Università degli Studi di Padova

Dipartimento di Ingegneria Dell'Informazione

Corso di laurea triennale in Ingegneria Informatica

Progettazione e implementazione di una interfaccia grafica in Java Swing per lo studio di classificatori automatici di testi

Laureando: Andrea Bisson

Relatore: Prof. Giorgio Maria Di Nunzio

21 Febbraio 2012

Anno Accademico 2011/2012

Indice

Elenco delle figure

Capitolo 1

Introduzione

La classificazione di un testo (TC - text classification) è l'abilità di assegnare un testo a una determinata categoria presa da un insieme predefinito. Con l'avvento di internet, il problema di classificare un testo in base al contenuto si è fatto sempre più importante, dalla ricerca di documenti nel web in base all'argomento trattato, alla loro archiviazione automatica. Altre applicazioni molto importanti sono: il filtro adattivo per il riconoscimento dello spam, la generazione automatica di metadati per un testo e la disambiguazione del senso delle parole. [5]

Tutto questo viene svolto, ad esempio, da modelli probabilistici ad apprendimento automatico che permettono di classificare senza la presenza di un addetto che lo faccia manualmente, ottenendo così un contenimento dei costi a lungo termine, oltre che una velocizzazione della procedura.

Risulta quindi importante cercare di migliorare questi modelli in modo da rendere la sua capacità di riconoscimento di testi il più simile possibile a quella umana.

Questa tesi ha lo scopo di realizzare un'interfaccia grafica per avere una visione immediata delle conseguenze nel cambio dei parametri del modello che rappresentano la nostra conoscenza a priori della distribuzione della probabilità delle parole che costituiscono i documenti. Il valore di questi parametri, che chiameremo α e β, influisce sulla probabilità dei termini del documento e quindi sulla decisione della loro categoria di appartenenza. Il punto che rappresenta tale probabilità viene disegnato sul piano cartesiano e la visualizzazione realizzata permette di rappresentare un documento su un piano bidimensionale. La posizione sul piano determina la categoria del documento a seconda se esso si trova sopra o sotto la retta bisettrice $y = x$. Di conseguenza il cambio di valore di tali parametri porta ad una traslazione dei punti, e quindi ad una diversa classificazione dei documenti da essi rappresentati.

Il capitolo 2 parla dello schema di progettazione MVC utilizzato per lo sviluppo del programma, il quale ha dato una certa modularità nella sua realizzazione oltre che agevolare eventuali future modifiche. A seguire, nel capitolo 3, si elencano gli obiettivi e i requisiti da conseguire nell'interfaccia grafica e infine, nel capitolo 4, si analizza in dettaglio il modello probabilistico e si discute la realizzazione e il funzionamento dell'interfaccia realizzata per interagire con esso.

Capitolo 2

Il Pattern MVC

2.1 Generalità

Nato nel 1979 nel Palo Alto Research Center (PARC)[1] di Xerox in California, il pattern MVC (Model-View-Controller) ha lo scopo principale di rendere il più possibile indipendenti i dati dall'interfaccia grafica. Esso è formato da tre elementi:

- **M**odello: racchiude i dati su cui l'applicazione deve lavorare
- **V**ista: possono essere più di una, e permettono di visualizzare graficamente i dati e di interagire con essi
- **C**ontrollore: gestisce lo scambio di informazioni tra vista e modello

In questo modo le informazioni possono essere visualizzate indipendentemente da come sono strutturate nel modello, inoltre, un software così strutturato diventa sì più complesso, ma la sua manutenibilità e la sua modularità aumentano: infatti è possibile modificare l'interfaccia grafica senza dover mettere mano al modello e le successive modifiche possono quindi essere svolte in maniera più agevole.

2.2 Funzionamento

Il modello e le viste sono collegate al controllore, che svolge il lavoro di mediatore tra i due. Ogni cambiamento dei dati del modello viene segnalato al controllore, il quale segnala alle viste i cambiamenti avvenuti. Viceversa ogni modifica dei dati fatta dall'utente sulle vista, viene trasmessa al controllore che a sua volta le passerà al modello. Nel caso di più viste, una modifica su una di esse, porta necessariamente alla modifica di tutte le altre viste relative allo stesso modello. L'algoritmo che mostra l'interazione tra le componenti del modello MVC per un

[1] http://www.parc.com/

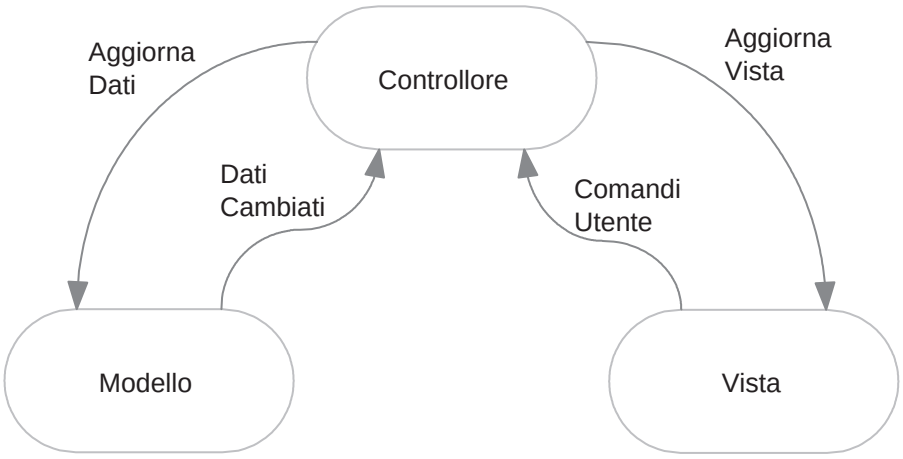

Figura 2.1: Topologia lineare del pattern MVC

evento di un utente sulla vista è:

1. L'utente agisce sulla vista azionando un particolare listener che chiama il rispettivo metodo del controller
2. Il controller segnala al modello che è stato modificato un certo dato
3. Il modello, se è stato modificato, si aggiorna e segnala al controller i dati che sono stati cambiati.
4. Il controller passa le modifiche alle viste
5. Le viste si modificano secondo l'aggiornamento del modello

Seguendo i vari passi, si può subito notare un certo loop fra modello e viste. Bisogna quindi evitare che il ciclo continui indefinitamente pur essendo tutto sincronizzato, implementando degli opportuni controlli sulle modifiche da effettuare. [1, 2]

Esistono tuttavia anche altre implementazioni dello schema MVC, le quali prevedono una comunicazione diretta del modello sulla vista tramite dei listeners senza passare per il controllore, senza però rendere note le viste al modello. Inoltre le viste richiedono un controllore per ogni funzionalità implementata e tutto ciò comporta una certa dipendenza tra le componenti, a differenza dell'altra implementazione. Il pattern MVC, quindi, può avere due topologie differenti: a triangolo o lineare.

Capitolo 3

Obiettivi e Requisiti

3.1 Obiettivi

Il progetto ha l'intento di realizzare un'interfaccia grafica per vedere le conseguenze nel cambio dei parametri del modello probabilistico per la classificazione di testi.

Lo sviluppo segue lo schema progettuale del pattern MVC, già trattato nel capitolo precedente, che permette di garantire indipendenza tra il modello e l'interfaccia grafica. Si analizzerà poi anche il funzionamento del modello probabilistico per la categorizzazione dei documenti di testo. Infine si tratterà delle scelte progettuali e dell'implementazione dell'interfaccia grafica fatta in Java Swing [10], creata in base ai requisiti richiesti, e della libreria JFreeChart [12] utilizzata per realizzare i grafici richiesti per la visualizzazione del modello.

3.2 Requisiti dell'interfaccia grafica

I requisiti richiesti per la realizzazione dell'interfaccia sono:

1. Realizzare un grafico che permetta di visualizzare su un piano cartesiano i punti ricavati dai documenti di addestramento

2. Realizzare un pannello che permetta di immettere il valore dei due parametri del modello

3. Possibilità di far comparire su richiesta un altro grafico che contenga la rappresentazione dei documenti di test

4. Permettere all'utente di regolare il range di valori dei due parametri

5. Possibilità di scegliere il valore dei parametri anche tramite uno slider

6. Disegnare il grafico dei punti di un modello (Multinomiale o Bernoulli) alla pressione di un apposito pulsante

7. Possibilità di disegnare automaticamente il grafico senza premere il pulsante, dopo aver modificato il valore di uno dei parametri. Ciò può essere disattivato su richiesta dell'utente

8. Permettere all'utente di scegliere il modello nel caso sia attiva la modalità di grafico automatico

9. Far comparire le informazioni relative al documento, nel caso in cui l'utente clicchi su un punto del grafico

10. Permettere di cambiare i parametri del modello relativi al solo documento selezionato dall'utente

Capitolo 4

Progettazione e Realizzazione

4.1 Il Modello

Il modello, nel caso specifico della classificazione automatica dei testi, consta in un database contenente i documenti da classificare e altre informazioni come: le categorie, il modello probabilistico [3, 5] e i suoi parametri.

4.1.1 Il modello probabilistico

Sia dato un insieme finito prestabilito di categorie $C = \{c_1, \ldots, c_n\}$ e sia d un documento preso da un insieme finito di documenti D, ciascuno formato da un numero finito di termini. La probabilità che un termine t_k compaia in un documento appartenente a una categoria c_i é:

$$P(t_k|c_i)$$

la quale viene stimata dai documenti di addestramento (training documents) usando una stima di massima verosimiglianza (ML - Maximum Likelihood):

$$P_{ML}(t_k|c_i) = \frac{\tau_{k,i}}{m_i} \tag{4.1}$$

dove $\tau_{k,i}$ è il numero di documenti della categoria c_i in cui appare il termine t_k, e m_i rappresenta il numero totale di documenti appartenenti a c_i. Tuttavia la (4.1) potrebbe essere pari a zero nel caso in cui un termine t_k non compaia nella categoria c_i.

Assumendo che i termini t_k siano indipendenti tra loro, questi si possono considerare come sottoinsiemi dell'insieme documento d. Pertanto per la regola del prodotto per probabilità condizionate, si ricava la probabilità che un documento appartenga a una certa categoria c_i:

$$P(d|c_i) = \prod_k P(t_k|c_i) \tag{4.2}$$

ed equivalentemente la probabilità che un documento non appartenga alla categoria c_i:

$$P(d|\bar{c}_i) = \prod_k P(t_k|\bar{c}_i) \tag{4.3}$$

con $\bar{c_i} = C - c_i$.
Inoltre per il teorema di Bayes, si ha l'uguaglianza:

$$P(d|c_i)P(c_i) = P(c_i|d)P(d)$$

dove $P(c_i)$ viene chiamata probabilità a priori e $P(d|c_i)$ viene chiamata verosimiglianza (likelihood), entrambe ricavate dai documenti di addestramento. Quindi, la probabilità che sia la categoria c_i dato il documento d è:

$$P(c_i|d) = \frac{P(d|c_i)P(c_i)}{P(d)} \tag{4.4}$$

equivalentemente, la probabilità che non sia la categoria c_i dato il documento d:

$$P(\bar{c_i}|d) = \frac{P(d|\bar{c_i})P(\bar{c_i})}{P(d)} \tag{4.5}$$

Si definisce ora il rapporto, detto *odds*, tra le equazioni (4.4) e (4.5):

$$\frac{P(c_i|d)}{P(\bar{c_i}|d)} = \frac{P(d|c_i)P(c_i)}{P(d|\bar{c_i})P(\bar{c_i})} \tag{4.6}$$

e sostituendo la (4.2) e la (4.3) rispettivamente al numeratore e al denominatore, si ottiene

$$\frac{P(c_i|d)}{P(\bar{c_i}|d)} = \frac{\prod_k P(t_k|c_i)P(c_i)}{\prod_k P(t_k|\bar{c_i})P(\bar{c_i})} = \frac{P(c_i)}{P(\bar{c_i})} \prod_k \frac{P(t_k|c_i)}{P(t_k|\bar{c_i})} \tag{4.7}$$

Se il rapporto (4.6) è maggiore di 0.5, allora il documento d viene assegnato alla categoria c_i, altrimenti viene assegnato a $\bar{c_i}$.

4.1.2 Approssimazione a priori Bayesiana

Preso un insieme finito di documenti tra loro indipendenti $D = \{d_1, \ldots, d_m\}$, dove ogni documento d_i assume il valore 1 con probabilità θ (Bernoulli), la sua distribuzione a priori è definita

$$P(\theta) = \theta^{\alpha}(1-\theta)^{\beta}$$

dove $\alpha > 0$ e $\beta > 0$ sono detti iperparametri. Tramite il teorema di Bayes, si può definire la distribuzione a posteriori dati i documenti

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

Tuttavia, il calcolo della probabilità a posteriori può essere difficoltoso e quindi si ricorre alla distribuzione a priori coniugata della funzione di verosimiglianza $P(D|\theta)$ che, per una Bernoulliana, è la distribuzione beta:

$$P(\theta|\alpha,\beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha,\beta)} = beta(\theta;\alpha,\beta) \tag{4.8}$$

dove α e β sono gli iperparametri e $B(\alpha, \beta)$ è la funzione beta che serve per mantenere l'area della distribuzione unitaria. Quindi la probabilità di avere una certa sequenza D di documenti è:

$$P(D|\theta) = P(\tau_k, m|\theta) = \theta^{\tau_k}(1-\theta)^{m-\tau_k} \tag{4.9}$$

dove τ_k è il numero totale di volte in cui il termine t_k appare negli m documenti. La distribuzione a posteriori diventa:

$$P(\theta|\tau_k, m) = \frac{P(\tau_k, m|\theta)P(\theta)}{P(\tau_k, m)} = beta(\theta; \tau_k + \alpha, m + \beta - \tau_k) \tag{4.10}$$

Grazie a questi risultati è possibile predire la probabilità che un termine t_k compaia in una data categoria c_i, calcolando l'aspettazione della distribuzione a posteriori (4.10) (sapendo che l'aspettazione di $beta(\alpha, \beta)$ è $\frac{\alpha}{\alpha+\beta}$):

$$\begin{aligned} P(t_k|c_i; \hat{\theta}) &= E[\theta|\tau_k, m] = \\ &= \int \theta P(\theta|\tau_{k,i}, m_i)d\theta = \frac{\tau_{k,i} + \alpha}{m_i + \alpha + \beta} \end{aligned} \tag{4.11}$$

dove $\hat{\theta}$ è il parametro previsto per la Bernoulli. Il risultato della (4.11) è una approssimazione della stima di massima verosimiglianza (4.1) ($P_{ML}(t_k|c_i)$).

4.1.3 Visualizzazione grafica per la scelta degli iperparametri

La scelta di α e β nella distribuzione a priori è molto importante, in quanto i loro valori contribuiscono nell'assegnare o meno i documenti alla giusta categoria. Per agevolare la scelta si tracciano graficamente le probabilità $P(c_i|d)$ e $P(\bar{c_i}|d)$ del rapporto (4.6), che corrispondono alle coordinate del punto che rappresenta un documento. Tuttavia, poiché queste tendono a zero velocemente, si applicano i logaritmi alle probabilità che portano a disporre i punti nel terzo quadrante del piano cartesiano. Il loro rapporto (4.6) può essere considerato come la bisettrice primo-terzo quadrante, la quale indica che tutti i documenti che sono sopra di essa sono considerati appartenenti alla categoria c_i e quelli sotto appartenenti a $\bar{c_i}$.

La scelta dei valori degli iperparametri α e β porta quindi a una traslazione del punto lungo gli assi, e di conseguenza a seconda della relativa disposizione rispetto alla bisettrice, ad una diversa decisione per la categoria di appartenenza.

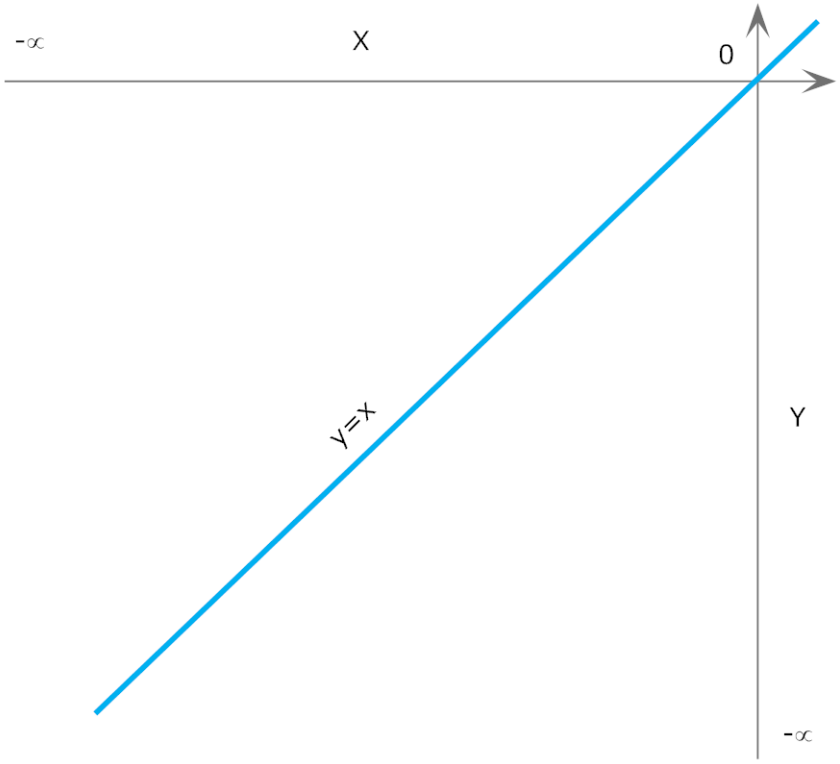

Figura 4.1: Il grafico rappresenta il terzo quadrante del piano cartesiano con la bisettrice primo-terzo quadrante. Su questa porzione di piano vengono disegnati i punti che rappresentano i documenti: quelli al di sopra della bisettrice verranno considerati appartenenti alla categoria.

4.2 La Vista

In questo progetto, la vista ha l'obiettivo di rendere visibile il modello e di permettere all'utente di agire su di esso nel modo precedentemente descritto. Essa è formata principalmente dai grafici che permettono di visualizzare la disposizione dei punti in base alla categoria scelta e in base al valore degli iperparametri scelti, e da un pannello da cui è possibile interagire per la scelta di questi ultimi.

4.2.1 Principi di progettazione di un'interfaccia grafica

Lo scopo di un'interfaccia è sicuramente quello di permettere all'utente di svolgere un lavoro velocemente e nella maniera più intuitiva possibile. La loro progettazione è una disciplina in continua evoluzione a partire dalla fine degli anni '60 con le prime rudimentali interfacce grafiche, passando per la comparsa del web, fino ad oggi. Tuttavia esistono delle norme, come l'ISO 9241 - *Ergonomics of Human System Interaction*, che espongono gli obiettivi per una loro realizzazione ideale, discutendo soprattutto sull'usabilità e sull'ergonomia. Di questa norma sono particolarmente rilevanti le parti 11 e 110, che trattano delle interfacce grafiche per terminali.

Si elencano qui i 7 punti della norma ISO 9241-110 (revisionata nel 2006, sostituendo la 9241-10) che definiscono i principi di dialogo tra uomo e computer:

- Adeguatezza al compito
- Autodescrizione
- Conformità alle aspettative dell'utente
- Adeguatezza all'apprendimento
- Controllabilità
- Tolleranza verso gli errori
- Adeguatezza alla individualizzazione

L'ISO 9241-11, invece, definisce l'usabilità come:

"Il grado in cui un prodotto può essere usato da specifici utenti per raggiungere specifici obiettivi con efficacia, efficienza, soddisfazione in uno specifico contesto d'uso"

in cui:

- *Efficacia*: l'accuratezza e la completezza dei risultati raggiunti
- *Efficienza*: la quantità di risorse impiegate per raggiungere l'obiettivo
- *Soddisfazione*: il comfort e l'accettabilità del sistema di lavoro da parte degli utenti

In sintesi, secondo queste regole, un'interfaccia deve soddisfare i requisiti dell'utente a cui si rivolge in base a quello per cui è stata progettata di fare inoltre essa deve essere chiara da capire e semplice da imparare fin dall'inizio.

Ci sono però anche parecchi autori che discutono sull'argomento in questione. Tra i più famosi che hanno teorizzato delle linee guida per la progettazione di interfacce ci sono Norman [8] e Nielsen [9] il primo discute più che altro sulla progettazione in base all'utilizzo degli oggetti di tutti i giorni, applicando poi questi principi al computer, Nielsen invece, da esperto di interfacce web, dà un decalogo di regole da seguire in fase di progettazione di pagine web, noto con il nome di *10 Euristiche di Nielsen*:

1. Informare l'utente sullo stato del sistema
2. Prevedere un linguaggio semplice e naturale
3. Dare controllo e libertà di uscita all'utente
4. Essere coerenti e tenere conto degli standard
5. Prevenire gli errori
6. Favorire il riconoscimento piuttosto che il ricordo
7. Fornire delle scorciatoie per gli utenti più esperti
8. Adottare un estetica ed un design minimalista
9. Aiutare l'utente a riconoscere, diagnosticare e recuperare gli errori.
10. Fornire aiuto e documentazione

4.2.2 Strumenti utilizzati

Java Swing

Swing[1] è una libreria che permette la costruzione di interfacce grafiche compresa nel linguaggio di programmazione a oggetti Java a partire dalla versione 1.2. La prima versione di Java (1.0) prevedeva per le interfacce grafiche soltanto la libreria Abstract Window Toolkit (AWT). Tuttavia la visualizzazione e il comportamento di un programma che utilizzava tale libreria cambiava in base alla piattaforma su cui doveva girare, andando contro il principio di Java: *write once, run anywhere*. A causa di questi problemi, già nella versione 1.1 di Java viene introdotta Swing come libreria esterna, successivamente integrata nella versione 1.2, tuttavia non sostituendo AWT, ma basandosi su di essa. Swing, essendo scritto interamente in Java, diventa completamente indipendente dalla piattaforma, garantendo di essere visualizzato nello stesso modo in qualunque sistema operativo e permettendo anche di personalizzare l'aspetto grafico dell'interfaccia stessa. Swing è realizzato secondo un'architettura MVC appositamente modificata per via del fatto che

[1] http://java.sun.com/products/jfc/tsc/articles/architecture/

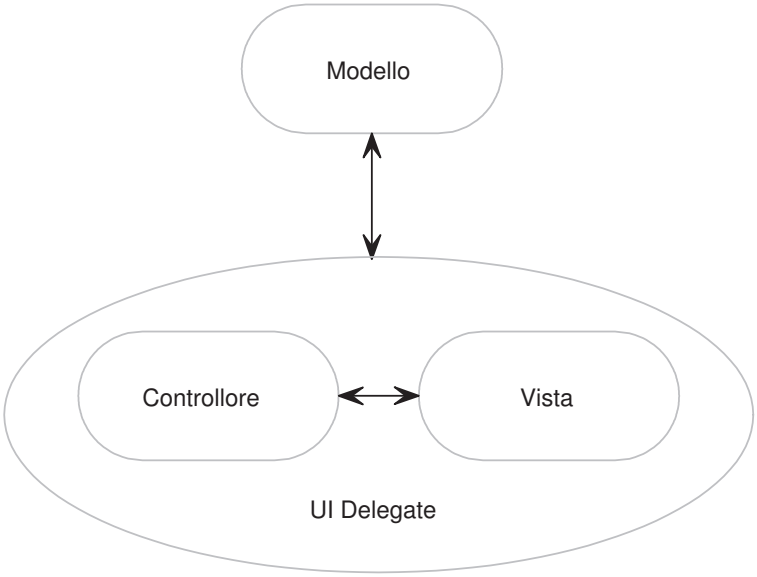

Figura 4.2: Rappresentazione grafica dello schema MVC utilizzato in Java Swing

risultava difficile fare un controller che non sapesse come fosse realizzata la vista. Perciò si è deciso di inglobare Vista e Controllore in un'unica entità nominata *UI Delegate* che gestisce gli eventi e la grafica, mentre il Modello viene mantenuto e ha la funzione di conservare le informazioni sullo stato.

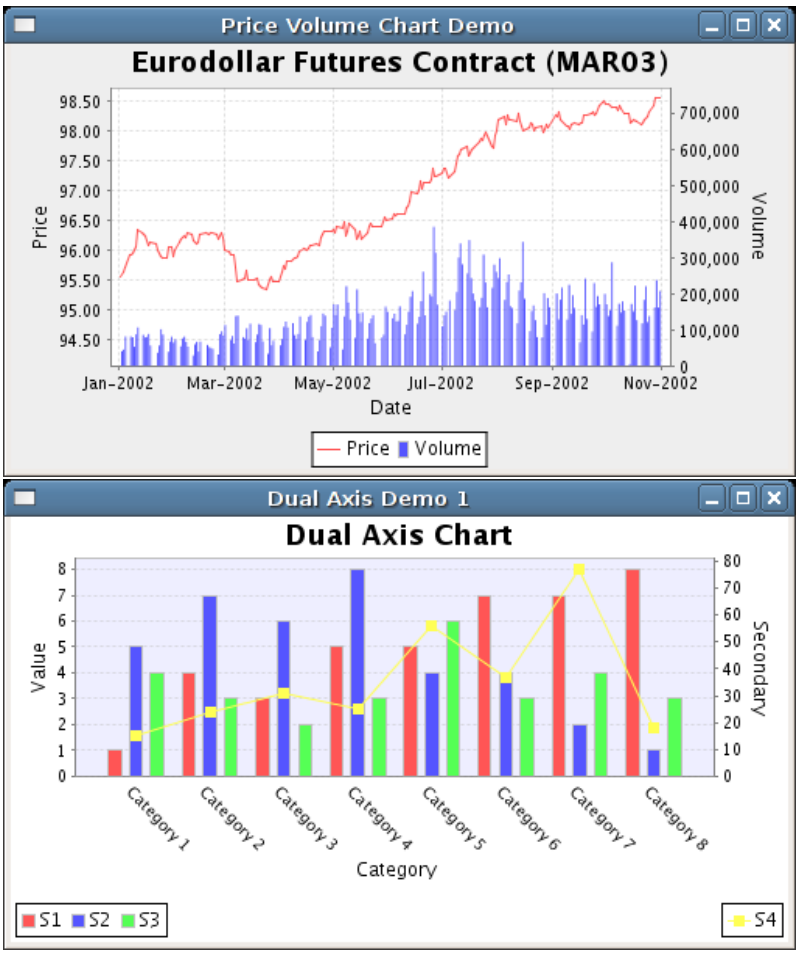

Figura 4.3: Due grafici di esempio della libreria JFreeChart

La libreria JFreeChart

JFreeChart[2] è una libreria esterna open source, scritta in Java, per la creazione di grafici a due e a tre dimensioni a partire da una serie di dati. La sua realizzazione è iniziata nel 2000 dallo sviluppatore David Gilbert, il quale segue tutt'ora il suo sviluppo.

La libreria JFreeChart permette di visualizzare: bar charts, line charts, pie charts, scatter charts, e molti altri tipi di grafici. Una caratteristica interessante di questa libreria è che i grafici vengono generati in runtime, quindi se vengono modificati dei dati durante l'esecuzione, questi ultimi vengono immediatamente ridisegnati sul grafico.

[2]http://www.jfree.org/jfreechart/

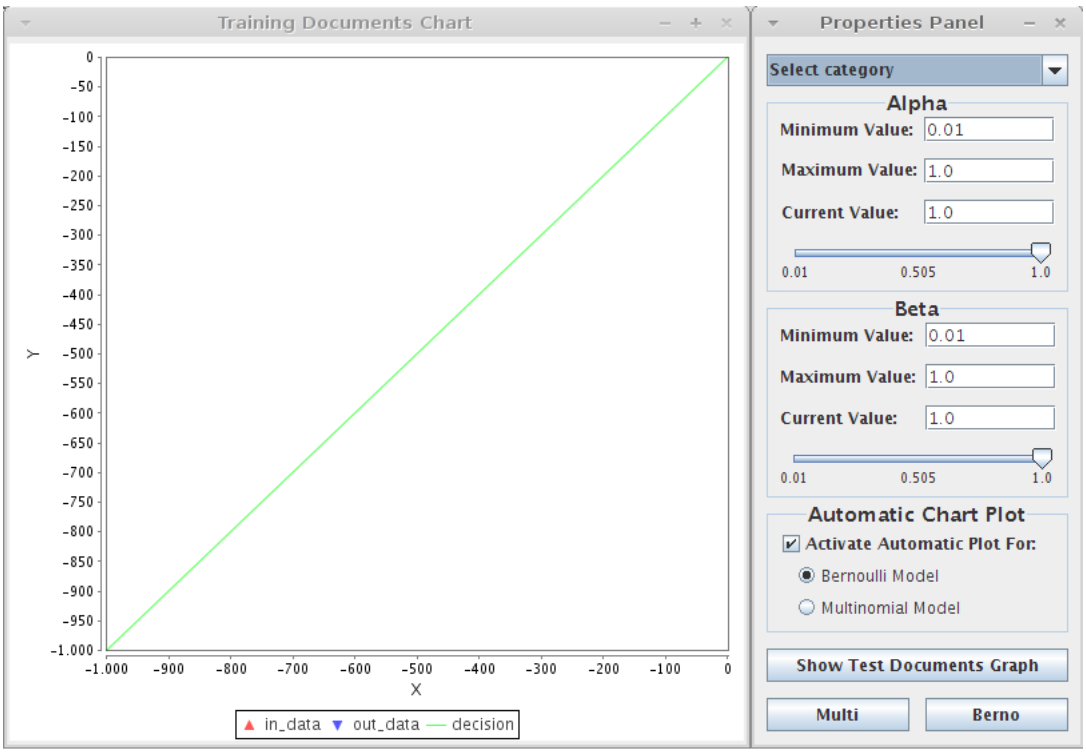

Figura 4.4: Schermata del programma appena avviato

4.2.3 Implementazione

Lo sviluppo del progetto è stato fatto interamente in Java, con le librerie Java Swing e JFreeChart, utilizzando l'ambiente di sviluppo Eclipse.

La realizzazione è stata fatta cercando di adempiere nel modo migliore i requisiti richiesti e anche di soddisfare il più possibile le linee guida elencate per la progettazione di un'interfaccia grafica, tenendo conto che l'applicazione è rivolta a ricercatori nell'ambito della classificazione dei testi, quindi ad una fascia di utenti esperti. Tuttavia si è cercato di rendere l'interazione più chiara ed essenziale possibile, in modo che chi esegue per la prima volta l'applicazione non si senta troppo inesperto di fronte ad essa.

All'avvio, il programma si apre al centro dello schermo un pannello con il grafico dei documenti di addestramento, chiamato *Training Documents Chart*, e alla sua destra è situato il pannello di controllo che implementa le seguenti funzioni: selezionare la categoria, regolare gli iperparametri del modello e disegnare i punti che rappresentano i documenti. Il grafico corrisponde a quanto teorizzato in precedenza relativamente al modello, ovvero esso è il terzo quadrante del piano cartesiano, le cui rispettive coordinate sono limitate tra 0 e -1000, con all'interno disegnata la retta bisettrice di decisione $y = x$. Per la sua implementazione è stato necessario realizzare un apposito frame (chiamato *CustomJFrame*), che contiene il grafico, in modo che quando l'utente ridimensiona o mette a tutto schermo la finestra, anche il grafico al suo interno si adatta alle dimensioni impo-

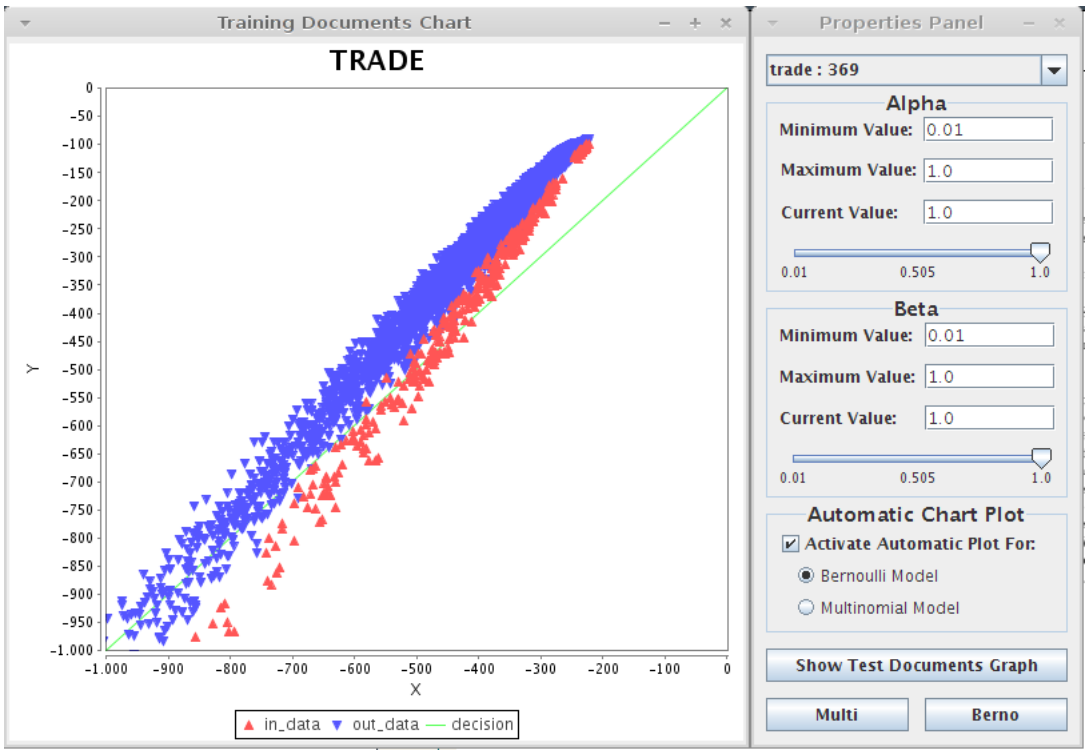

Figura 4.5: Schermata del programma in cui è stata scelta la categoria, mentre il valore dei parametri è restato quello di default e infine è stato premuto il pulsante *Berno* per disegnare il grafico relativo ai parametri scelti. Nel grafico, oltre alla retta di decisione, si possono notare due tipi di punti che corrispondono alle due categorie di documenti: *Rilevanti* e *Non Rilevanti*. I primi sono rappresentati dai triangoli rossi, mentre i secondi dai triangoli blu.

state. Inoltre, grazie a una funzionalità già implementata dagli sviluppatori della libreria JFreeChart, è possibile selezionare dei riquadri sul grafico al fine di fare uno zoom su di esso permettendo così di visualizzare la posizione di certi punti con maggiore dettaglio.

Il pannello intitolato *Properties Panel* è invece organizzato a zone. Nella parte più alta è situato un ComboBox, da cui è possibile scegliere la categoria di documenti da analizzare. Quando se ne seleziona una, sopra il grafico si imprime il nome della categoria dei documenti e, nel caso in cui ci sia già un grafico rappresentato, la scelta di una categoria comporta anche la cancellazione dei punti già presenti.

Al di sotto di esso sono situati i riquadri *Alpha* e *Beta* che, con i textBox e le slider al loro interno, permettono di impostare il range di modifica degli iperparametri e quindi di regolarne i valori. Una volta impostati il *Minimum Value* e il *Maximum Value*, viene svolto un controllo sui valori impostati nello slider e nel textBox *Current Value*, facendo in modo che il valore scelto in quest'ultimi sia appartenente al range ammesso: nel caso sia minore di quello minimo consentito,

viene cambiato con il valore impostato in *Minimum Value*. Viceversa, nel caso sia maggiore di quello massimo viene cambiato con il valore dato da *Maximum Value*. Inoltre, una volta impostato il valore di un parametro scelto sul textBox *Current Value*, questo viene segnalato al modello tramite il controller che poi avviserà del cambiamento anche lo slider secondo il meccanismo dello schema MVC. Viceversa se viene impostato un valore sullo slider, la modifica si ripercuote sul textBox del valore corrente nello stesso modo.

L'ultimo riquadro del pannello si chiama *Automatic Chart Plot* e implementa una scorciatoia per permettere la visualizzazione immediata sul grafico delle modifiche fatte ai parametri: se il checkBox al suo interno è selezionato, viene ridisegnato il grafico non appena il valore corrente di Alpha o di Beta viene cambiato, secondo il tipo di modello scelto dai radio buttons indentati immediatamente sotto. Se viene scelto *Multinomial Model*, poiché esso dipende solamente da alpha, tutto il contenuto del riquadro *Beta* viene disabilitato. Invece nel caso in cui il checkBox non sia selezionato, i radio buttons si disabilitano. Queste inibizioni dell'interfaccia grafica permettono all'utente di focalizzare l'attenzione soltanto su ciò che è necessario.

Infine, all'estremo inferiore del pannello, sono presenti tre pulsanti: due intitolati *Multi* e *Berno* che servono per disegnare il grafico rispettivamente del modello Multinomiale e Bernoulli, e uno più grande chiamato *Show Test Documents Graph*, che ha la funzione di aprire un secondo grafico uguale come aspetto a quello dei documenti di addestramento, ma con la differenza che serve a rappresentare i documenti di test. Quando è aperto, ogni modifica fatta nel pannello ai parametri alfa e beta si ripercuote su entrambi i grafici visualizzati. Inoltre il pulsante *Show Test Documents Graph* si rinomina in *Hide Test Documents Graph* e permette di richiudere il grafico. Invece nel caso in cui questo non sia aperto quando l'utente traccia il grafico, il programma non ridisegna anche i punti del *Test Documents Chart*, ma solo quelli del *Training Documents Chart*. Così facendo, il disegnare un nuovo insieme di punti diventa più veloce poiché lo fa per un grafico soltanto.

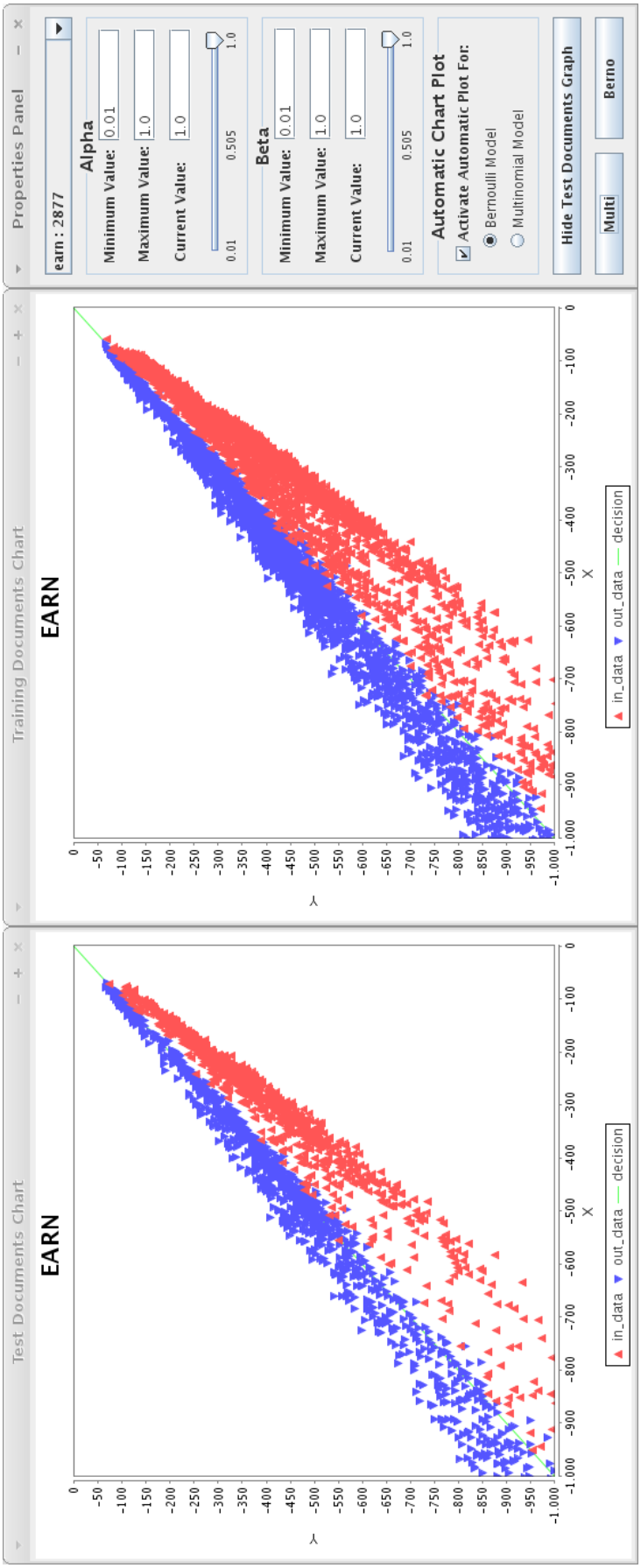

Figura 4.6: Schermata in cui è stata scelta una categoria e premuti rispettivamente il tasto *Multi* e il tasto *Show Test Documents Graph*: si possono infatti notare i due grafici contemporaneamente aperti.

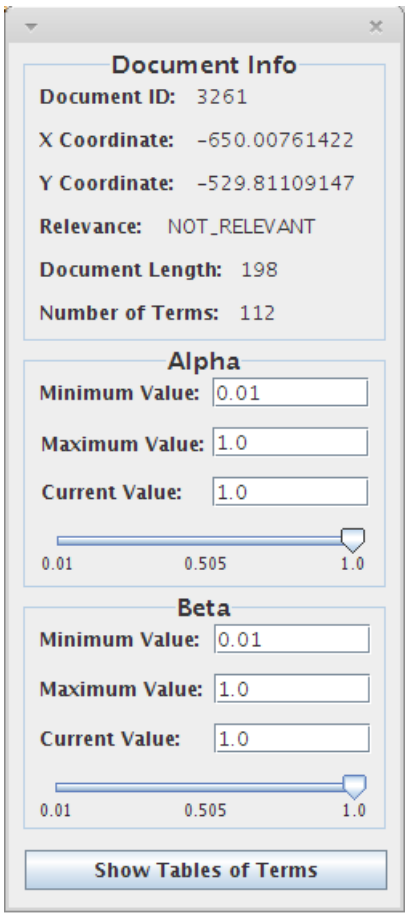

Figura 4.7: Una volta cliccato su un punto del grafico, compare questo pannello, da cui è possibile visualizzare le informazioni sul documento e modificarne i singoli parametri alpha e beta.

Una interessante funzionalità implementata sul grafico è che cliccando su di un suo punto compare un pannello dello stesso stile del *Properties Panel*, anch'esso organizzato a riquadri. Il primo riquadro a partire dall'alto si chiama *Documents Info* e al suo interno raccoglie tutte le informazioni riguardanti il documento selezionato sul grafico, che consistono in: ID del documento, coordinate del punto, rilevanza, lunghezza del documento e numero di termini. Al di sotto sono situati due riquadri *Alpha* e *Beta* identici a quelli del pannello principale del programma, che però servono a modificare i parametri del singolo documento considerato. In fondo c'è un pulsante, chiamato *Show Tables of Terms*, che ha lo scopo di far vedere le tabelle dei termini contenuti nel documento con a fianco il numero di volte in cui compare tale termine nel testo. Se le tabelle sono parecchie, viene visualizzata una scrollBar orizzontale per poterle scorrere tutte, limitando così orizzontalmente la dimensione del pannello nel caso di un documento con molti termini. Infine ripremendo lo stesso pulsante, che ora ha preso il nome di *Hide Tables of Terms*, il pannello ritorna al suo stato iniziale nascondendo le tabelle dei termini.

Inizialmente l'ultimo pannello descritto non era stato previsto. Al suo posto quando si sostava sopra ad un punto del grafico con il puntatore, compariva un tooltip, scritto in linguaggio HTML[3], con le stesse informazioni. Inoltre, cliccando sullo stesso punto, il tooltip veniva esteso con le informazioni relative alle tabelle dei termini. Tuttavia questa funzionalità[4] presente nella libreria JFreeChart richiedeva, oltre alla struttura dati dei punti, una lista parallela di stringhe in cui ognuna rappresentava un tooltip. L'inconveniente è che la lista doveva essere inizializzata contemporaneamente all'insieme dei punti a cui apparteneva, pena la non corrispondenza di un tooltip con il documento a cui faceva realmente riferimento. Ciò portava ad un aggravio delle prestazioni ogniqualvolta si doveva disegnare il grafico.
Inoltre, su entrambi i pannelli, è possibile scorrere dall'alto verso il basso con il tasto di tabulazione per poter interagire con le varie componenti senza utilizzare il puntatore.

[3]Hyper-Text Markup Language - http://www.w3.org/html/wg/
[4]www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/labels/CustomXYToolTipGenerator

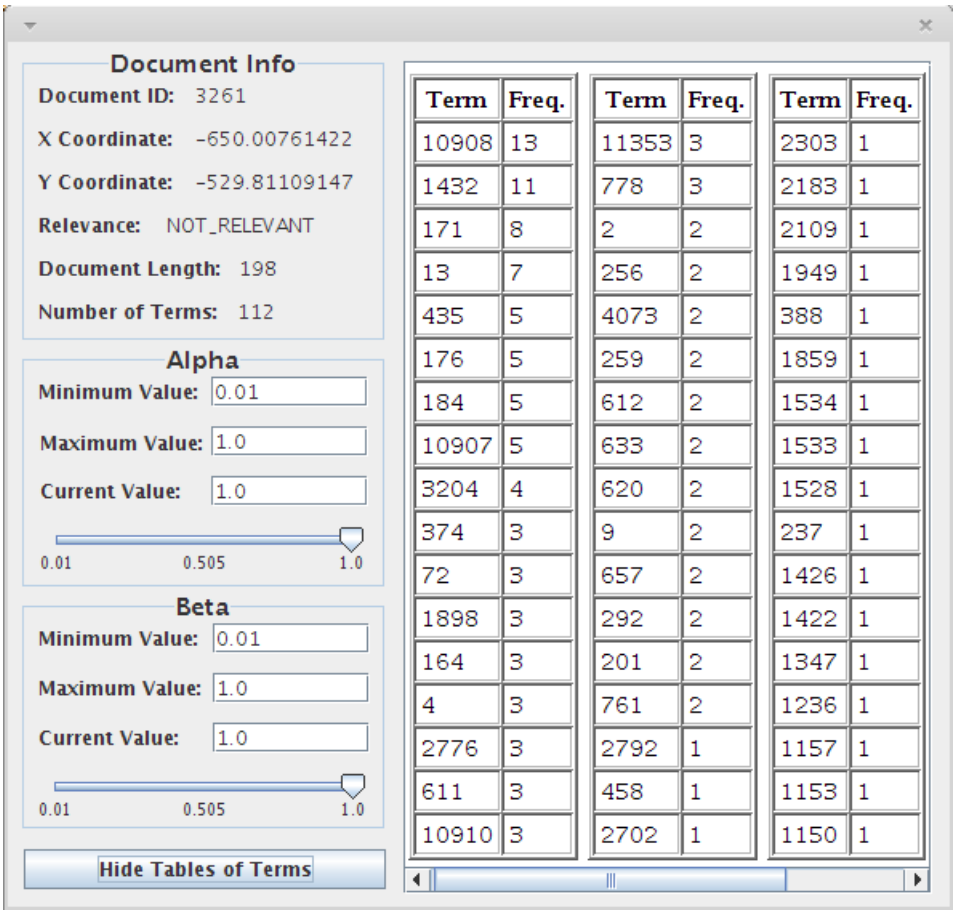

Figura 4.8: Lo stesso pannello dell'immagine precedente a cui è stato premuto il bottone *Show Tables of Terms*.

Capitolo 5

Conclusioni

La finalità di questa tesi è stata sviluppare un software che permettesse a utenti ricercatori di vedere in maniera immediata le conseguenze dovute alla scelta degli iperparametri nel modello probabilistico, in modo da rendere più esatta possibile la classificazione dei documenti. Esso è stato implementato usando il pattern MVC per permettere una certa modularità e quindi rendere ogni futura modifica o aggiunta al programma più semplice, In particolare, anche la vista è realizzata in maniera modulare, così da permettere di aggiungere, togliere o modificare le finestre agevolmente. Il tutto è stato implementato in Java, utilizzando le librerie Swing e JFreeChart, e l'interfaccia grafica è stata costruita cercando di seguire le regole precedentemente elencate di buona progettazione per l'interazione uomo-computer. Infatti l'interfaccia è stata realizzata in maniera semplice ed essenziale, facendo comunque in modo che essa fosse il più possibile chiara all'utente fin dal primo utilizzo. Come tutte le interfacce, però, è soggetta a modifiche: per esempio si può modificare per renderla magari più veloce per certe operazioni di routine svolte dall'utente, incrementando il numero di scorciatoie oppure semplicemente spostando alcuni comandi, aumentandone così l'efficienza d'uso.

Nonostante le caratteristiche implementate, anche le funzionalità che possono ancora essere aggiunte sono molteplici. Ad esempio, a partire dal pannello di modifica dei parametri α e β di un singolo documento, una delle cose che si potrebbe implementare sarebbe quella di far vedere come si modifica il grafico, e quindi la classificazione dei documenti, pesando in maniera differente ogni singolo termine appartenente ad una categoria.

Bibliografia

[1] Eckstein, R.: Java SE Application Design With MVC, http://www.oracle.com/technetwork/articles/javase/mvc-136693.html, (2007)

[2] Buschman, F., Meunier, R., Heuner, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture - A System of Patterns, Wiley, (2001)

[3] Articolo: A Smooth View of Text Categorization

[4] Ross, S.: Calcolo Delle Probabilità, Apogeo, (2008)

[5] Sebastiani, F.: Machine Learning in Automated Text Categorization, ACM Computing Surveys, (2002)

[6] Krug, S.: Don't Make Me Think, Hops-Tecnologie, (2006)

[7] Tidwell, J.: Designing Interfaces, O'Reilly Media, (2010)

[8] Norman, D.: Designing of Everyday Things, (2002)

[9] Nielsen, J.: Usability Engineering, Morgan Kaufmann, (1994)

[10] Fowler, A.: A Swing Architecture Overview, http://java.sun.com/products/jfc/tsc/articles/architecture/

[11] HTML wiki: http://www.w3.org/community/webed/wiki/HTML/Elements

[12] Sito ufficiale JFreeChart: http://www.jfree.org/jfreechart/