente per selezionare i termini più significativi nel testo.

[QD-04] Introduzione di una componente per individuare e evidenziare termini simili o correlati.

Requisiti architetturali e di interfacciamento[A]

- **Obbligatori**

  [AO-01] Il sistema di classificazione è integrato al sistema aziendale.

  [AO-02] I documenti devono essere mantenuti in una forma indicizzata all'interno del sistema di classificazione.

  [AO-03] All'interno dei documenti indicizzati la presenza di dati sensibili deve essere ridotta al minimo.

  [AO-04] La classificazione deve essere fornita all'utente in un formato comprensibile.

  [AO-05] Durante il *loading* iniziale dei dati le e-mail sono identificate dal *id_activity*.

  [AO-06] Il file con il risultato della classificazione specifica l'*id_activity* del messaggio di posta.

  [AO-07] La lettura del *feedback* di una e-mail avviene specificandone l'*id_activity* associato.

- **Desiderabili**

  [AD-01] Il sistema può integrare la gestione delle diverse fonti da cui provengono i documenti da classificare.

  [AD-02] Il sistema non necessita di un DB Mysql dedicato, ma memorizza e gestisce i dati direttamente attraverso il DB aziendale.

4.4 Revisione del sistema

Una volta individuati i requisiti, legati al nuovo sistema aziendale coinvolto, è stato possibile definire un piano delle attività programmate per procedere all'adattamento e alla realizzazione di un sistema che sia in grado di soddisfare e rispecchiare le nuove esigenze dell'azienda coinvolta.

1. Analisi del sistema aziendale e delle modalità di scambio informazioni con il sistema di classificazione.

2. Definizione di un nuovo protocollo di comunicazione con il sistema aziendale.

3. Modifica modalità pubblicazione dei risultati di classificazione.

4. Realizzazione di un nuovo modulo per la lettura del *feedback* conforme al nuovo protocollo.

5. Valutazione iniziale delle prestazioni del classificatore.
6. Analisi dettagliata del contenuto dei messaggi di posta elettronica.
7. Studio approfondito delle possibili tecniche di *pre-processing* e indicizzazione applicabili.
8. Introduzione di nuove componenti per il raffinamento della rappresentazione delle e-mail.
9. Valutazione di una possibile integrazione di modelli di classificazione più sofisticati e performanti.
10. Analisi e valutazione delle prestazioni del classificatore al variare di particolari condizioni e specifici parametri.
11. Studio di possibili sviluppi futuri e estensioni del sistema di classificazione.

4.4.1 Interfacciamento con il nuovo sistema aziendale

In accordo con l'azienda si è deciso di installare il classificatore all'interno della loro struttura, in modo da evitare dispersione di informazioni e dati sensibili. É stato quindi predisposto l'ambiente tecnologico, che prevede l'adozione di un DB Mysql per la memorizzazione dei dati e di un *web server*, Apache Tomcat 5.5, per la gestione delle *servlet*. Di seguito vengono illustrate in dettaglio le modifiche che sono state apportate per soddisfare i nuovi requisiti architetturali, per adattare il sistema di classificazione esistente al nuovo sistema aziendale coinvolto.

Gestione identificativo aziendale

Nel nuovo sistema di classificazione è stato necessario modificare la modalità di identificazione delle e-mail trattate. All'interno del *body* delle e-mail prelevate dalla casella di posta dedicata al popolamento iniziale della base di conoscenza, all'azienda è stato chiesto di riportare l'*id_activity* corrispondente al messaggio, seguito poi dal codice numerico associato alla classe corretta di appartenenza della e-mail. La procedura viene mantenuta anche per le e-mail che devono essere classificate, e quindi nella prima riga del *body* sarà indicato l'*id_activity* associato al messaggio. La prima operazione che viene eseguita, prima di manipolare le e-mail, sia quelle iniziali che quelle a regime, è l'estrazione dell'*id_activity* associato al messaggio. È stato necessario modificare il modulo di caricamento dei dati e quello di estrazione delle e-mail da classificare, aggiungendo un *parser* che fosse in grado di estrapolare dal testo l'*id_activity* riportato. Così facendo questa informazione può essere inserita all'interno della struttura definita e mantenuta per tutti i dati supervisionati elaborati dal sistema di classificazione.

Per quanto riguarda invece le e-mail che vengono classificate, l'*id_activity* costituisce un'informazione essenziale per la pubblicazione dei risultati di classificazione.

Pubblicazione risultati classificazione

La comunicazione tra il classificatore e il sistema aziendale avviene sempre tramite scambio di file XML via HTTP. È stato necessario modificare la struttura del file XML che contiene i risultati di classificazione. La gestione delle e-mail non avviene più tramite lotti progressivi; viene prodotto un file XML per ogni e-mail classificata, riportando all'interno l'*id_activity* corrispondente, e inserendolo anche nel nome del file stesso. Questo ha reso il sistema di classificazione più simile a un *Web Service*, in cui è possibile richiedere la presentazione dei risultati di classificazione di una data e-mail, specificandone l'*id_activity*. In una prima fase all'interno del file pubblicato vengono riportate solamente le prime 5 categorie di primo livello suggerite dal classificatore, ordinate per valore decrescente di probabilità. Di seguito viene riportato un esempio di file di presentazione dei risultati.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<classificatore>
   <data>DD/MM/YYYY HH:MM:SS</data>
   <e-mail>
     <idActivity>id_activity dell'azienda</idActivity>
     <uid>uid assegnato da protocollo IMAP</uid>
     <mid>message ID assegnato da protocollo IMAP</mid>
     <coda>
       <id1>"id classificazione 1 livello I</id1>
       <id2>"id classificazione 2 livello I</id2>
       <id3>"id classificazione 3 livello I</id3>
       <id4>"id classificazione 4 livello I</id4>
       <id5>"id classificazione 5 livello I</id5>
     </coda>
   </e-mail>
</classificatore>
```

Quando nella fase finale dell'attività di tesi, la classificazione è stata estesa anche alle sottocategorie, è stato necessario modificare ulteriormente il file di pubblicazione dei risultati di classificazione, come riportato di seguito.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<classificatore>
   <data>DD/MM/YYYY HH:MM:SS</data>
   <e-mail>
     <idActivity>id_activity dell'azienda</idActivity>
```

```
      <uid>uid assegnato da protocollo IMAP</uid>
      <mid>message ID assegnato da protocollo IMAP</mid>
      <coda>
        <id1 value = "id classificazione 1 livello I">
           <subId1>id classificazione I-II livello</subId1>
           <subId2>id classificazione I-II livello</subId2>
           <subIdn>id classificazione I-II livello</subIdn>
        </id1>
        <id2 value = "id classificazione 2 livello I">
           <subId1>id classificazione I-II livello</subId1>
           <subId2>id classificazione I-II livello</subId2>
           <subIdn>id classificazione I-II livello</subIdn>
        </id2>
        <idN value = "id classificazione N di livello I">
           <subId1>id classificazione I-II livello</subId1>
           <subId2>id classificazione I-II livello</subId2>
           <subIdn>id classificazione I-II livello</subIdn>
        </idN>
      </coda>
    </e-mail>
</classificatore>
```

All'interno del file viene presentata la classificazione proposta per la data e-mail inserendo tutte le classi presenti nella tassonomia (N) con le relative sottoclassi (n), disponendole in ordine decrescente di confidenza, attribuita dal classificatore. Questa scelta è stata compiuta in accordo con l'azienda che ha manifestato l'esigenza di avere a disposizione tutte le informazioni possibili, per procedere poi ad una selezione dei dati rilevanti ai fini aziendali.

Per quanto riguarda invece la presentazione dei risultati, è stato necessario modificare la *servlet* su cui il sistema si appoggia. In seguito ad una *request* viene reso disponibile il file XML corrispondente alla e-mail specificata come parametro:

$$http://<indirizzo\ servizio>?id=<id\_activity>$$

Nel caso in cui la e-mail richiesta tramite l'*id_activity* non è disponibile, viene restituito il codice HTTP 404:*Not Found*.

4.4.2 Gestione della lettura del *feedback*

Il differente protocollo di comunicazione adottato nel nuovo contesto applicativo ha richiesto una modifica sostanziale della modalità di lettura del *feedback* fornito dall'azienda. Innanzitutto, la *servlet* di lettura dei risultati di *feedback* è stata implementata in modo tale da mettere a disposizione tre differenti modalità di *request* a seconda delle informazioni che devono essere

ottenute. Una prima modalità è stata sviluppata per ottenere la lista di file, corrispondenti al *feedback* delle e-mail, messi a disposizione dall'azienda in un'apposita cartella, fissata come parametro nel file `web.xml` della *servlet*. Chiaramente è necessario un meccanismo che permetta la distinzione tra i file già letti, e integrati nel sistema, e quelli ancora da leggere. Si è deciso quindi di modificare i nomi dei file XML, dopo la lettura e l'inserimento nella cartella dedicata nel classificatore, aggiungendo il suffisso `DONE`.
Invocando la *servlet* come di seguito,

$$http://<indirizzo\ servizio>?listFiles=true \tag{4.1}$$

si avrà in risposta la lista completa di tutti i file XML di *feedback* che non sono marcati come `DONE` e che devono essere quindi integrati nel sistema; il file XML contenente la lista verrà salvato nella sottocartella `temp` della `Home` del classificatore.

Utilizzando il parametro *getFile* nell'invocazione della *servlet* è possibile richiedere un particolare file di *feedback*:

$$http://<indirizzo\ servizio>?getFile=<id\_activity> \tag{4.2}$$

Nel caso in cui la e-mail corrispondente all'*id_activity* richiesto non sia disponibile, viene restituito il codice HTTP di errore 404:*Not Found*.

Infine l'ultima modalità prevista consente di modificare il nome del file XML, specificato attraverso il suo *id_activity*, introducendo il suffisso `DONE` per segnalare che la lettura del file è stata completata con successo.

$$http://<indirizzo\ servizio>?setDone=<id\_activity> \tag{4.3}$$

All'interno del classificatore, la componente di gestione del *feedback* è stata completamente rivisitata per adattarsi al nuovo protocollo di scambio. In particolare per fare in modo che tutti i file di *feedback* disponibili vengano letti in modo corretto senza perdite, vengono eseguiti i seguenti passi.

1. Invocazione della *servlet* (4.1) per ottenere la lista dei file XML di *feedback* forniti dall'azienda non ancora letti. Viene salvato un file XML (`fileList.xml`) nella sottodirectory `temp` della `Home` del classificatore che contiene la lista degli *id_activity* corrispondenti alle e-mail da integrare nel sistema.

2. Per ogni *id_activity* presente nel file `fileList.xml` viene invocata la *servlet* (4.2) per la lettura dei file specifici di *feedback*. Viene salvata una copia di ogni file, dopo l'esecuzione del *parsing*, nella sottocartella `Feedback` della `Home` del classificatore.

3. Se l'operazione di salvataggio del file XML di *feedback* è completata con successo, viene invocato la *servlet* (4.3), che consente la modifica

del nome del file salvato, con l'aggiunta del suffisso `DONE`. Così facendo ad ogni successiva lettura dei file di *feedback* si evita di rileggere file per cui già stato effettuato il *parsing* e che sono già stati integrati nel sistema di classificazione. L'operazione di `setDONE` viene eseguita per tutti file con *id_activity* contenuto nella lista di file (*not* `DONE`) generata nella prima fase.

4.5 Realizzazione dell'ambiente di test

Per poter procedere con la valutazione delle prestazioni del classificatore è stato necessario prima di tutto predisporre un ambiente adatto. A questo scopo, è stato realizzato una sorta di clone del progetto originale, chiaramente modificato a seconda delle esigenze. All'interno dell'ambiente di test, l'unica casella di posta elettronica a cui si fa riferimento è la casella predisposta per il caricamento iniziale dei dati supervisionati. In un primo momento le e-mail sono state ripartite in due sottoinsiemi rispettivamente pari al 75% e 25% del totale dei dati a disposizione. Il 75% delle e-mail durante l'esecuzione del test viene usato come *Training Set* e quindi per l'apprendimento del modello, il restante 25% costituisce l'insieme di dati, su cui verificare la bontà dei risultati di classificazione. Per avere un valore medio corrispondente alle prestazioni del classificatore sono state eseguite 10 iterazioni successive in cui in ognuna di esse veniva eseguito il ciclo completo di apprendimento e classificazione, basandosi ogni volta su sottoinsiemi selezionati casualmente dall'insieme di dati inziali. Successivamente però, data la numerosità dei parametri da valutare, è stato necessario introdurre una nuova modalità di testing basata sull'approccio *k-fold cross validation* ($k = 10$) con alcuni script, per automatizzare i vari controlli. Per ogni *fold* venivano valutate le prestazioni, e il valore globale medio era calcolato ricorrendo alla media dei valori delle varie iterazioni.

POSIZIONE	*ERROR-SCORE*
0	0
1	1
2	2
3	3
4	4
altre	5

Tabella 4.2: Valutazione *Error-Score* in base alla posizione che la classificazione corretta assume nel *ranking*.

Per valutare le prestazioni del classificatore, ad ogni nuova e-mail classificata viene assegnato un *Error-Score*(ES), un intero che incrementa in base alla posizione assunta dalla classificazione corretta nel *ranking* proposto. Pertanto, se la classificazione corretta compare in posizione 0, l'*Error-Score* assegnato sarà pari a 0, se compare in posizione 1 l'*Error-Score* sarà pari a 1, e così via. Se la classificazione corretta non compare nel *ranking* proposto, che, come descritto in precedenza, riporta solamente le prime 5 label con probabilità più alta, l'*Error-Score* assumerà il valore massimo, pari a 5 (vedi tabella 4.2). Per illustrare i risultati dei test effettuati, oltre al valore di *Error-Score* calcolato, viene anche riportata la distribuzione media delle percentuali di classificazioni corrette che il classificatore presenta nel *ranking*

finale. Vengono specificate le percentuali rilevate per le prime 5 posizioni più un valore aggiuntivo che corrisponde alla percentuale di classificazioni corrette inserite dal classificatore oltre la quinta posizione. Chiaramente più basso è il valore di *Error-Score* associato al *ranking* proposto, migliore sarà il risultato di classificazione; questo corrisponde all'avere la più alta percentuale possibile di classificazioni corrette nelle prime posizioni.

Nel corso dell'attività di tesi i dati disponibili per i test sono stati aggiunti in modo progressivo in base alla disponibilità dell'azienda coinvolta (500 → 1000 → 2000 e-mail complessive). Pertanto nel riportare i diversi risultati conseguiti, verrà specificato anche l'ammontare dei dati disponibili, ripartiti tra *Training Set* e *TestSet*.

Capitolo 5

Raffinamento rappresentazione degli esempi

Prima di procedere con l'analisi vera e propria delle operazioni di classificazione, è stato necessario uno studio dettagliato del testo contenuto nelle e-mail. Da questa analisi sono emerse alcune particolarità che hanno condotto a determinate scelte mirate a un raffinamento della fase di *pre-processing* e di indicizzazione dei documenti.

5.1 Valutazione iniziale risultati classificazione

Dopo aver apportato al sistema di classificazione le modifiche necessarie per adattarlo al nuovo protocollo di comunicazione con il sistema aziendale, è stato possibile procedere con il primo test per valutare le prestazioni del classificatore preesistente nel nuovo dominio. Inizialmente sono stati rilevati dei risultati ampiamente sotto le aspettative. Infatti solo per il 24% delle e-mail di *Test Set*, il classificatore proponeva la classificazione corretta in posizione 0 (vedi Fig. 5.1).

Si è resa necessaria un'analisi dettagliata della fase di *training* e apprendimento del modello all'interno del sistema. Particolare attenzione è stata rivolta alla fase di costruzione del dizionario utilizzato per la rappresentazione degli esempi. Sono state valutate le probabilità utilizzate dal modello bayesiano, associate ai singoli termini delle varie classi. Ciò ha permesso di individuare un *bug* nel sistema: durante la fase di classificazione delle e-mail, anzichè adottare il dizionario costruito in fase di *training*, veniva generato un nuovo dizionario, perdendo quindi la corrispondenza tra gli indici dei termini degli esempi preclassificati e quelli delle e-mail da classificare.

Una volta individuato e corretto il *bug* del sistema preesistente è stato possibile procedere con la revisione dettagliata delle fasi che precedono

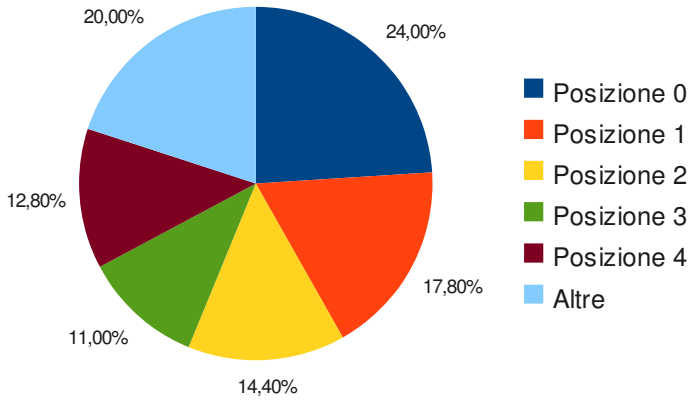

Figura 5.1: Test iniziale: risultati classificazione con il classificatore preesistente (375 + 125).

la classificazione vera e propria. Le attività che sono state svolte si sono focalizzate su due fasi principali:

- **Fase di *pre-processing***
- **Fase di indicizzazione**

5.2 Fase di *pre-processing*

Il *pre-processing* viene eseguito attraverso i metodi di una classe già presente nel progetto iniziale, `org.ictlab.cluster4.indexing.EmailProcessor`, che si occupano di manipolare e di analizzare i singoli *token* presenti nel testo dell'e-mail, e di restituire al chiamante la sequenza di *token* risultante. In figura 5.2 viene descritto il flusso di operazioni che venivano eseguite nel sistema preesistente.

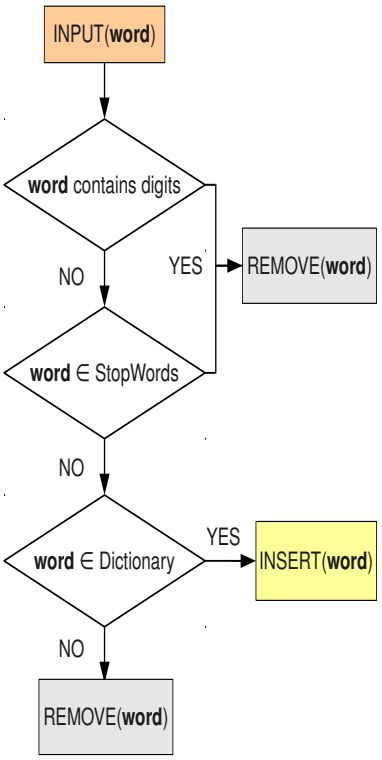

Figura 5.2: Sistema preesistente: sequenza operazioni in fase di *pre-processing*.

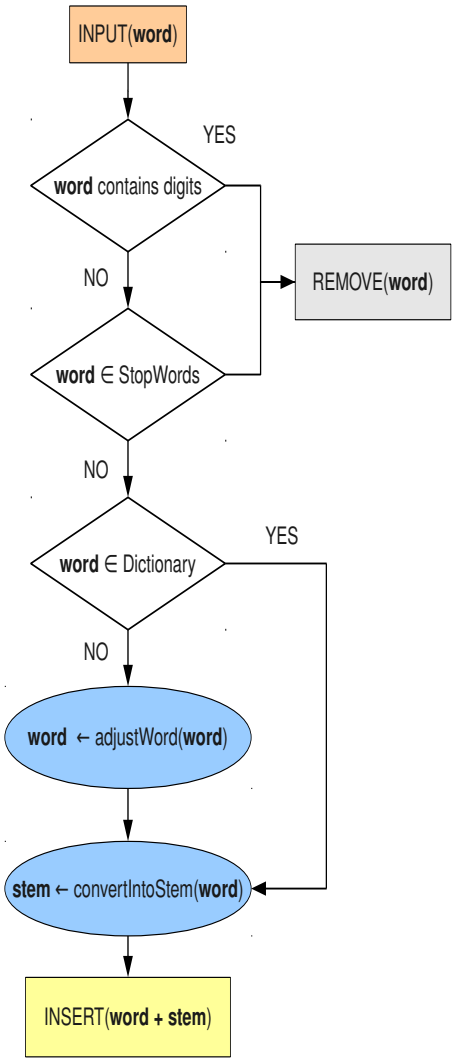

Figura 5.3: Nuovo sistema: sequenza operazioni in fase di *pre-processing*.

Analizzando in dettaglio il contenuto dei messaggi di posta elettronica sono emersi alcuni elementi che hanno reso necessario un raffinamento delle operazioni presenti e nuovi interventi, al fine di ottenere una rappresentazione che fosse il più possibile significativa per la successiva fase di classificazione. Il nuovo flusso di operazioni che vengono eseguite in fase di *pre-processing* viene riportato in figura 5.3.

Eliminazione informazioni superflue

I dati a disposizione hanno confermato la presenza di alcune parti di testo che rappresentano solo rumore inutile ai fini della classificazione. Alcune

e-mail, ad esempio, seguono una precisa regola di formattazione dovuta alla fonte originale da cui provengono (Es. form dedicati). Per estrarre solo il testo del messaggio scritto dall'utente è stato realizzato un *parser ad-hoc* in grado di scartare tutte le altre informazioni superflue. In modo analogo sono state eliminate dal testo anche tutte le informazioni riportate in seguito ad un eventuale *forwarding* della e-mail.

Modifica lista *stop words*

Nel sistema precedente era già presente una lista di *stop words* per rimuovere quei termini che non contribuiscono all'identificazione della classificazione corretta; attualmente a questa è aggiunta un'ulteriore lista di tutti i possibili tag html da eliminare dalla rappresentazione del testo. Trattandosi infatti di messaggi di posta elettronica all'interno del *body* si ritrovano diversi tag html relativi alla struttura e alla formattazione del contenuto. Alcuni termini tuttavia, che generalmente rientrano nella categoria di *stop words*, (Es. "non") si sono dimostrati essere dei discriminanti importanti per l'individuazione di alcune categorie; per questo motivo sono state rimosse dalla lista per contribuire a formare la rappresentazione del testo in fase di indicizzazione.

Dizionario adottato

Il dizionario in precedenza adottato, di circa 35.000 termini si è rivelato piuttosto povero per il nuovo contesto: molte parole presenti nelle e-mail non trovavano infatti corrispondenza e venivano eliminate dalla rappresentazione. Per questo motivo è stato introdotto un nuovo set più ricco di termini (circa 280.000) in grado di garantire una copertura maggiore.

Nelle sezioni successive vengono descritti in dettaglio i nuovi interventi eseguiti nella fase di *pre-processing*, giustificando le scelte fatte con i risultati dei diversi test realizzati.

5.3 Correzione ortografica

Durante la fase di analisi del testo delle e-mail da classificare è stata riscontrata una alta percentuale di errori ortografici, riconducibili molto spesso a errori di battitura. Chiaramente la presenza di questi termini influenza negativamente la procedura di classificazione; infatti le parole che non trovano corrispondenza con i termini presenti nel dizionario vengono direttamente eliminate dalla rappresentazione. Così facendo si rischia però di rimuovere diversi *token* utili ai fini della classificazione.

Per risolvere questo problema si è deciso di ricorrere a una particolare classe, messa a disposizione dalla libreria Lucene, `Spellchecker`, che si occu-

pa di suggerire una versione corretta della parola che non ha corrispondenza con alcun temine del dizionario di riferimento.

Modulo introdotto: `Lucene Spellchecker`

All'interno del modulo che esegue il *pre-processing* dei dati, è stata aggiunta la classe wrapper `TokenChecker`, che costruendo un oggetto della classe `Spellchecker` è in grado di invocare la funzione principale che fornisce il suggerimento alla parola data. Per procedere è necessario prima di tutto indicare il riferimento al dizionario utilizzato e mettere a disposizione una cartella dedicata all'oggetto `Spellchecker` per memorizzare al suo interno tutti i file generati automaticamente.
Il metodo `TokenChecker::adjustWord(String)` prende in input una stringa corrispondente al termine affetto da errore e ritorna in output un array contenente i suggerimenti disponibili per la stringa passata come parametro. Il numero di suggerimenti è un valore che può essere modificato in base alle esigenze. In questo caso vengono restituiti 5 termini di cui però viene considerato solo il primo proposto, che andrà poi a sostituire direttamente la parola "scorretta" nel testo della e-mail, senza ulteriori controlli. A questo punto risulta interessante capire quali scelte vengano compiute per restituire i suggerimenti alternativi alla parola data. L'idea che sta alla base è di fornire dei termini che siano il più possibile simili a quello dato in input. Al riguardo esistono diverse metriche che valutano la similarità tra termini; in questo caso è stata mantenuta la misura di default (distanza di Levenstein). La distanza di Levenshtein tra due stringhe A e B è il numero minimo di modifiche elementari che consentono di trasformare A in B. Per modifica elementare si intende:

- la cancellazione di un carattere,
- la sostituzione di un carattere con un altro, o
- l'inserimento di un carattere.

La distanza di Levenshtein ha alcuni semplici limiti superiori ed inferiori:

- è almeno la differenza fra le lunghezze delle due stringhe;
- è 0 se e solo se le due stringhe sono identiche;
- se le lunghezze delle due stringhe sono uguali, la distanza di Levenshtein non supera la distanza di Hamming, cioè pari alla lunghezza delle stringhe;
- il limite superiore è pari alla lunghezza della stringa più lunga.

Il grafico in figura 5.4 descrive i risultati dei test eseguiti su 500 e-mail disponibili, dopo la correzione del *bug* e l'introduzione della componente di correzione ortografica. Dalla distribuzione delle percentuali nel *ranking* appare evidente un netto miglioramento rispetto al test iniziale.

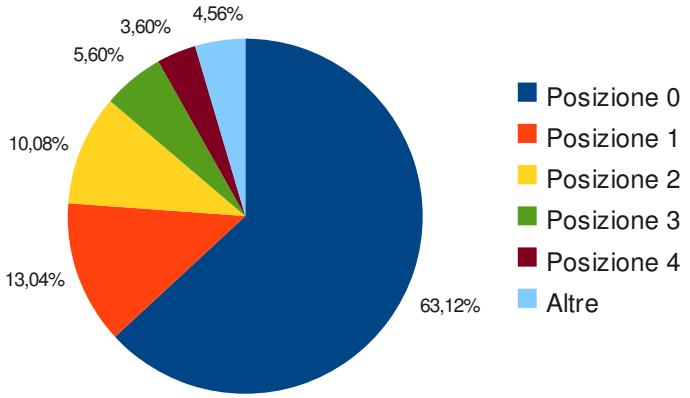

Figura 5.4: Risultati dei test (375 + 125) dopo la correzione del *bug* (*ES*: 0,872).

È bene precisare che la correzione ortografica automatica delle parole che non trovano corrispondenza nel dizionario, non sempre fornisce i risultati attesi. Si possono verificare infatti sostituzioni scorrette e quindi casi in cui il primo termine proposto, come suggerimento per la parola scorretta, non corrisponde alla soluzione esatta. Attualmente il sistema effettua la sostituzione in ogni caso, senza ulteriori controlli. In futuro si potrebbe pensare ad un ulteriore raffinamento di questa procedura di correzione automatica in modo da verificare che la sostituzione avvenga correttamente.

5.4 Stemming

Come descritto nelle sezioni introduttive lo *stemming* rappresenta una tecnica di manipolazione dei dati che consente di individuare le radici delle parole presenti nel testo.

Modulo introdotto: `Lucene Snowball`

Nella libreria Lucene è stato individuato un modulo che implementa le funzionalità per lo *stemming*. All'interno del modulo di *pre-processing* del progetto è stata aggiunta la classe `TokenStemmer` che costruisce un oggetto `ItalianStemmer` presente nel package `org.tartarus.snowball.ext`. Il metodo `TokenStemmer::convertIntoStem(String)` invoca al suo interno la funzione `ItalianStemmer::stem()` che ritorna in output la radice della parola passata come parametro dal chiamante. Per verificare l'effettivo valore aggiunto legato all'adozione del modulo di *stemming* sono stati realizzati due diversi test.

Nel primo test i termini presenti nel testo della e-mail vengono sostituiti con le rispettive radici. Dal grafico in figura 5.5 emergono dei risultati peggiori rispetto ai precedenti e questo ha portato a considerare l'ulteriore possibilità di affiancare alle parole le radici, senza eliminare i termini originali dal testo. In figura 5.6 vengono riportati i risultati ottenuti con

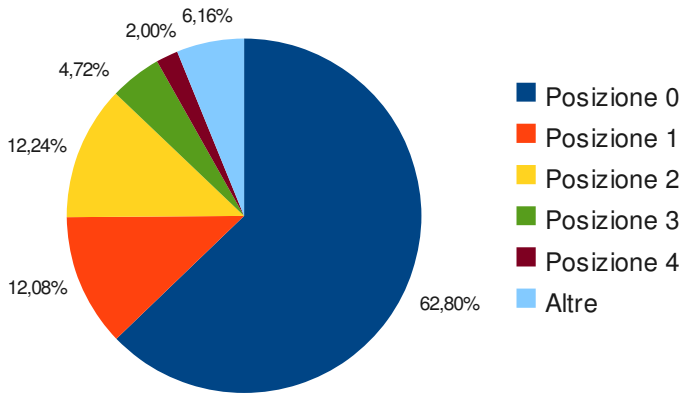

Figura 5.5: Risultati dei test (375 + 125) sostituendo ogni termine con la propria radice (*ES*: 0,895).

l'aggiunta delle radici dei termini. Dai valori riportati si riscontra un leggero miglioramento, riconducibile al fatto che aggiungendo gli *stem* nel testo, parole simili o con origine comune vengono mappate nella stessa radice, incrementando la frequenza di specifici termini. Le parole originali vengono mantenute in modo tale da non perdere termini che rappresentano delle par-

ticolari caratteristiche che contraddistinguono le e-mail appartenenti a classi diverse.

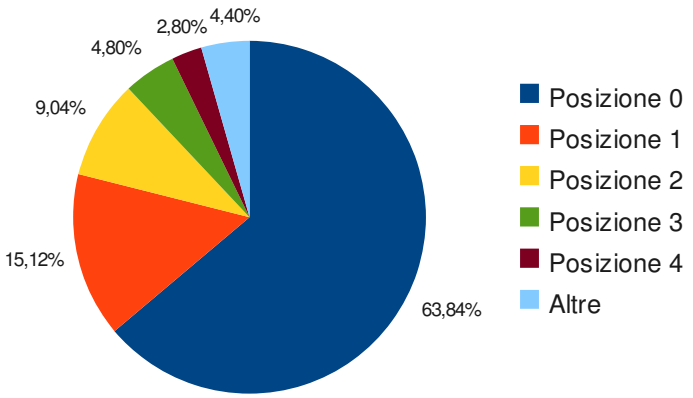

Figura 5.6: Risultati dei test (375 + 125) affiancando ad ogni termine la propria radice (*ES*: 0,718).

Lo *stemming* può apportare sia vantaggi che svantaggi. Se da una parte infatti riconduce parole correlate alla stessa radice, i sinonimi non sono riconosciuti e di conseguenza in alcuni casi, termini con lo stesso significato hanno radici differenti. La lingua italiana inoltre risulta essere particolarmente difficile da gestire rispetto quella inglese; questo è dovuto alla presenza di diversi termini irregolari. `Snowball` ha messo a disposizione una procedura che si basa sull'algoritmo di Porter per lo *stemming* inglese adattandola però alla lingua italiana.

5.5 Incremento dei dati a disposizione

A questo punto è stato rinforzato il set di dati a disposizione, ed è stato possibile procedere con i test con 1000 e-mail, 750 per il *Training Set* e le restanti 250 come *Test Set*. È stato ripetuto l'ultimo test, per determinare l'entità del miglioramento che ha apportato l'aggiunta di nuovi dati; in figura 5.7 vengono riportati i risultati conseguiti. Per analizzare la capacità del classificatore di individuare in modo corretto le singole classi, è stata costruita una matrice di confusione. Si tratta di una matrice che consente

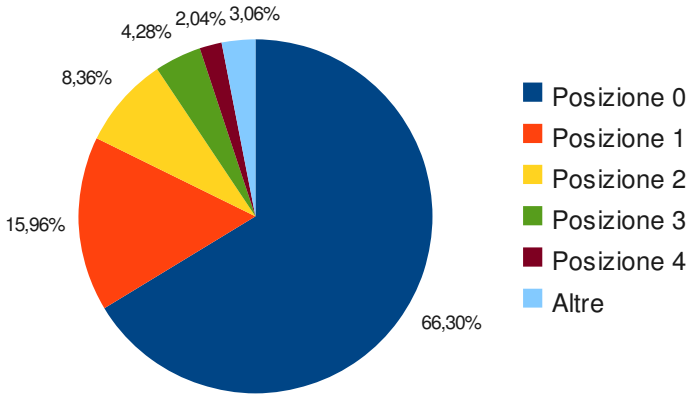

Figura 5.7: Risultati dei test (750 + 250) affiancando ad ogni termine la propria radice (*ES*: 0,715).

di mettere in evidenza eventuali relazioni che si instaurano tra le classi, e la frequenza con cui il classificatore esegue stime accurate. Le righe corrispondono alle classi corrette, e le colonne alle classi predette dal modello di classificazione. L'elemento della matrice m_{ij} è un valore percentuale che corrisponde alle e-mail di classe corretta i a cui è stata assegnata in prima posizione nel *ranking*, la classse j. Dalla figura 5.8 emerge in modo chiaro che vi sono alcune difficoltà nel distinguere correttamente alcune coppie di classi. In particolare ciò succede per le classi a cui sono associati gli identificativi ⑬ e ⑱. Questa situazione è riconducibile al fatto che le due classi sono decisamente simili e sovrapponibili, tanto che la scelta di associare una e-mail all'una o all'altra categoria può essere spesso soggettiva.

In accordo con l'azienda, come già illustrato nel capitolo precedente, si è deciso di procedere ad una modifica della tassonomia iniziale incorporando in un'unica classe le due categorie in esame. Questa variazione ha permesso di migliorare sensibilmente gli esiti della classificazione (Fig. 5.9). Così facendo è stato possibile generare una matrice di confusione in cui sono emerse altre relazioni tra le classi presenti. Come mostrato in figura 5.10, le classi ⑬ e ⑭, presentano una certa difficoltà di distinzione. Anche in questo caso la causa può essere ricondotta al fatto che le e-mail appartenenti alle due categorie molto spesso presentano termini comuni e formulazioni simili.

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	24,4	0	0,28	0	0	0	0	1,2	0
2	0	0,96	0	0,04	0	0	0	0	0,44	0
3	0	1,32	0	12,28	0,22	0,52	0	0	6,8	0
4	0	0,4	0	2,92	1,6	0,16	0	0	3,04	0
5	0	0,04	0	2,48	0	8,16	0	0	1,12	0
6	0	0,68	0	0,4	0	0	0	0	0,24	0
7	0	2,08	0	1,08	0,08	0,14	0	0,2	3,12	0
8	0	1,68	0	2,56	0,04	0,08	0	0,12	19,12	0
9	0	0	0	0	0	0	0	0	0	0

Figura 5.8: Matrice di confusione di 250 e-mail di TS classificate con la prima tassonomia.

5.6 Fase di indicizzazione

Completata l'operazione di *pre-processing* del testo contenuto nella e-mail, è stato necessario procedere ad una serie di interventi per definire il dizionario dei termini da mantenere nella rappresentazione degli esempi. Durante l'indicizzazione viene generata una codifica compatta del testo da classificare, che di solito corrisponde a un insieme di pesi, che indicano la frequenza dei termini tratti da un insieme fissato.

Import degli esempi

Per poter procedere con la fase di *training* e quindi di apprendimento del modello a partire da dati supervisionati, è indispensabile importare gli esempi a disposizione in un formato specifico, conforme con il *trainer* adottato. La libreria Mallet utilizzata nel sistema di classificazione, mette a disposizione una serie di metodi che realizzano l'*import* delle istanze presenti nella cartella contenente il *Training Set*. Viene così costruita una `InstanceList`, che contiene tutte le istanze importate appartenenti al TR, e che mantiene l'associazione con due alfabeti, uno per i termini presenti nelle istanze, e un'altro distinto per tener traccia del set di *label* corrispondenti alle diverse categorie presenti.

Uno dei problemi legati alla *Text Categorization* è l'alta dimensionalità dello spazio delle *feature* che in alcuni casi può condurre a situazioni di

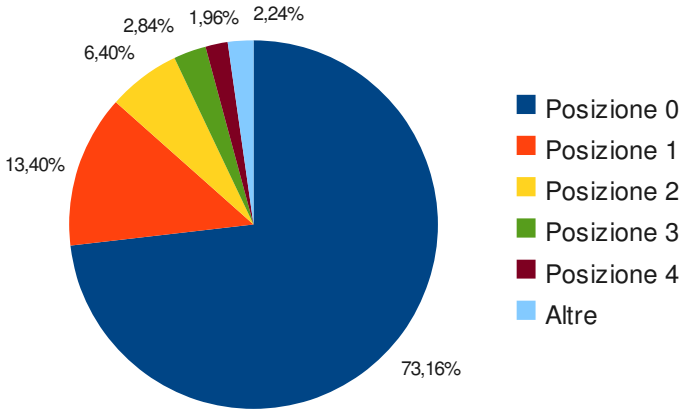

Figura 5.9: Risultati dei test (750 + 250) adottando la nuova tassonomia (*ES*: 0,537).

overfitting. Inoltre gli algoritmi di "*learning*" non scalano facilmente su grandi valori della dimensione.

Per ridurre lo spazio delle *feature* è stata sviluppata una classe apposita `ManageInstanceList`, che attraverso metodi specifici consente, in base a certi criteri, la selezione dei termini presenti nel dizionario costruito durante l'*import* degli esempi.

È bene ricordare che una delle assunzioni che sta alla base di un sistema di classificazione testuale, è che la frequenza, con cui un determinato termine compare in un testo, fornisce una misura di quanto quel termine sia significativo per il documento in esame. Da questo ne deriva che la frequenza può essere presa in considerazione per estrarre parole da utilizzare per la rappresentazione di un documento.

Sulla base di queste premesse la classe implementata mette a disposizione due metodi per eseguire la *feature selection*:

- il primo metodo calcola la *Document Frequency* di ogni termine;
- il secondo si basa sul calcolo dell'*Information Gain* dei termini rispetto le singole classi.

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	39,08	0	1,56	0	0	0	0,12	0
2	0	1,64	0	0,16	0	0	0	0	0
3	0	3,68	0	28,24	0	0,04	0,16	0	0
4	0	0,4	0	5,64	0,4	0,08	0	0	0
5	0	0,24	0	5,16	0,04	5,44	0	0	0
6	0	1,6	0	0,4	0	0	0	0	0
7	0	2,12	0	3,8	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0

Figura 5.10: Matrice di confusione di 250 e-mail di TS classificate con la nuova tassonomia.

Document Frequency

La prima operazione consiste nel calcolare in quanti esempi del *Training Set* ciascun termine compare almeno una volta (DF). Dopo aver individuato la frequenza di ogni *feature*, il dizionario che viene generato sarà costituito dai soli termini che hanno registrato una frequenza superiore ad un certo parametro fissato. In questo caso si è deciso di utilizzare come soglia il valore tre, e di conseguenza tutte le parole che compaiono in un numero di documenti minore o uguale a tre sono state rimosse dal dizionario. Questa scelta deriva dal fatto che termini poco frequenti non danno un contributo significativo alla classificazione del testo. Al momento è stato introdotto solamente questo limite inferiore, ma chiaramente è possibile adottare anche una soglia superiore per scartare dal dizionario quei termini che essendo piuttosto comuni (alto numero di occorrenze nei diversi documenti), non rappresentano delle discriminati utili per l'individuazione corretta delle categorie. In figura 5.11 vengono illustrati i risultati di classificazione ottenuti aggiungendo il calcolo della *Document Frequency*. Sia dal grafico delle percentuali di classificazione corrette che dalla matrice di confusione in figura 5.12 appare evidente un sensibile miglioramento nei risultati di classificazione.

Information Gain

La seconda operazione che è stata eseguita per la riduzione dei termini presenti nel dizionario è stata quella di individuare i termini con *Information Gain* più alto per una categoria fissata. Per ogni singola categoria l'insieme degli esempi di *Training Set* a disposizione, viene diviso in due sottoinsiemi:

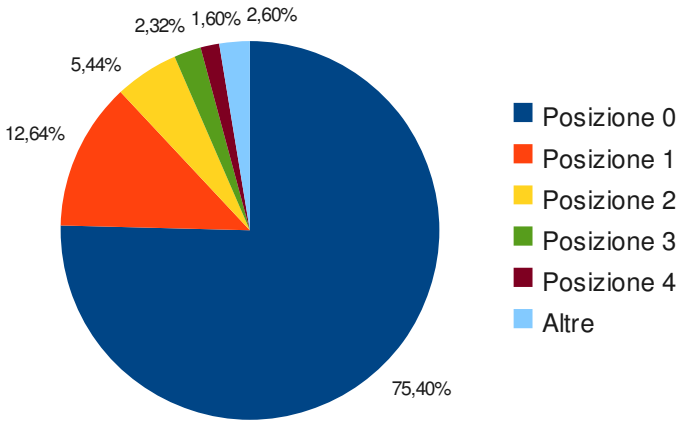

Figura 5.11: Risultati dei test (750 + 250) introducendo il calcolo della *Document Frequency* (*ES*: 0,497).

uno con i soli esempi appartenenti alla classe considerata (positivi), e un'altro con tutti gli altri esempi del TR (negativi). A questo punto per ogni termine del dizionario viene calcolato il valore di *Information Gain* rispetto alla categoria fissata. I termini vengono ordinati in modo decrescente sulla base del IG e in base al valore di un fattore α (il parametro è pari a $|initialDictionary|/|categorySet|$) vengono selezionati i primi α termini nella lista. Completate le operazioni appena descritte per ogni classe presente, viene eseguito il *merge* di tutte le liste di termini locali alle singole classi, per formare il dizionario globale. Sulla base di quest'ultimo poi si procede con un *update* di tutte le istanze. Di seguito viene riportato lo pseudo-codice relativo.

Definizione delle variabili coinvolte:

- $instanceList$ = lista delle istanze di TR
- $categorySet$ = insieme delle label corrispondenti alle categorie presenti
- $initialDictionary$ = dizionario iniziale
- $label_i$ = label associata all'istanza i-esima
- $positiveExampleSet_c$ = insieme degli esempi cha appartengono alla classe c

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	36,64	0,44	0,96	0,08	0,24	0,28	0,48	0
2	0	1,88	0,24	0,16	0	0	0	0	0
3	0	2,8	0,2	27,32	1,04	1,48	0,4	0,64	0
4	0	0,32	0	2,04	2,64	0,68	0	0,28	0
5	0	0,24	0	2,24	0,24	7,48	0	0	0
6	0	1,52	0	0,56	0	0,12	0,2	0	0
7	0	1,52	0,16	2,44	0,76	0,36	0,04	0,88	0
8	0	0	0	0	0	0	0	0	0

Figura 5.12: Matrice di confusione di 250 e-mail di TS classificate adottando il calcolo della *Document Frequency*.

- $negativeExampleSet_c$ = insieme di tutti gli esempi di TR che non appartengono alla classe c
- $localDict_c$ = dizionario locale ad una specifica categoria
- $globalDict$ = dizionario globale ritornato come output

Algoritmo:: mergeLocalDict()

1: $globalDict \leftarrow empty$
2: **for all** $c \in categorySet$ **do**
3: $localDict_c$, $positiveExampleSet_c$, $negativeExampleSet_c \leftarrow empty$
4: **end for**
5: **for all** $c \in categorySet$ **do**
6: **for all** $i \in instanceList$ **do**
7: **if** $label_i == c$ **then**
8: $positiveExample_c \leftarrow positiveExample_c \cup i$
9: $label_i \leftarrow +$
10: **else**
11: $negativeExample_c \leftarrow negativeExample_c \cup i$
12: $label_i \leftarrow -$
13: **end if**
14: **end for**
15: **for all** $t \in initialDictionary$ **do**
16: calculate $InformationGain(t)$
17: **end for**
18: sort(initialDict)

```
    for i = 1 to α do
        localDict ← localDict ∪ initialDict[i]
    end for
    globalDict ← globalDict ∪ localDict_c
end for
return globalDict
```

Il dizionario finale era quindi costituito da tutti i termini presenti nei dizionari locali alle classi (eliminando i duplicati) più significativi per ciascuna categoria (Fig. 5.13).

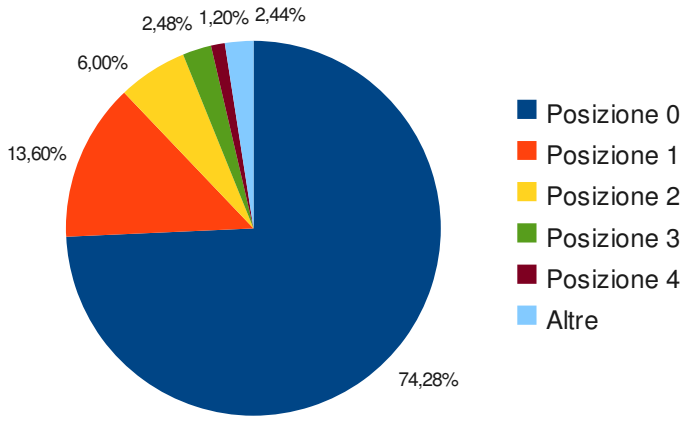

Figura 5.13: Risultati dei test (750 + 250) introducendo il calcolo dell'*Information Gain* (*ES*: 0,498).

Si è rivelato interessante verificare le variazioni delle prestazioni del classificatore introducendo entrambe le tecniche di *feature selection*, alternando anche l'ordine con cui vengono eseguite. I grafici in figura 5.14 e 5.15 riportano rispettivamente i risultati di classificazione ottenuti effettuando il calcolo della DF prima e dell'IG poi, e viceversa.

La matrice in figura 5.16 descrive le relazioni tra le classi presenti, inserendo il calcolo dell'IG seguito dalla selezione dei termini per DF. Dalle percentuali riportate si può verificare che la classificazione proposta non si scosta di molto da quella corretta. I risultati migliori sono stati conseguiti

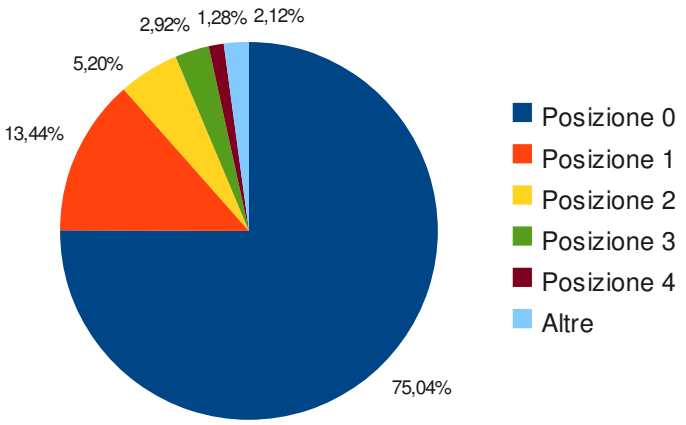

Figura 5.14: Risultati dei test (750 + 250) introducendo il calcolo della DF e dell'IG (*ES*: 0,485).

proprio in questo ultimo caso, introducendo il calcolo dell'IG prima e della DF poi, e pertanto nei tet che seguiranno la *feature selection* verrà realizzata adottando questo approccio.

In base alle esigenze aziendali è stato necessario concentrarsi ulteriormente sull'individuazione corretta di due classi specifiche, ⑬ e ⑭. Queste due categorie coinvolgono aree aziendali separate e comportano differenti priorità, ed è necessario quindi cercare di migliorarne l'identificazione. Nel capitolo seguente verranno esposti in dettaglio i diversi approcci che sono stati analizzati e applicati per cercare di migliorare ulteriormente i risultati della classificazione, con particolare attenzione rivolta verso le due classi specificate.

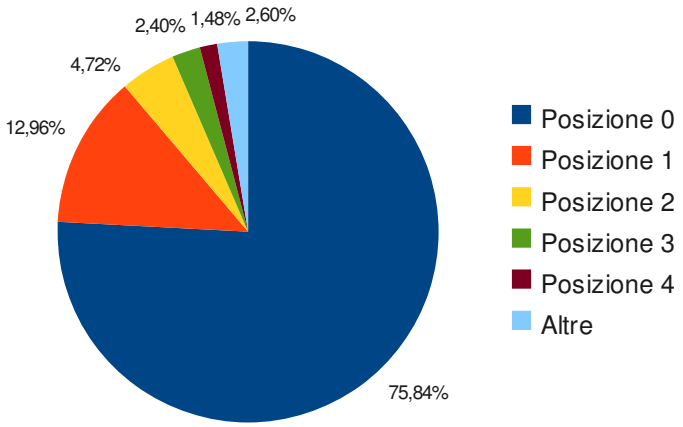

Figura 5.15: Risultati dei test su (750 + 250) introducendo il calcolo dell'IG e della DF (*ES*: 0,482).

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	38,72	0,84	1,16	0,28	0,2	0,32	0,48	0
2	0	1,76	0,32	0,2	0	0	0	0	0
3	0	2,72	0,04	24,96	0,92	1,6	0,48	0,64	0
4	0	0,28	0	2,28	3,08	0,52	0	0,24	0
5	0	0,48	0	1,84	0,16	7,6	0	0,04	0
6	0	1,32	0	0,68	0,04	0	0,04	0,04	0
7	0	1,52	0,2	2,24	0,32	0,24	0,08	1,12	0
8	0	0	0	0	0	0	0	0	0

Figura 5.16: Matrice di confusione di 250 e-mail di TS classificate adottando il calcolo dell'IG seguito dalla DF.

Capitolo 6

Revisione della fase di classificazione

Come già introdotto nel capitolo precedente, la revisione del sistema esistente ha interessato in particolare la fase di classificazione vera e propria. Di seguito vengono descritte in modo costruttivo le scelte che sono state compiute per cercare di migliorare il più possibile i risultati di classificazione, introducendo anche un altro modello di classificazione basato sulle *Support Vector Machine*.

6.1 Libreria `LibSVM`

L'implementazione della *Support Vector Machine* è stata resa possibile adottando una particolare libreria, LibSVM. LibSVM è un software che mette a disposizione una serie di classi da poter usare per la classificazione con SVM, la regressione e la stima di particolari distribuzioni; l'archivio di classi precompilate Java è stato aggiunto alle librerie di progetto. Per poter richiamare quindi metodi su oggetti definiti nelle classi di LibSVM, è stata implementata una nuova classe, `org.ictlab.cluster4.ml.SupportVectorMachine` (Fig. 6.1). Il campo `problem` di tipo `svm_problem`, è una particolare struttura di LibSVM che descrive il problema. Al suo interno viene memorizzato il numero di esempi di *Training Set* (l), un puntatore (y) alle etichette corrispondenti alle classi di appartenenza dei singoli dati, e un array di puntatori (x) a particolari oggetti di tipo `svm_node`, in cui viene mantenuta la rappresentazione per le e-mail.
Ad esempio avendo a disposizione il *Training Set* in tabella 6.1:

LABEL	ATTR1	ATTR2	ATTR3	ATTR4	ATTR5
1	0	0.1	0.2	0	0
2	0	0.1	0.3	-1.2	0
1	0.4	0	0	0	0
2	0	0.1	0	1.4	0.5
3	-0.1	-0.2	-0.1	1.1	0.1

Tabella 6.1: Esempio di *Training Set*.

in cui ad ogni riga corrisponde un esempio con label indicata nella prima colonna, il problema generato sarà così definito:

$$
\begin{aligned}
& l = 5 \\
& y \rightarrow 1\ 2\ 1\ 2\ 3 \\
x \rightarrow & [\,] \rightarrow (2, 0.1)\ (3, 0.2)\ (-1, ?) \\
& [\,] \rightarrow (2, 0.1)\ (3, 0.3)\ (4, -1.2)\ (-1, ?) \\
& [\,] \rightarrow (1, 0.4)\ (-1, ?) \\
& [\,] \rightarrow (2, 0.1)\ (4, 1.4)\ (5, 0.5)\ (-1, ?) \\
& [\,] \rightarrow (1, -0.1)\ (2, -0.2)\ (3, 0.1)\ (4, 1.1)\ (5, 0.1)\ (-1, ?)
\end{aligned}
$$

La coppia (*indice, valore*) corrisponde alla struttura `svm_node`, e la fine del vettore di rappresentazione dell'esempio è indicata con $(-1, ?)$. Data questa definizione appare subito evidente che per procedere alla creazione del problema, e del modello poi, è prevista una trasformazione delle istanze, abbandonando quindi il formato adottato dalla libreria Mallet. L'altro campo definito nell'oggetto `SupportVectorMachine` è `parameter` di tipo `svm_parameter`. Esso definisce i parametri necessari alla SVM, tra cui il tipo di kernel da adottare con le relative variabili e alcuni valori necessari in fase di training (Es. variabile C). LibSVM offre l'opportunità di poter fissare pesi diversi alle classi presenti, a seconda di determinati criteri. Questo si è rivelato essere particolarmente utile nel nostro caso, in cui è stata riscontrata la presenza di classi sbilanciate.

L'iperpiano separatore di una SVM base attribuisce lo stesso peso agli errori compiuti sugli esempi, indipendentemente dalla classe corretta di appartenenza. Questo comporta che nel caso in cui vi siano due classi sbilanciate, il classificatore identifichi gli esempi con la classe più numerosa, avendo così maggiore garanzia di successo e senza pagare una penalità superiore in caso di errore. Per risolvere questo problema è necessario quindi attribuire pesi differenti agli errori nelle varie categorie a seconda della numerosità, impostando alcuni parametri all'interno del campo `svm_parameter:parameter`. Una volta costruito il problema e fissati i parametri necessari è possibile procedere con l'operazione di *training* che genera il modello da utilizzarsi poi per la classificazione dei nuovi dati.

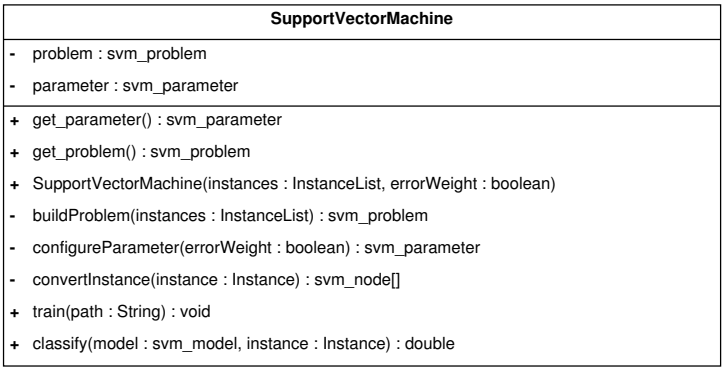

Figura 6.1: Classe `SupportVectorMachine` (*binary*).

6.2 Integrazione classificatore con SVM

All'interno del sistema preesistente, la classificazione delle nuove e-mail si basa sull'apprendimento di uno dei più semplici, ma allo stesso tempo efficiente modello disponibile, quello di Naïve Bayes. La classificazione ottenuta per una data e-mail consiste in un *ranking* delle categorie presenti, in cui ad ogni classe viene associato un valore di probabilità, ossia il grado di appartenenza alla relativa classe, che il classificatore stima per l'e-mail data. Per l'implementazione delle fasi di *training* e di *classifying* il sistema si appoggia su alcune classi messe a disposizione dalla suite Java Mallet, come descritto nel capitolo 3.

Dai primi risultati dei test è emerso che il modello appreso non è in grado di identificare e di distinguere due classi in particolare. Di conseguenza gran parte delle e-mail appartenenti ad una delle due categorie è etichettata con l'altra, e sebbene in misura minore, si verifica anche il viceversa.

Una delle cause che conduce a questi risultati potrebbe essere legata al fatto che le due classi in questione sono decisamente sbilanciate, e di conseguenza le relative probabilità a priori sono nettamente differenti. Va inoltre considerato il fatto che vi sono molti termini comuni ad entrambe le categorie, e questa ne complica ulteriormente la distinzione.

La soluzione implementata prevede l'adozione di una SVM specifica, basata sui soli esempi di *Training Set* che rientrano nelle due categorie coinvolte. L'idea di base è quella di ottenere per ogni e-mail una doppia classificazione utilizzando entrambi i modelli appresi, quello bayesiano e quello della SVM. Il *ranking* fornito dal primo modello di Naïve Bayes viene modificato e aggiornato con la classificazione proposta dalla SVM. La *Support Vector Machine* ha a disposizione due scelte possibili corrispondenti alle due categorie che comportano una maggiore difficoltà nel distinguerle. La categoria selezionata dalla SVM deve comparire nella lista di classi, ordinate

	$\gamma = 2^{-13}$		$\gamma = 2^{-11}$		$\gamma = 2^{-9}$		$\gamma = 2^{-7}$		$\gamma = 2^{-5}$		$\gamma = 2^{-3}$	
C	*ES*	%	*ES*	%	*ES*	%	*ES*	%	*ES*	%	*ES*	%
$0,01$	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9
$0,1$	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9	0.509	74.9
1	0.509	74.9	0.509	74.9	0.509	74.9	0.502	68.9	0.505	72.2	0.505	72.2
10	0.510	74.9	0.49	58.9	0.478	29.5	0.488	39.1	0.505	72.2	0.505	72.2
100	**0.479**	**28.5**	**0.479**	**25.1**	0.482	28.8	0.488	39.1	0.505	72.2	0.505	72.2
1000	0.491	27.9	0.489	29.1	0.482	28.8	0.488	39.1	0.505	72.2	0.505	72.2

Tabella 6.2: Selezione dei parametri richiesti dal kernek RBF

per valore di probabilità attraverso il modello bayesiano, in una posizione più alta rispetto a quella scartata. Nel caso in cui così non fosse vengono invertite le due label e i relativi valori di probabilità. Di seguito vengono riportati i risultati dei test che sono stati realizzati distinguendo i diversi kernel adottati.

6.2.1 Kernel RBF

In una prima fase iniziale, la valutazione delle prestazioni del classificatore è stata realizzata appoggiandosi sulla funzione kernel gaussiana, così definita:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma ||\mathbf{x} - \mathbf{x}'||^2\right) \quad \text{con } \gamma > 0$$

Chiaramente è stato necessario procedere a una serie di test per scegliere la combinazione dei parametri, C (per la *Soft-Margin SVM*) e γ che prevedono i migliori risultati di classificazione. La tabella 6.2 mette in evidenza per ogni possibile combinazione di C e γ l'*error-score* rilevato e una percentuale che rappresenta l'ammontare delle e-mail appartenenti alla categoria ⑬ e ⑭, classificate erroneamente con la classe ⑬ quando invece la classe corretta è ⑭, e viceversa. Analizzando il contenuto della tabella, è possibile mettere in evidenza alcune combinazioni di parametri che hanno portato a dei buoni risultati di classificazione. In figura 6.2 viene riportato il dettaglio della media dei risultati di classificazione ottenuti fissando C a 100 e γ a 2^{-11}.

Dalle valutazioni descritte nel capitolo precedente era emerso il fatto che le due categorie in esame sono particolarmente sbilanciate. La libreria adottata per l'implementazione (`LibSVM`) della SVM offre la possibilità di impostare il peso degli errori, compiuti dal modello di classificazione, diversificandolo a seconda della categoria di appartenenza. Agli errori commessi per le e-mail della classe più numerosa è stata assegnata una bassa penalità, mentre per quelli commessi sull'altra classe la penalità è più alta. Il rapporto dei pesi degli errori mantiene la proporzione tra il numero di esempi per le due classi.

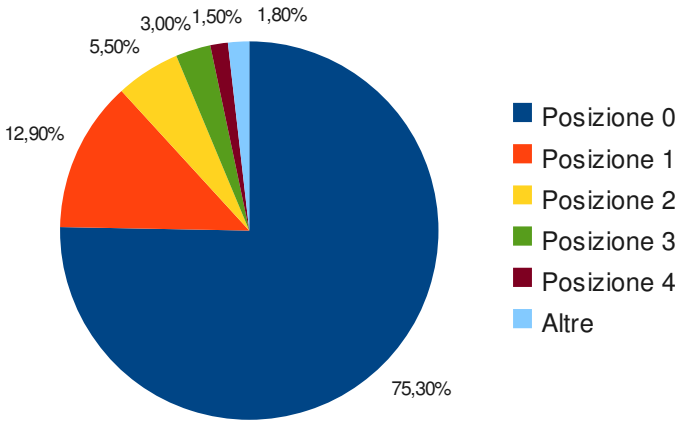

Figura 6.2: Risultati ottenuti usando kernel RBF con $C = 100$ e $\gamma = 2^{-11}$.

Ripetendo il test precedente con l'aggiunta di pesi differenti agli errori commessi sulle due classi in esame (in rapporto 5:1) è stato ottenuto un sensibile miglioramento. In figura 6.3 vengono riportati i migliori risultati ottenuti per $C = 10$ e $\gamma = 2^{-11}$. Inoltre la percentuale del livello di confusione tra le due classi è passata dal 25,1% (senza pesare gli errori) al 21,3%, apportando anche un miglioramento globale dei risultati di classificazione.

6.2.2 Kernel polinomiale

Nella fase successiva è stato adottato il kernel polinomiale, definito come:

$$k(\mathbf{x}, \mathbf{x}') = (\gamma * \mathbf{x} * \mathbf{x}' + coef0)^d$$

É stato necessario eseguire una serie di test per fissare alcuni parametri previsti dalla funzione kernel e individuare così la combinazione ottimale degli stessi. Sono stati eseguiti 3 test differenti variando il rapporto dei pesi degli errori associati alle due classi allo scopo di ridurre ulteriormente la percentuale di confusione tra le due, anche a costo di ridurre la bontà dei risultati di classificazione globali a tutte le categorie. Nel grafico in figura 6.4 vengono evidenziate le percentuali rilevate per ogni posizione nel *ranking* risultante, variando il peso degli errori sulle due categorie specifiche 3 e 4.

Nella tabella 6.3 sono indicati invece gli *Error-Score* medi e le percentuali di scambi tra le due classi che sono stati registrati variando il rappor-

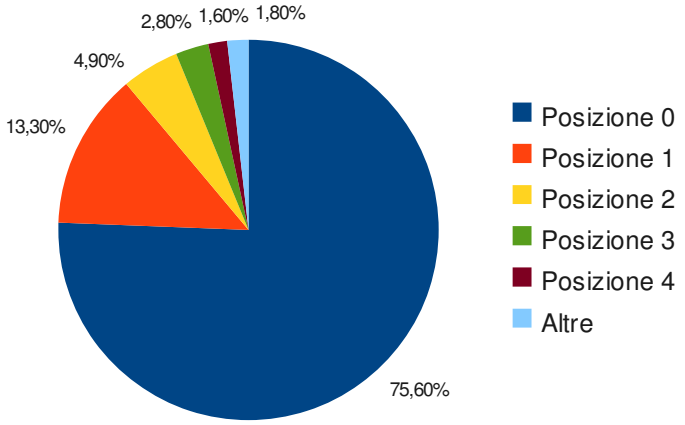

Figura 6.3: Risultati ottenuti usando kernel RBF (pesati errori 5:1) con $C = 10$ e $\gamma = 2^{-11}$ (*ES*: 0,469).

	Rapporto dei pesi sugli errori		
	5:1	**25:1**	**50:1**
Error-Score	0,469	0,498	0,620
% confusione	19%	11,7%	6,8%

Tabella 6.3: Confronto di *Error-Score* e percentuale errori su specifiche categorie, al variare del rapporto dei pesi degli errori.

to dei pesi degli errori. Comparando queste percentuali con i risultati di classificazione proposti nel grafico di confronto, appare subito evidente che aumentando la penalità della classe meno numerosa, si riduce il grado di confusione tra le due classi specifiche, ma diminuisce anche la percentuale di classificazioni corrette in prima posizione. Vi è quindi un *trade-off* tra l'esigenza di avere in posizione 0 la percentuale più alta possibile di classificazione corrette, e ridurre al minimo gli scambi tra le due categorie che hanno evidenziato maggiori problemi di distinzione.

Infine sono stati eseguiti dei test per valutare le prestazioni del classificatore, variando il dizionario usato dalla SVM specifica per le due categorie. Un primo test è stato completato costruendo una sorta di dizionario locale, importando le sole istanze appartenenti alle due classi; il successivo invece introduceva il calcolo dell'*Information Gain* dei termini rispetto alle sole due

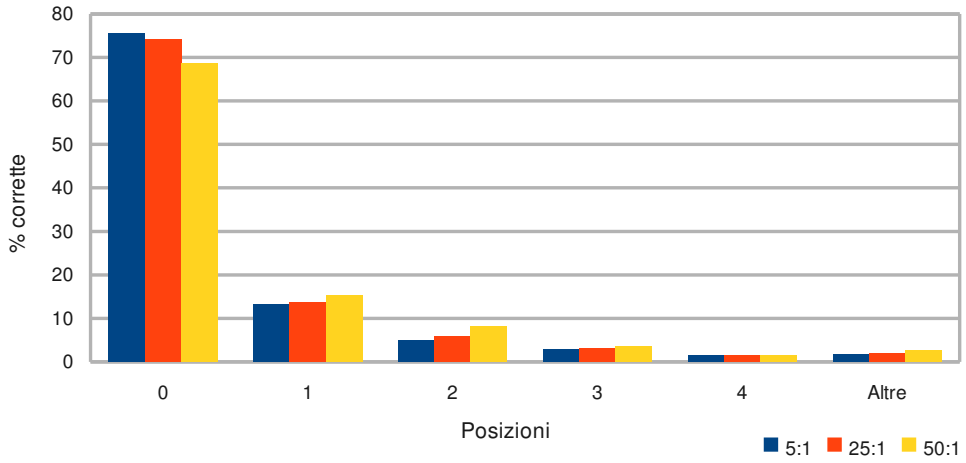

Figura 6.4: Confronto risultati di classificazione con kernel polinomiale, variando il rapporto tra i pesi degli errori.

classi e manteneva solo quelli più significativi. I risultati dei test hanno dimostrato che le prestazioni della SVM migliorano mantenendo un dizionario ricco di termini, e quindi tutte le riduzioni che sono state testate, hanno portato ad un peggioramento. Di conseguenza si è deciso di mantenere per la SVM lo stesso dizionario generato per il modello bayesiano.

6.3 Adozione SVM *multilabel*

LibSVM implementa in modo efficiente anche la classificazione *multilabel*, che si distingue dal caso binario in quanto per ogni esempio la scelta di classi da associare non è più limitata a due categorie, ma è estesa ad un numero superiore di alternative. In questo contesto si è deciso di sfruttare questa opportunità per implementare una versione ancora più sofisticata del classificatore adottato. Dopo uno studio iniziale delle potenzialità delle libreria, sono state apportate delle modifiche alla implementazione della classe `SupportVectorMachine` per gestire la classificazione basandosi su tutte le categorie presenti. L'idea è quella di fornire come classificazione un *ranking* simile a quello proposto dal modello bayesiano, in cui vengono ordinate tutte le categorie a seconda del valore di probabilità stimato dal classificatore. In particolare è stato necessario modificare il metodo `SupportVectorMachine::classify()`, aggiungendo tra i parametri un vettore per le probabilità stimate per i risultati di classificazione e richia-

mando all'interno un metodo della classe `LibSVM.svm` specifico per una classificazione *multilabel*.

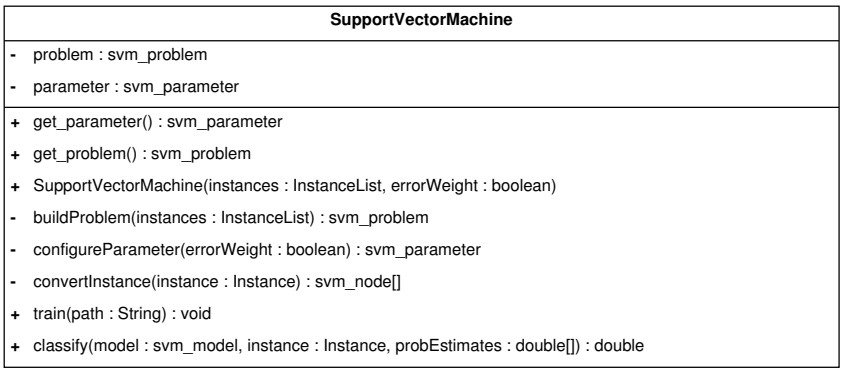

Figura 6.5: Classe `SupportVectorMachine` (*multilabel*)

Sono stati eseguiti due differenti test per verificare le prestazioni del classificatore, pesando o meno, in modo differente, gli errori commessi sulle diverse categorie in base alla numerosità di queste ultime.

Il primo test realizzato prevede di associare ad ogni errore compiuto sulle diverse classi lo stesso peso, indipendentemnte quindi dalla numerosità della classe coinvolta. Anche in questo caso sono stati eseguiti dei test per selezionare gli *hyperparameter* ottimali utilizzati dalla funzione kernel polinomiale. I migliori risultati ottenuti, con due diverse combinazioni di *hyperparameter*, vengono riportati in figura 6.6, mettendo in evidenza la percentuale di e-mail delle classi ⑬ e ⑭ che sono state scambiate tra le due categorie. La differenza tra le percentuali nel *ranking* riportato è minima, tanto che il valore di *Error-Score* associato è lo stesso.

Il modelo di classificazione è stato ulteriormente raffinato assegnando pesi diversi agli errori commessi sulle classi, in modo inversamente proporzionale alla numerosità di queste ultime. Dai risultati riscontrati emerge che l'introduzione di pesi differenti ha ridotto sensibilmente il livello di confusione tra le due classi in esame, fino a raggiungere la percentuale del 29,6%. Assegnando infatti una penalità più alta nel caso di classificazione scorretta per esempi appartenenti a classi poco numerose, si evita che il modello di classificazione associ agli esempi la classe più popolosa avendo maggiori garanzie di successo.

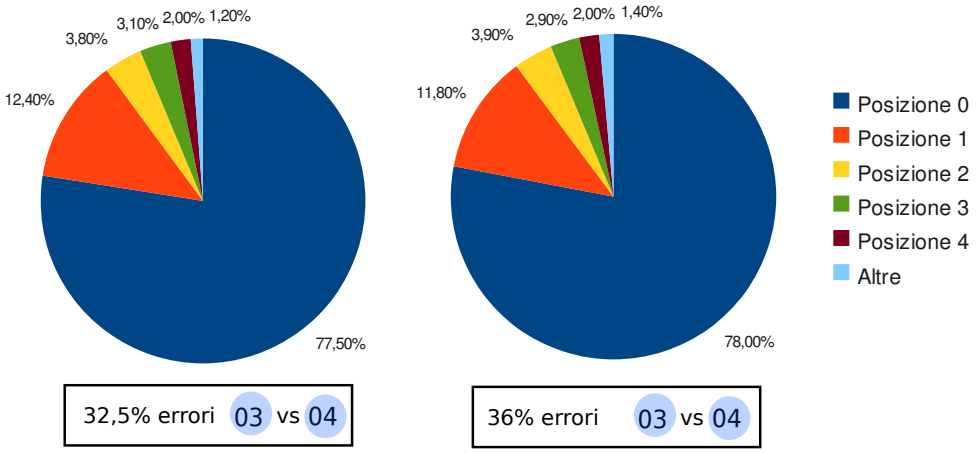

Figura 6.6: Risultati dei test (750 + 250) ottenuti con *SVM multilabel equal weights* (*ES:* 0,433).

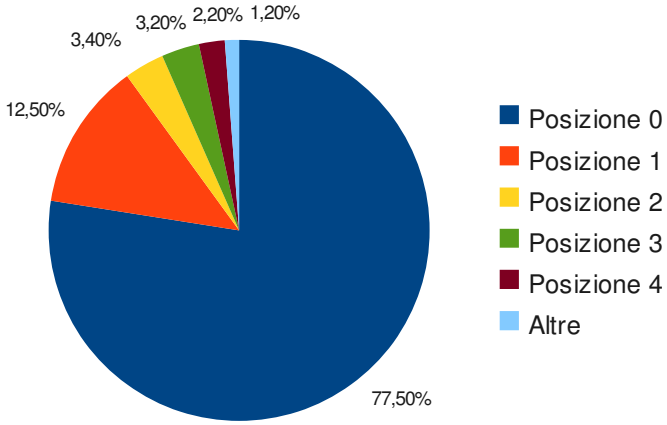

Figura 6.7: Risultati dei test (750 + 250) ottenuti con *SVM multilabel different weights* (*ES:* 0,436).

In figura 6.7 vengono riportate le percentuali di classificazioni corrette riscontrate assegnando pesi diversi agli errori sulle varie categorie. I risultati sono stati ottenuti adottando un kernel polinomiale di grado 2, fissando $C = 100$ e $\gamma = 2^{-}13$. La scelta di questi parametri è il risultato di una serie di test per la selezione degli *hyperparameter* ottimali, eseguiti variando appunto i valori di C, γ e del grado del polinomio.

6.4 Test conclusivi

I test sono stati conclusi con un ulteriore incremento dei dati a disposizione, 1500 e-mail di *Training Set* e 500 di *Test Set*. Le prestazioni del classificatore sono state rivalutate con i nuovi dati di *training*, per ogni modello di classificazione testato in precedenza, ossia il modello di Naïve Bayes semplice, NB integrato con la *Support Vector Machine* specifica per due categorie e infine la SVM *multilabel*.

Chiaramente l'aggiunta di dati preclassificati ha apportato un notevole raffinamento nei risultati di classificazione. Nel grafico in figura 6.8 vengono evidenziate le prestazioni dei tre diversi modelli testati, dopo aver eseguito una serie di test per la scelta di specifici parametri ove fosse richiesto. Tra i tre approcci proposti, quello bayesiano garantisce la percentuale minore di classificazioni corrette in posizione 0 del *ranking* (76,8%), ma la percentuale di confusione tra le due categorie ⑬ e ⑭ è la minore in assoluto, pari al 12,4%.

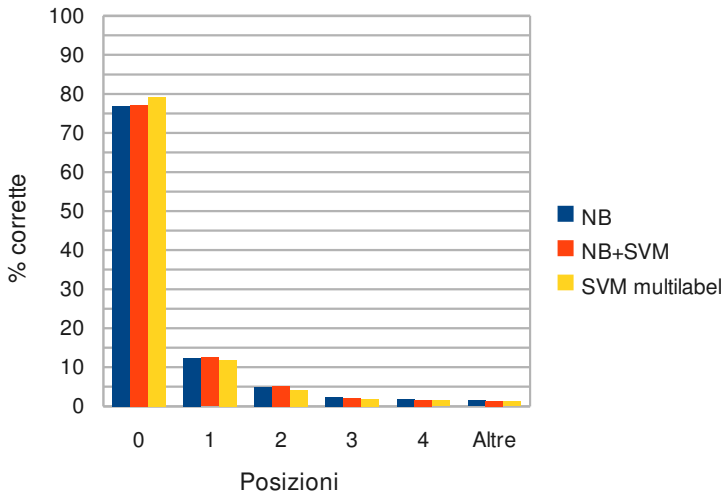

Figura 6.8: Confronto risultati dei test (1500 + 500) adottando NB, NB+SVM, SVM *multilabel*.

Le prestazioni migliori si hanno adottando la SVM *multilabel*, e questo è verificabile analizzando le percentuali riportate in figura 6.9. Nonostante questo però, la percentuale di e-mail della classe ⑬ confuse con la classe ⑭, e viceversa, resta comunque abbastanza alta (20,6%). I risultati di questi

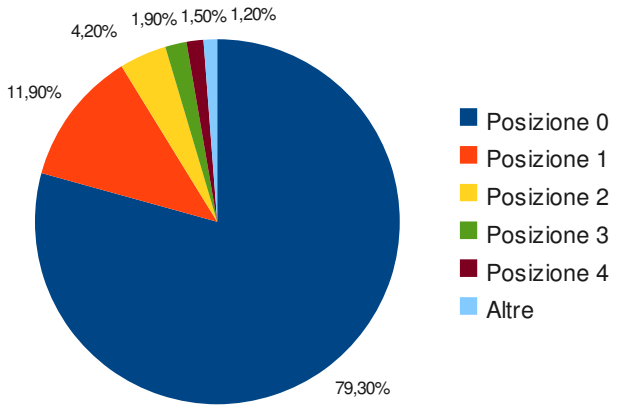

Figura 6.9: Risultati dei test (1500 + 500) ottenuti con *SVM multilabel different weights* (*ES:* 0,381).

ultimi test sono stati sottoposti ai responsabili aziendali, per discutere pro e contro di ogni risultato conseguito. Al termine della discussione, si è deciso di scegliere come modello di classificazione la SVM *multilabel* a cui corrisponde la percentuale di classificazioni corrette più alta in posizione 0.

6.5 Estensione della classificazione alle categorie di secondo livello

Prima di concludere l'attività di tesi è stata realizzata l'estensione della classificazione alle sottocategorie presenti nella tassonomia fornita dall'azienda. Pertanto oltre al modello globale appreso per classificare le e-mail con le categorie di primo livello, è stato necessario introdurre l'apprendimento di un modello per ogni singola classe, in modo da poter estendere la classificazione alle sottocategorie presenti nella gerarchia. In seguito alla fase di *training* vengono memorizzati nella cartella `temp` della *Home* del classificatore, i modelli appresi per ciascuna macro-categoria. Durante la classificazione viene adottato il modello globale per generare un *ranking* delle categorie di primo livello, e successivamente, per ognuna delle classi mantenendo l'ordine

in cui compaiono, viene invocato il modello relativo a ciascuna di esse e viene proposto un ulteriore *ranking*, relativo alle sottocategorie della classe considerata.

Il risultato di classificazione pubblicato contiene tutte le categorie e sottocategorie presenti nella tassonomia, in ordine decrescente di probabilità associata dal classificatore.

6.5.1 Rilevazione prestazioni del sistema a regime

Per poter rilevare le prestazioni del sistema in funzione presso l'azienda, è stato necessario realizzare una nuova modalità di valutazione dei risultati che prenda in considerazione anche la classificazione di secondo livello. L'idea di base è quella di individuare la posizione in cui il classificatore colloca la classificazione corretta data una certa e-mail. Si è rivelato interessante analizzare le prestazioni del classificatore riportando tre diverse statistiche che si focalizzano su diversi aspetti. Il primo risultato (Fig. 6.10) mette in evidenza le percentuali di classificazioni corrette collocate nelle prime 5 posizioni del *ranking* finale, considerando però solo la categoria di primo livello, e trascurando quindi i risultati della sottoclassificazione. I valori percentuali

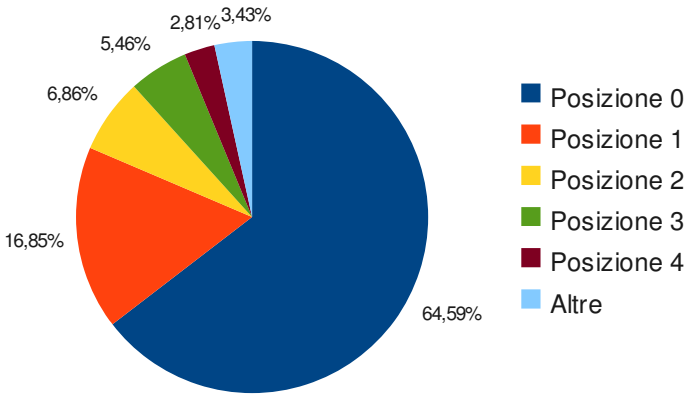

Figura 6.10: Valutazione percentuali corrette di primo livello del sistema attualmente in uso.

riscontrati sono evidentemente peggiori rispetto alle percentuali di classificazioni corrette rilevate con gli ultimi test. Questo è dovuto sicuramente

Categoria I livello	% corrette di II livello prime 5 posizioni					
	0	**1**	**2**	**3**	**4**	**Altre**
⑴	36,05	25,00	15,12	5,23	6,40	12,20
⑵	46,67	33,32	6,67	6,67	6,67	-
⑶	49,33	18,00	8,67	5,33	5,33	13,34
⑷	56,76	12,61	10,81	9,91	4,50	5,41
⑸	77,17	14,96	5,51	2,36	-	-
⑹	46,34	31,71	9,75	12,20	-	-
⑺	60,00	24,00	4,00	8,00	0,00	4,00
⑻	-	-	-	-	-	-

Tabella 6.4: Collocazione delle % corrette di II livello per ogni categoria di I livello.

alla tipologia di dati da classificare che sono stati sottoposti al classificatore. In fase di testing, le e-mail fornite dall'azienda erano state preventivamente selezionate, eliminando eventuale spam, considerando solamente i messaggi in lingua italiana e e-mail appartenenti ad un'unica classe. Attualmente il sistema viene utilizzato senza effettuare alcun tipo di selezione a priori, e i messaggi di posta elettronica provengono dalle diverse fonti a disposizione senza applicare dei filtri. Questa scelta deriva dalla volontà di verificare la prestazioni del sistema senza interventi esterni, per poter individuare ulteriori problematiche, legate alla natura dei dati da classificare, e eventuali modifiche che possono essere apportate. Nella tabella 6.4 viene messa in evidenza, l'accuratezza che il classificatore ha dimostrato nella classificazione di secondo livello, distinguendo per ogni macro-categoria, la collocazione in percentuale delle classificazioni corrette di livello inferiore, riportando in modo esplicito i valori percentuali relativi alle prime 5 posizioni.

Infine viene proposto un'ulteriore risultato che consente di avere una visione globale della capacità del classificatore di individuare la classificazione completa corretta, quindi di primo e di secondo livello. In base alle esigenze aziendali, l'analisi dei risultati è stata limitata ad un certo numero di posizioni all'interno del *ranking* completo. Nello specifico in tabella

Ulteriore causa che può aver condotto ai risultati illustrati, potrebbe essere anche la scelta degli *hyperparameter* della funzione kernel adottata dalla SVM. Attualmente, nel momento in cui il sistema acquisisce nuova conoscenza (dati di feedback), viene rieseguita la fase di *training*, mantenendo però sempre gli stessi *hyperparameter* appresi e fissati nelle fasi precedenti di testing. Questa procedura tuttavia non è quella ottimale: la variazione dei dati di training comporta la necessità di rieseguire il *tuning* dei parametri periodicamente, in modo tale da individuare il modello ottimale per le nuove distribuzioni dei dati. Questa attività costituisce una possibile modifica del sistema, descritta nel capitolo relativo agli sviluppi futuri.

Parte III

Conclusioni

Capitolo 7

Considerazioni finali

Capitolo 8

Sviluppi futuri

Bibliografia

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

[2] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys, 34(1):1–47,* 2002.

[3] R. Bekkerman, A. McCallum and G. Huang. Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. *Center for Intelligent Information Retrieval, Technical Report IR, Vol. 418,* 2004.

[4] H. Drucker, D. Wu and V. N. Vapnik. Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks, Vol. 10, No. 5,* September 1999

[5] D. D. Lewis and K.A. Knowles. Threading Electronic Mail: A Preliminary Study. *Information Processing and Management, 33:2,209-217,* 1997.

[6] E. Giacoletto and K. Aberer. Automatic Expansion of Manual Email Classifications Based on Text Analysis. *In Proceedings of ODBASE'2003, the 2nd International Conference on Ontologies, Databases, and Applications of Semantics, pp.785 802,* 2003.

[7] T. R. Payne and P. Edwards. Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence, 11(1):1–32,* 1997.

[8] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning, 3(4):261–283,* 1989.

[9] W. W. Cohen. Learning rules that classify e-mail. *In Proceedings of AAAI Spring Symposium on Machine Learning and Information Retrieval,* 1996.

[10] J. Provost. Naive-bayes vs. rule-learning in classification of email. *Technical Report AITR-99-284, University of Texas at Austin, Artificial Intelligence Lab,* 1999.

[11] J. Rennie. ifile: An application of machine learning to e-mail filtering. *In Proceedings of KDD'2000 Workshop on Text Mining,* 2000.

[12] S. Kiritchenko and S. Matwin. Email Classification with CoTraining. *In Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research,* 2001.

[13] R. Segal and J. Kephart. Incremental learning in swiftfile. *In Proceedings of ICML-00, 17th International Conference on Machine Learning,* 2000.

[14] S. Kiritchenko, S. Matwin, and S. Abu-Hakima. Email classification with temporal features. *In M. Klopotek, S. Wierzchon, and K. Trojanowski, editors, Intelligent Information Processing and Web Mining, pages 523–533. Springer,* 2004.

[15] A. Lampert, R. Dale and C. Paris. Segmenting email message text into zones. *Proceeding EMNLP '09 Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2,* 2009.

[16] A. Ben-Hur and J.A. Weston. A user's guide to support vector machines. *Department of Computer Science, Colorado State University, Fort Collins, CO, USA,* 2008.