

Applicazione di tecniche di Information Retrieval per la classificazione di documenti

Progetto di Pattern Recognition, A.A. 2007/2008

Nome software: DocumentClassifier

Autore: Salvo Danilo Giuffrida (B05/000145)

1 Problema

I problemi che il software sviluppato nell'ambito di questo progetto tenta di risolvere sono i seguenti:

1.1 Problema della classificazione di documenti

Dato un training set composto da N documenti pre-classificati, e suddiviso in M categorie (che possono anche sovrapporsi, in quanto uno o più documenti del training set possono appartenere a più di una categoria; per esempio una notizia del sito web dell'ANSA può essere di cultura ma anche di cinema), e dato in input un documento non classificato, denominato “*documento query*”, determinare la sua categoria tra le M previste dal training set.

Per fare ciò, si usano algoritmi di information retrieval, per calcolare la similarità di ogni documento (pre-classificato) del training set con il documento query, e successivamente si applica l'algoritmo di classificazione dei K Nearest-Neighbours (K -NN^[1]) per stimare la categoria del documento query dalla categoria più frequente tra quelle dei K documenti ad esso più simili.

E' necessario puntualizzare che, il fatto che il documento da classificare venga denominato “*query*” non è casuale, perché in realtà una parte del sistema sviluppato si comporta come un motore di ricerca, con la differenza che in questo caso la query, anziché essere formata da poche keywords significative, è formata dal testo intero di un documento.

1.2 Determinare quale tra le seguenti metriche è la migliore per il calcolo della similarità tra documenti

- TF-IDF (Term Frequency-Inverse Document Frequency)

- Distanza di Bhattacharrya

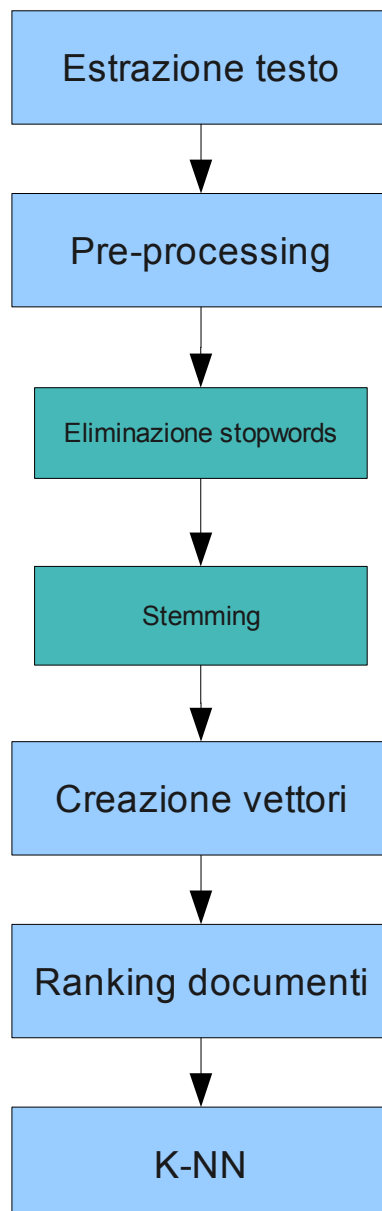
1.3 Determinare il valore ottimale dell'unico parametro dell'algoritmo di classificazione: Il valore di K per l'algoritmo di K-NN

Il valore ottimale rende minimo l'errore di classificazione.

Nella sua forma attuale, il programma riesce a classificare notizie del sito web dell'ANSA^[2], leggendole da file HTML locali o direttamente dal sito web, rispetto ad un training set su disco (sempre di notizie del sito web dell'ANSA), ma nel seguito si farà riferimento a documenti generici, in quanto esso è stato sviluppato in modo da renderlo facilmente estensibile con plug-ins.

2 Algoritmi

Partendo dal presupposto che, nel dominio della classificazione dei documenti, le features di ogni elemento da classificare sono le frequenze relative di ogni suo termine, si presentano, in ordine di esecuzione, gli algoritmi utilizzati dal software durante le varie fasi del processo di classificazione:



2.1 Pre-processing del testo

Le seguenti due operazioni vengono applicate prima di ogni altra operazione sul testo di ogni singolo documento, ed hanno lo scopo di ridurre la quantità di falsi negativi durante la successiva fase di Information Retrieval, nonché di ridurre la dimensione dell'insieme dei termini che occorrono in tutto il training set e nella query, aumentando così le performance del sistema.

2.1.1 Eliminazione stopwords

E' consigliato eliminare dal testo di ogni documento tutte quelle parole di uso comune, dette "stopwords", come articoli, verbi di uso frequente ("essere", "avere", ecc..), pronomi, e così via...

Questo perché esse, essendo di uso comune, sono presenti in tutti i documenti,

costituendo quindi una sorta di “rumore” rispetto agli altri termini, che invece contribuiscono a formare il contesto di riferimento di ogni documento, il quale permette di riconoscere documenti simili tra loro. L'eliminazione di tale rumore aiuta anche a migliorare la velocità di esecuzione della fase successiva, in quanto riduce la dimensione di ogni singolo documento, e quindi il numero delle componenti del suo vettore.

Poiché ogni lingua ha un'insieme di stopwords differente, è necessario avere a disposizione un apposito dizionario per ognuna di esse.

2.1.2 Stemming

All'interno di un documento, o tra documenti differenti, possono sussistere termini simili, che esprimono lo stesso concetto, ma sono declinati in modo diverso: Si pensi ad esempio ai sinonimi, o alla varie forme di un verbo (che, in lingue come l'Italiano, sono diverse da persona a persona), oppure ancora alle differenze tra le declinazione femminile e quella maschile di un sostantivo. A questo scopo, per evitare che termini con lo stesso significato vengano considerati diversi (falsi negativi), dopo l'eliminazione delle stopwords, è consigliato applicare al testo di ogni documento un'algoritmo di stemming, implementato per ogni specifica lingua che si vuole supportare, e che trasforma ogni termine nella sua radice, secondo una relazione n:1, dato che a più termini può corrispondere la stessa radice (per es.: *abbandonata*, *abbandonate*, *abbandonati*, *abbandonato*, *abbandonava*, *abbandonerà*, *abbandoneranno* e *abbandonerò* vengono tutti ridotti in *abbandon*).

2.1.3 Creazione vettori

Dopo aver pre-processato il testo di tutti i documenti del training set, e del documento query corrente, per ciascuno di essi i singoli termini vengono letti ed indicizzati, e le rispettive frequenze formano un vettore associato a ciascun documento.

2.1.4 Information retrieval (Ranking)

La classificazione del documento query corrente viene effettuata applicando l'algoritmo dei K Nearest-Neighbors ai K documenti del training set più vicini al documento query corrente. In quest'ottica, quindi, è necessario un sistema per determinare la vicinanza tra due documenti (nel nostro caso il documento query ed ogni singolo documento del training set).

A tale scopo, si utilizza uno tra i seguenti indici di similarità, ciascuno dei quali si basa su una certa metrica, che assegna ad ogni termine di ogni documento un

certo peso:

2.1.4.1 Distanza di Bhattacharrya

Misura la distanza tra due distribuzioni di probabilità su valori discreti, con un valore compreso tra 0 (minima distanza, alta similarità) ed 1 (massima distanza, bassa similarità).

Nel nostro caso tali distribuzioni di probabilità sono rappresentate dai vettori delle frequenze relative dei termini di ogni documento, i quali possono anche essere visti come istogrammi.

La formula per calcolarlo è la seguente:

$$\sqrt{1 - \sum_{x \in X} \sqrt{p(x) \times q(x)}}$$

*Distanza di Bhattacharrya tra due documenti p e q.
X è l'insieme dei termini comuni a p ed a q*

Che rappresenta la radice quadrata del complemento ad 1 del prodotto scalare dei due vettori associati ai documenti p e q, in cui però il peso di ogni termine è dato dalla radice quadrata della sua frequenza relativa. Se un termine x occorre in p ma non in q, o viceversa, il risultato del prodotto $p(x) \times q(x)$ è 0, quindi X è l'insieme dei termini comuni a p e q.

2.1.4.2 Vector Space Model/Cosine^[3]

Con il termine “*Vector Space Model*” (abbreviato in VSM), si indica un modello matematico che rappresenta documenti appartenenti ad un certo insieme come vettori in uno spazio vettoriale V-dimensionale, in cui ciascun componente corrisponde ad un termine del document set, pesato secondo una certa metrica. La dimensione dello spazio vettoriale è V, data dal numero di termini distinti che occorrono all'interno di tutto il document set, considerando quindi l'unione degli insiemi di termini di ogni singolo documento.

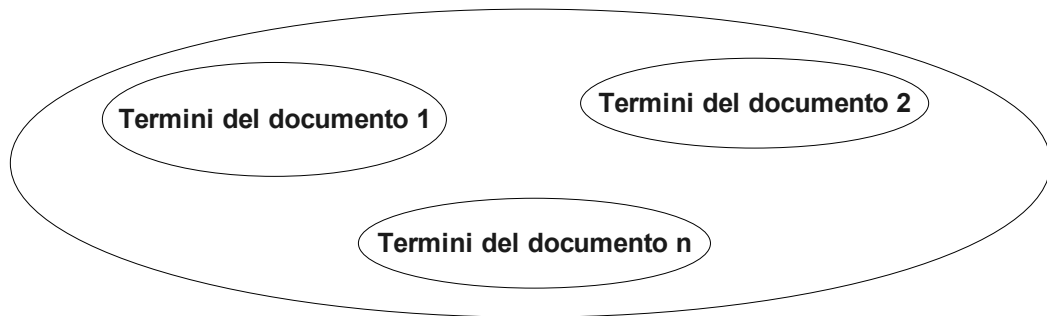


Figura 1: Unione dei termini di ogni documento

La metrica più comune associata a tale modello è la “*TF-IDF*” (acronimo di “*Term Frequency-Inverse Document Frequency*”), che, a differenza della frequenza relativa, la quale assegna un peso maggiore ai termini maggiormente frequenti all'interno di ogni singolo documento, tiene conto anche della frequenza di ogni termine in tutti gli altri documenti dell'insieme (training set+documento query corrente):

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i$$

*Valore di TF-IDF del termine i (t_i) all'interno del documento j (D_j)=Valore di "**Term Frequency**" del termine i all'interno del documento j (informazione locale) x Valore di "**Inverse Document Frequency**" del termine i in tutto il document set (informazione globale)*

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

*Valore di "**Term Frequency**" del termine i (t_i) all'interno del documento j=*Frequenza relativa

$$\text{idf}_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

Valore di "**Inverse Document Frequency**" del termine i (t_i) all'interno del document set=Logaritmo (in base 10) del rapporto tra la dimensione del document set ($|D|$), ed il numero di documenti nella quale è presente t_i , ovvero logaritmo dell'inverso della probabilità di trovare all'interno del document set un documento contenente t_i . Il logaritmo è usato per ridurre la crescita della funzione all'aumentare della dimensione del document set ($|D|$) o al diminuire del denominatore, in quanto ciò che è importante per il modello VSM è differenziare tra i seguenti casi:

- Un termine occorre in 1 documento ($\log(|D|)$)
- Un termine occorre in 2 documenti ($\log(|D|/2)$)
- Un termine occorre in molti documenti ($\log(|D|/n)$, $n < D$)
- Un termine occorre in moltissimi documenti ($\log(|D|/n)$, $n \approx |D| \Rightarrow \approx \log(1) = 0$)

Non è quindi importante differenziare se un termine occorre in n o in $n+1$ documenti, perché, per come è strutturato il calcolo del peso di ogni termine, tale differenza non è significativa

Per esempio, ecco alcuni casi possibili, ed i corrispondenti di valori di IDF:

$ D $	n	$ D /n$	$\log_{10}(D /n)$	Commento
1000	1	1000	3	Una parola che occorre in un solo documento su 1000 ottiene un valore alto di IDF
1000	100	10	1	Parola che occorre nel 10% dei documenti \Rightarrow Importanza media
1000	500	2	0.30103	Parola che occorre nel 50% dei documenti \Rightarrow Poco importante \Rightarrow IDF bassa
1000	10000	1	0	Parola che occorre in tutti i documenti \Rightarrow Non deve neppure essere presa in considerazione

In questo modo si utilizza sia un'informazione locale (la "Term Frequency"), che una globale (la "Inverse Document Frequency") per determinare il peso di un termine all'interno di un documento, peso che andrà poi ad influenzare il risultato del calcolo dell'indice di similarità con ogni altro documento dell'insieme (training set+documento query corrente).

Quindi, un termine che ha una frequenza alta all'interno di un documento, ma occorre anche in molti altri documenti, ottiene un valore di TF-IDF basso, in quanto viene considerato di uso comune (alla stregua di una stopwords), e quindi non contribuisce a discriminare un documento da un'altro. Viceversa, un termine molto frequente in singoli documenti, ma solo in pochi, ottiene un alto valore di TF-IDF, in quanto si tratta probabilmente di un termine tecnico, o comunque specifico del contesto a cui fanno riferimento i documenti in cui esso occorre.

Ricapitolando:

	Document Frequency bassa	Document Frequency alta
Term frequency bassa	Peso medio (termine poco frequente sia localmente che globalmente⇒Non abbastanza discriminante)	Peso basso (termine poco frequente localmente, ma molto globalmente⇒Non abbastanza discriminante)
Term frequency alta	Peso alto (termine molto frequente localmente, ma non globalmente⇒Determina il contesto del documento⇒Discriminante)	Peso medio (termine molto frequente sia localmente che globalmente⇒Non abbastanza discriminante)

Ovviamente se un termine non occorre in un particolare documento, il suo valore TF-IDF all'interno del corrispondente vettore è 0 (perché la frequenza locale è 0).

$$\mathbf{v}_d = [w_{1,d}, w_{2,d}, \dots, w_{N,d}]^T$$

Vettore TF-IDF del documento d: Le componenti 1, 2, ..., N sono i pesi TF-IDF dei termini 1, 2, ..., N del document set rispetto a d

L'ipotesi su cui si basa l'applicazione di questo modello alla risoluzione di problemi di Information Retrieval è che documenti semanticamente simili hanno i corrispondenti vettori vicini. A questo scopo, l'indice di similarità tra due vettori TF-IDF \mathbf{v}_1 e \mathbf{v}_2 è rappresentato dal coseno dell'angolo tra essi

$$\cos \theta = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\| \|\mathbf{v}_2\|}$$

In questo modo, un termine che in un vettore \mathbf{v}_D ha un valore di TF-IDF elevato (indice che esso è discriminante per il documento D rispetto a documenti di altre categorie), avrà un maggiore peso nel calcolo del coseno tra il vettore di cui esso fa parte, e qualunque altro documento/vettore.

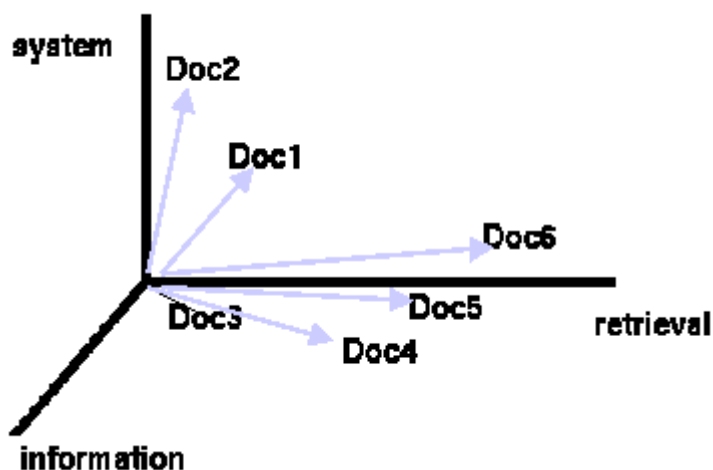


Figura 2: Rappresentazione di 6 vettori-documento in uno spazio 3-dimensionale. Solo 3 termini occorrono in tutto l'insieme, corrispondenti ai 3 assi cartesiani: 'Retrieval', 'Information' e 'System'.

Il valore di TF-IDF di ogni termine in ogni vettore/documento contribuisce al valore del coseno dell'angolo tra esso. Più due documenti sono simili, più il coseno dell'angolo tra i rispettivi vettori nel Vector Space Model sarà vicino a 1; viceversa, più due documenti sono dissimili, più il coseno di tale angolo è vicino a 0. In ogni caso esso non può essere negativo (tra $[-1,0[$), perché i due casi estremi sono:

1. Quando i documenti non hanno alcune termine in comune \Rightarrow Sono ortogonali tra loro \Rightarrow Il coseno è 0
2. Quando sono identici \Rightarrow Sono paralleli \Rightarrow Il coseno è 1

Qualunque sia l'indice di similarità adottato, i documenti del training set vengono ordinati dal più simile al meno simile rispetto al documento query.

2.2 Classificazione (K-NN)

Con l'ordinamento dei documenti del training set a disposizione, per classificare il documento query si applica l'algoritmo dei K-Nearest Neighbours all'insieme formato

dai primi K documenti del training set: La categoria a cui appartiene il maggior numero di documenti tra i K più vicini viene assegnata al documento query.

In caso di parità tra due o più categorie, viene assegnata quella a cui appartiene il documento più vicino.

2.3 K-Fold cross validation^{[4][5]}

Per poter valutare la qualità del sistema, e determinare il valore ottimale di K per il K-NN, viene utilizzato l'algoritmo della "*K-Fold cross validation*", il quale si compone delle seguenti fasi:

1 Per ogni possibile valore di K-NN (da 1 fino alla dimensione del training set corrente):

1.1 Dividere il training set in *K-Fold* partizioni di uguale dimensione, costruite estraendo documenti dal training set, in modo casuale o stratificato, ma sempre senza ripetizione.

1.2 Ripetere il processo di classificazione K-Fold volte, ciascuna volta tenendo fuori una delle partizioni dal training set, ed usandola come control set rispetto a tutte le altre $K-1$, che costituiscono il training set corrente.

1.3 Calcolare l'errore di classificazione per il valore corrente di K-NN.

2 Scegliere il valore di K-NN a cui corrisponde l'errore di classificazione minimo.

All'aumentare del valore di K-Fold, aumenta l'affidabilità della stima (perché il training set corrente rispetto a cui ogni documento del control set corrente viene classificato aumenta di dimensione), ma anche il costo computazionale dell'algoritmo (a causa della maggiore dimensione di ogni training set corrente, che impone un maggior numero di calcoli e confronti).

L'affidabilità massima si ottiene quando il numero di partizioni (dette anche "fold") è uguale alla dimensione del training set (n) \Rightarrow In tal caso ogni fold, e quindi ogni control set, è formato da esattamente 1 documento, e la dimensione del training set corrente ad ogni passo è la massima possibile ($n-1$). Questo caso particolare è detto "*Leave-one-out cross validation*", in quanto ad ogni passo un solo elemento viene eliminato dal training set per essere classificato.

Un buon compromesso tra affidabilità e costo computazionale, secondo la letteratura a riguardo, è il valore di $K=10$ ^{[5][6][7]}.

3 Uso del programma

Dopo aver spiegato quali algoritmi il sistema usa per effettuare la classificazione dei documenti in input, viene mostrato come essi sono stati implementati in un'applicazione Java, chiamata "*DocumentClassifier*", e come usare tale software.

3.1 Creazione training set

Per calcolare la similarità tra il documento query e tutti i documenti del training set, è necessario dapprima estrarne il testo.

All'avvio del programma, quindi, il training set viene letto da una directory su disco, e ne viene creata una rappresentazione in memoria, come insieme di sottoinsiemi di istanze della classe '*Documento*'.

Il percorso su disco della directory del training set viene specificato tramite l'interfaccia grafica, e di default è uguale alla sottodirectory '*TrainingSet*' della directory principale del programma.

In ogni caso, tale directory deve essere strutturata in modo da contenere una sottodirectory per ogni categoria, ed all'interno di ciascuna di esse vanno posti tutti i documenti appartenenti a tale categoria.

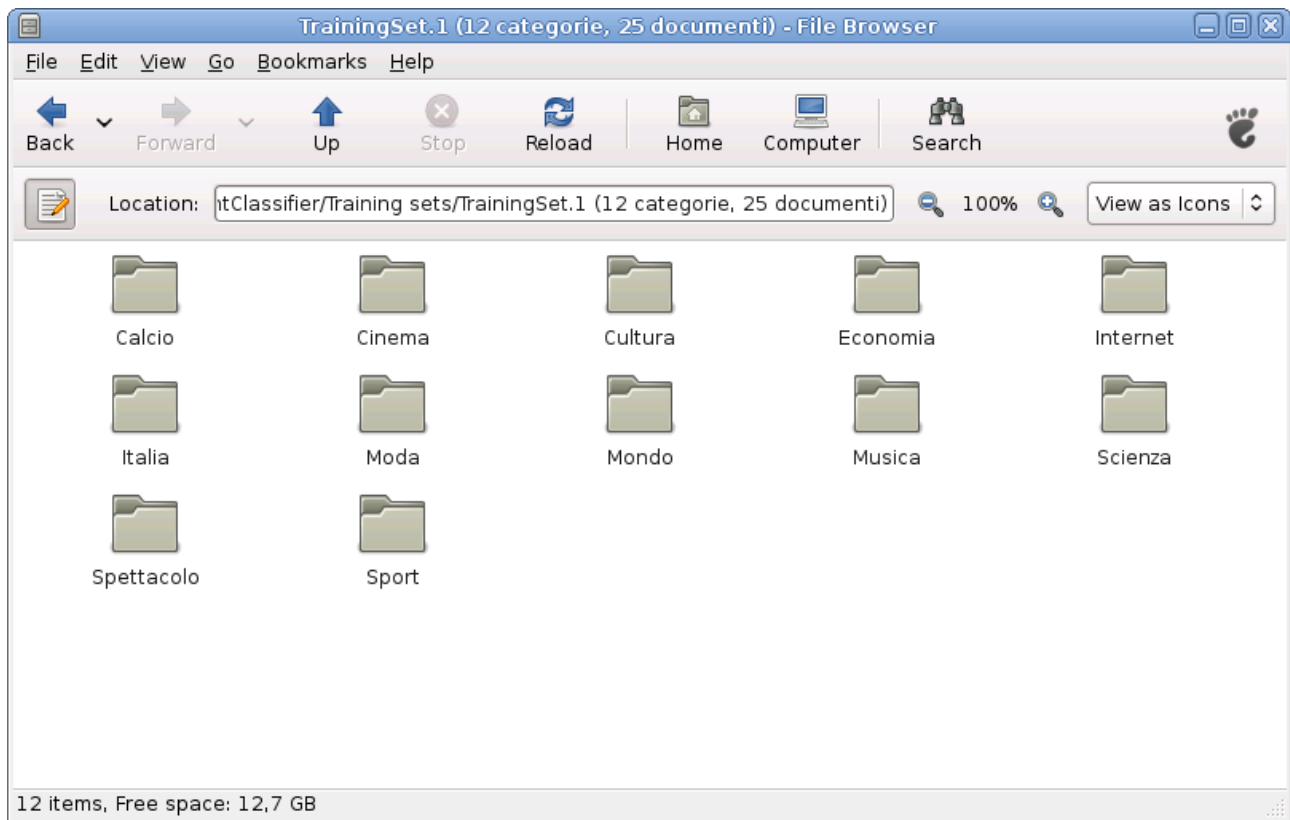


Figura 3: Training set composto da 12 categorie (12 sottodirectories), ciascuna delle quali contiene 25 documenti (in particolare notizie del sito dell'ANSA www.ansa.it). In fase di avvio del programma, viene creata una rappresentazione di questo in memoria, estraendo da ciascun documento il titolo ed il testo.

3.2 Interfaccia grafica

Una volta creato il training set, l'interfaccia grafica viene mostrata all'utente

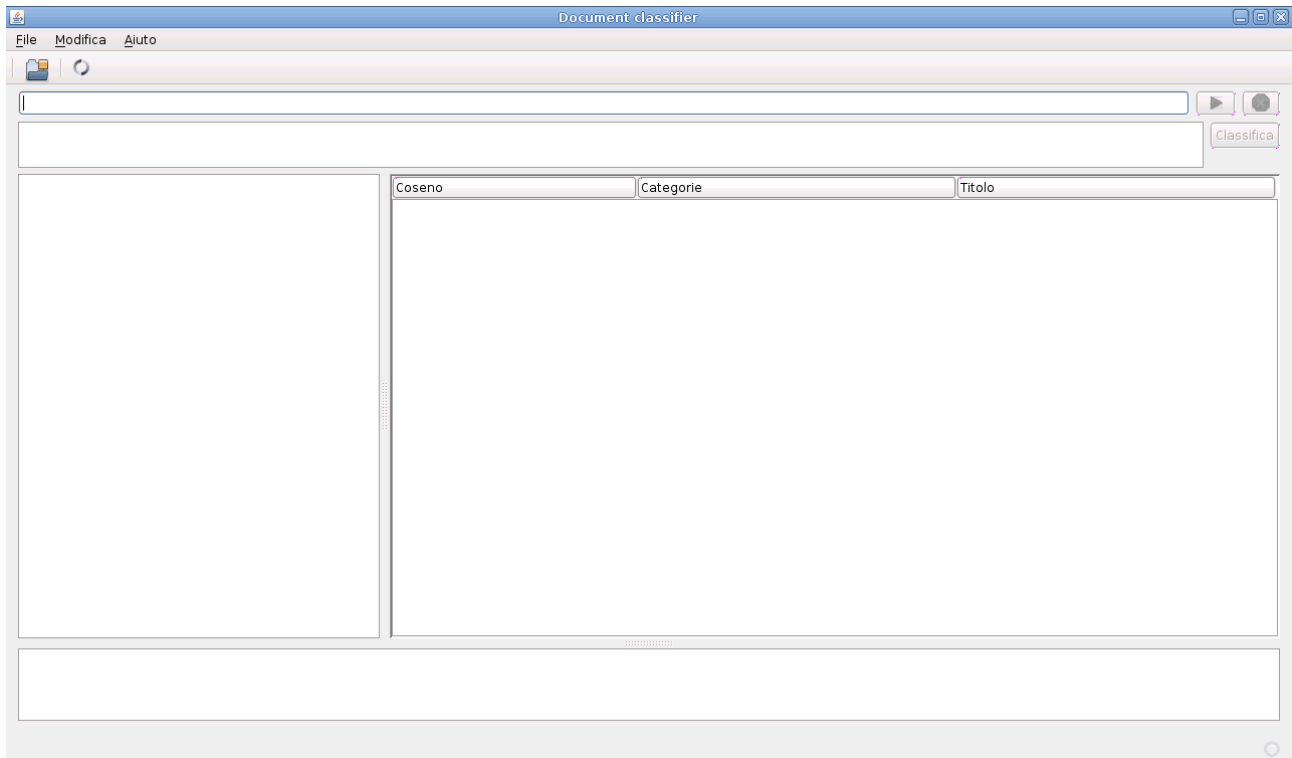




Figura 4: Screenshot dell'interfaccia grafica

Essa è composta da

- Una barra dei menù in alto
- Una toolbar con 2 icone per l'accesso rapido alla funzioni associate rispettivamente al menù 'File->Apri' ed al menù 'File->Validazione'
- Una serie di JComponents che occupano la maggior parte dello spazio

3.2.1 Menù e pulsanti

- **File->Apri/1° pulsante della barra delle icone:**  Apre un JFileChooser con la quale è possibile selezionare un documento su disco da classificare. L'indirizzo del file scelto verrà visualizzato nell'apposito JTextField dell'interfaccia grafica.
- **File->Validazione/2° pulsante della barra delle icone:**  Avvia la fase di validazione sul training set corrente. Durante tale fase viene visualizzato un JOptionPane con informazioni sullo stato corrente della computazione.
- **File->Esci:** Chiude il programma.
- **Modifica->Preferenze:** Apre il pannello di configurazione delle preferenze del programma, con la quale è possibile controllare in dettaglio il suo

funzionamento.

- **Aiuto->Guida utente:** Visualizza la guida utente.
- **Aiuto->About:** Visualizza informazioni sulla versione del programma e sul suo autore.

3.2.2 Componenti per l'Input/Output

Oltre alla barra dei menù, ed a quella delle icone, la maggior parte dell'interfaccia grafica dell'applicazione è composta da una serie di JComponents per fornire l'input o mostrare l'output delle varie funzioni del programma. Partendo dall'alto, tali componenti sono:

- Un *JTextField* nella quale inserire manualmente la URL del documento query da classificare, con alla sua destra un pulsante per estrarre il titolo ed il testo da tale documento



ed un pulsante per interrompere il task corrente



Tali pulsanti sono di default disabilitati, e cambiano stato, rispettivamente, solo se:

1. Il *JTextField* contenente la URL del documento query corrente non è vuoto.

Nel caso di indirizzi di rete, *http* o *https*, la loro validità viene controllata, nel senso che, prima di tentare di estrarre il titolo ed il testo dal documento specificato, viene controllato che la sua URL referenzi un elemento di rete effettivamente accessibile. In caso contrario viene visualizzato un messaggio di errore.

2. Il programma sta eseguendo uno tra i seguenti task:

- Estrazione titolo e testo dal documento query corrente
- Classificazione documento query corrente
- Validazione classificatore

Sotto tali componenti, è presente una *JLabel* in cui viene mostrato il titolo del documento query corrente, dopo averlo estratto premendo l'apposito pulsante. Accanto ad essa, è presente un pulsante dall'etichetta “*Classifica*” che, come suggerisce il nome, avvia il task di classificazione del documento query corrente. Al di sotto di essa, sono presenti due componenti che occupano la maggior parte dell'interfaccia grafica:

- Una *JLabel* dove invece viene mostrato il testo del documento corrente
- Una *JTable* in cui, dopo aver effettuato il ranking dei documenti del training set (parte del task di classificazione, avviato facendo click sul pulsante “*Classifica*” in alto a destra), viene visualizzata una lista di informazioni su ciascuno di essi (distanza dal documento query corrente, categorie di appartenenza, titolo), ordinata in base al rank.

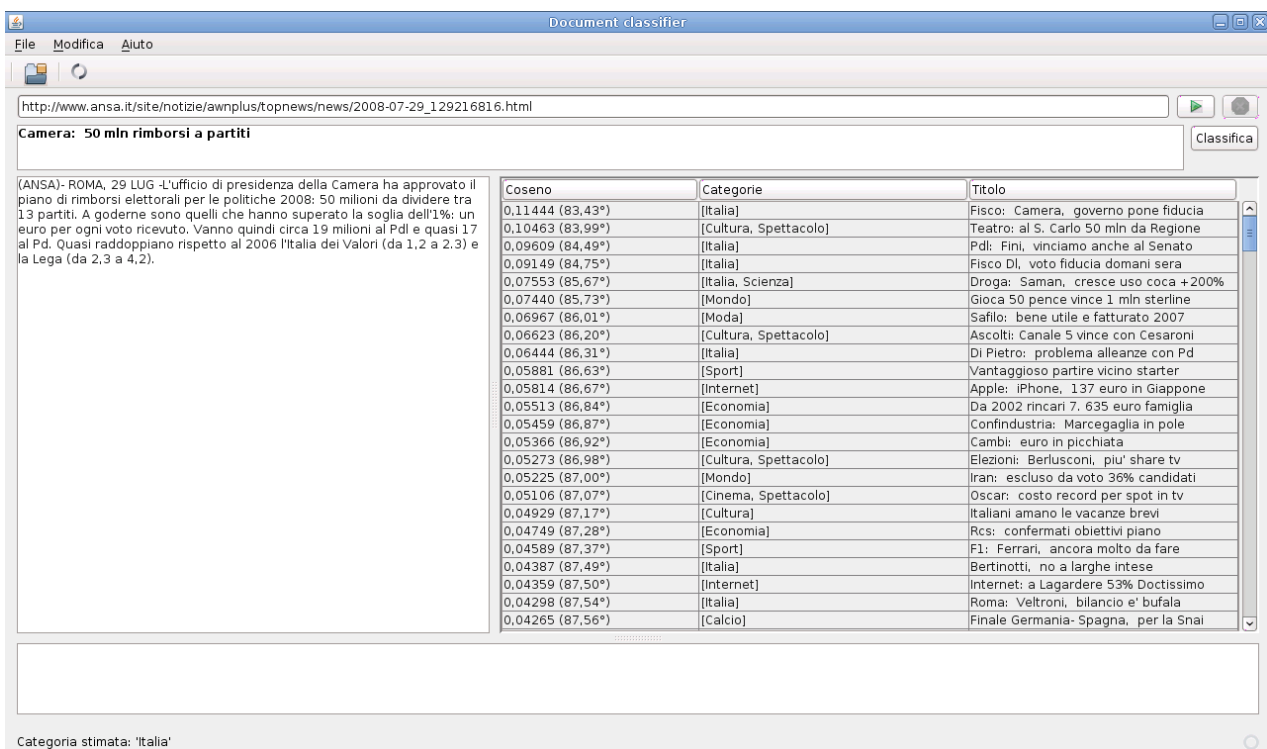


Figura 5: Ranking e classificazione di un documento query

Entrambi i componenti posso essere ridimensionati sia in orizzontale che in verticale, grazie all'aggiunta di due *JSplitPanel*, uno orizzontale ed uno verticale, nella gerarchia dei componenti del *JFrame* dell'applicazione.

Al di sotto dei due *JComponents* principali dell'interfaccia grafica, infine, è presente una *JTextArea*, non editabile, che durante la fase di validazione mostra messaggi sullo stato corrente della computazione.

Tali messaggi possono anche essere registrati su un file di log, specificato tramite il pannello di configurazione delle preferenze.

Infine, la barra di stato in basso mostra messaggi durante l'esecuzione di ognuno dei 3 task avviabili dall'utente (estrazione testo dal documento query, classificazione, validazione), nonché una *JProgressBar*, che, quando si trova in modalità “determinate”, mostra l'avanzamento del task di corrente.

3.3 Flusso di lavoro

Ricapitolando, per classificare un documento, l'utente deve seguire i seguenti passi:

1. Inserire nel JTextField in alto l'indirizzo (locale o di rete) del documento da classificare.
In alternativa, fare click sulla 1^a icona della barra delle icone, o sul menù **File->Apri**, e selezionare tramite l'apposito JOptionPane un file da disco
2. Una volta inserito l'indirizzo del documento da classificare nell'apposito campo di testo, premere il bottone caratterizzato dall'icona con la freccia verde



In questo modo viene avviato il task che tenta di estrarre il titolo ed il testo dal documento, e se esso termina in modo corretto, tali elementi vengono visualizzati nell'interfaccia grafica del programma

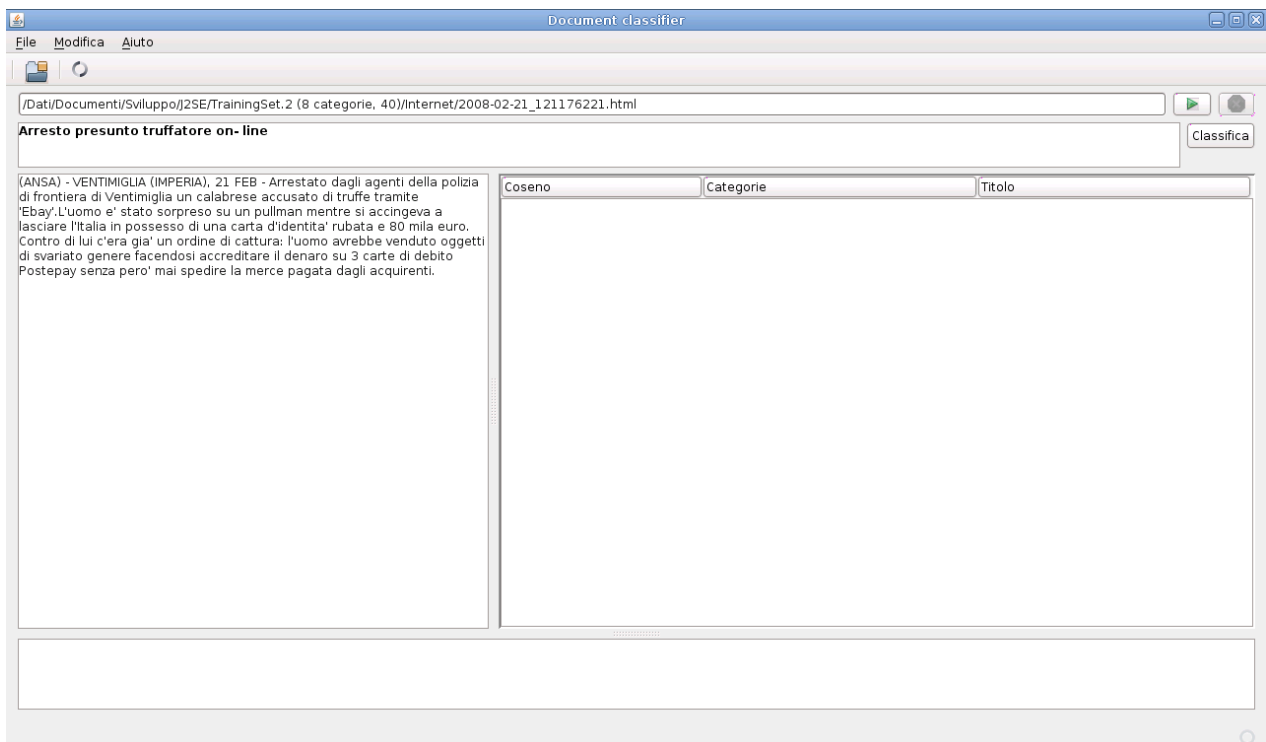


Figura 6: Lettura del documento query

- Una volta che il titolo ed il testo del documento query sono stati correttamente estratti, e visualizzati nelle apposite componenti dell'interfaccia grafica, è possibile classificarlo cliccando sul pulsante “**Classifica**”, che avvia il task che si occupa del ranking dei documenti del training set, secondo la metrica corrente (modificabile tramite il pannello di configurazione delle preferenze), e della classificazione tramite l'algoritmo K-NN (il cui parametro K è anch'esso modificabile tramite il pannello di configurazione delle preferenze).

Document classifier

File Modifica Aiuto

/Dati/Documents/Sviluppo/J2SE/TrainingSet.2 (8 categorie, 40)/Economia/2008-02-23_123177375.html

A Treviso benzinaio piu' low-cost

Classifica

(ANSA) - ROMA, 23 FEB - Gestione familiare, rifornimenti fatti autonomamente e lavoro duro: sono i motivi grazie ai quali vendeva benzina sotto la media. Lo spiega Galdino Gallina, il gestore piu' low-cost d'Italia: nelle sue 3 stazioni di servizio nel Trevigiano la settimana scorsa vendeva la verde a 1,208 euro quando i prezzi medi delle compagnie segnavano quota 1,375, offrendo cosi' risparmi fino a 16 centesimi al litro. A scovarlo e' stato il Codacons nell'elenco dei distributori indipendenti.

Coseno	Categorie	Titolo
1,00000 (0,00°)	[Economia]	A Treviso benzinaio piu' low-cost
0,17795 (79,75°)	[Economia]	Benzina: caro- autostrada, prima A-14
0,06364 (86,35°)	[Cinema, Spettacolo]	Oscar: costo record per spot in tv
0,05630 (86,77°)	[Economia]	Pil: cresce forbice Spagna- Italia
0,05330 (86,94°)	[Cultura, Economia, Spettacolo]	Cambi: euro poco mosso
0,05245 (86,99°)	[Internet]	Fidel Castro, sue riflessioni su web
0,05045 (87,11°)	[Cultura, Spettacolo]	Elezioni: Berlusconi, piu' share tv
0,04626 (87,35°)	[Economia]	Da 2002 rincarì 7. 635 euro famiglia
0,04275 (87,55°)	[Mondo]	Bosnia: Ue, no a indipendenza serba
0,04053 (87,68°)	[Cinema, Spettacolo]	Hugh Laurie: spendo solo per le moto
0,04042 (87,68°)	[Mondo]	Colombia: un oleodotto chiuso
0,03971 (87,72°)	[Economia]	Aerei: attenti a tutto- compreso
0,03795 (87,83°)	[Musica]	Police si scioglieranno per sempre
0,03773 (87,84°)	[Calcio]	Finale Germania- Spagna, per la Snai
0,03694 (87,88°)	[Cultura, Spettacolo]	Teatro: al S. Carlo 50 mln da Regione
0,03410 (88,05°)	[Calcio]	Incontro Abete- ct entro venerdi'
0,03379 (88,06°)	[Internet]	Boeing lascia la Borsa di Tokyo
0,03299 (88,11°)	[Internet]	Quattro ruote su telefonini 3 Italia
0,03299 (88,11°)	[Sport]	F1: Ferrari, ancora molto da fare
0,03295 (88,11°)	[Scienza]	Legge Bossi- Fini, non ai ricercatori
0,03241 (88,14°)	[Mondo]	Baghdad: colpi mortali su zona verde
0,03230 (88,15°)	[Mondo]	Turchia: 200 delitti d'onore/anno
0,03187 (88,17°)	[Economia]	Rcs: confermati obiettivi piano
0,03184 (88,18°)	[Cinema]	Ad asta immagini ultimo film Monroe

Categoria stimata: "Economia"

Figura 7: Ranking e classificazione del documento query

- Se si vuole invece effettuare la fase di validazione del sistema, per determinare il valore ottimale di K, ed il corrispondente errore di classificazione minimo, è necessario fare click sulla voce di menù **File->Validazione**, oppure sul 2° pulsante della barra delle icone.

Alla fine di tale task, il programma si autoconfigura in modo da utilizzare per il K-NN il valore di K a cui corrisponde l'errore di classificazione minimo.

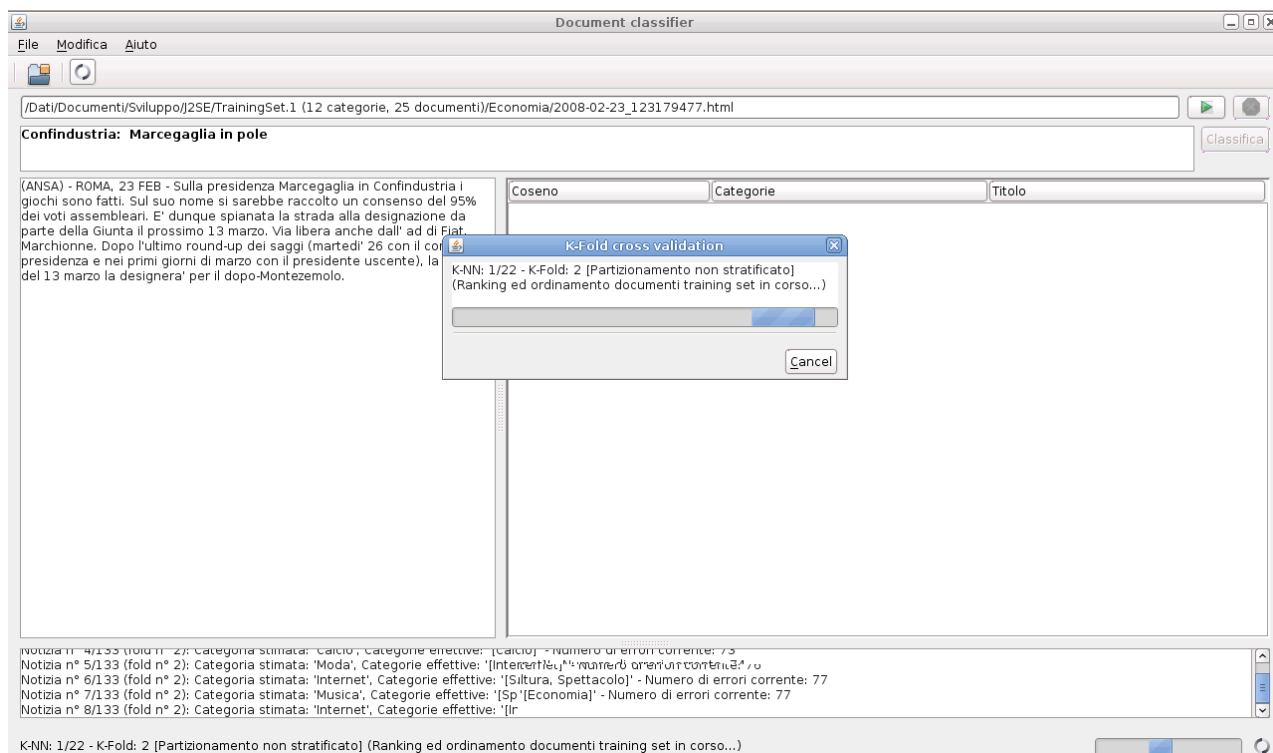


Figura 8: Validazione classificatore

4 Sviluppo

Il linguaggio utilizzato per lo sviluppo del software “*DocumentClassifier*” è Java, nella sua versione “*Standard Edition*”. Per poter essere eseguito esso necessita del pacchetto JRE dalla versione 5.0 in su (http://java.sun.com/javase/downloads/index_jdk5.jsp), in quanto alcune delle classi e delle funzionalità utilizzate sono state introdotte con la versione 5 (1.5) del linguaggio, rilasciata nel 2005, la quale ha rappresentato un notevole miglioramento rispetto alle precedenti, offrendo agli sviluppatori funzionalità come il “*Collections framework*”, per la gestione flessibile ma potenziata di strutture dati (insiemi, mappe e liste), o i generici, per il controllo a compile-time della type-safeness delle istruzioni di inserimento, lettura e modifica di elementi all'interno di tali strutture dati.

Inoltre, è stato utilizzato lo “*Swing Application Framework*” (<https://appframework.dev.java.net/>), un framework sperimentale che dovrebbe essere incluso nella prossima versione del linguaggio (7), e che semplifica lo sviluppo di applicazioni desktop in Java, soprattutto la gestione di task da eseguire in background. Le classi di tale framework sono comunque incluse nel programma, sia all'interno del file jar, che all'interno della sottodirectory 'lib' della directory contenente i file del progetto Netbeans, per cui sia l'utente che lo sviluppatore non necessitano d'installare nulla.

L'IDE utilizzato per lo sviluppo è infatti *Netbeans 6.1* (www.netbeans.org), scelto per la sua natura open-source, e per il suo supporto avanzato per Java (anche e soprattutto dello Swing Application Framework, nonostante la natura ancora sperimentale di quest'ultimo).

4.1 Architettura

4.1.1 Modularità

Durante la fase di sviluppo del software, dopo aver implementato le funzionalità di base, si è creduto necessario, o perlomeno utile, modificarne l'architettura, in modo da renderlo estendibile, ed utilizzabile per la classificazione di qualunque tipo di documento: non solo notizie del sito dell'ANSA, ma anche di altri siti, o di collezioni di file PDF, Word, ODT, ecc...

L'idea alla base di questo requisito, introdotto in una fase successiva dello sviluppo, dopo aver implementato i requisiti di base, è di rendere questo software un tool utilizzabile da chiunque abbia la necessità di classificare documenti rispetto ad un training set.

Per poter classificare altri tipi di documenti, oltre alle notizie del sito dell'ANSA, è necessario creare una classe all'interno del package '*documentclassifier.Scrapers*', che implementi l'interfaccia '*documentclassifier.Scraper.Scraper*' (maggiori informazioni sono disponibili nel JavaDoc allegato), in particolare il metodo '*getDocumento*' che, dato un *InputStream* da cui leggere un particolare tipo di documento, ne restituisce il titolo ed il testo (sotto forma di stringhe). Infatti, lo scraper per le notizie del sito dell'ANSA è esso stesso un plug-in, l'unico finora implementato e fornito. Esso utilizza un'analizzatore lessicale sviluppato con *Jlex* (<http://www.cs.princeton.edu/~appel/modern/java/JLex/>), ed uno sintattico sviluppato con *Cup* (<http://www2.cs.tum.edu/projects/cup/>), per analizzare il codice HTML di una pagina web, ed estrarne titolo e testo.

4.1.2 Stemming

Gli algoritmi di stemming per le lingue principali, selezionabili dal pannello delle preferenze, non sono stati implementati da me: E' stato invece utilizzato un package di algoritmi già pronti, implementati in Java usando il meta-linguaggio *Snowball* (<http://snowball.tartarus.org/index.php>), che consente di sviluppare algoritmi di stemming in modo dichiarativo, per i quali viene poi generato il corrispondente codice C, Java o Python.

Per il training set delle notizie del sito dell'ANSA, ovviamente lo stemmer da

utilizzare è quello per la lingua italiana.

5 Risultati

Con il programma vengono forniti sei training set costituiti da notizie presenti nel sito dell'ANSA nel periodo che va dal Marzo al Luglio 2008. Essi sono memorizzati in quattro directory, il cui nome segue il formato '*TrainingSet.n (x categorie, y notizie per categoria)*'. Vengono quindi di seguito presentati i risultati del processo di validazione rispetto a ciascun training set, con entrambe le metriche supportate (distanza di Bhattacharrya e TF-IDF/VSM), ed utilizzando sempre come valore di K per la K-Fold cross validation il massimo possibile, corrispondente alla dimensione del training set corrente. In questo modo infatti si ottiene la stima più affidabile per l'errore di classificazione (vedi paragrafo 2.3).

Nel file '*Grafici.ods*' (per OpenOffice Calc) sono presenti le rappresentazioni grafiche del processo di validazione, per ogni training set, e per ognuna delle due metriche supportate. L'ultimo foglio di lavoro di tale file contiene inoltre un confronto dell'andamento dell'errore di classificazione tra le due metriche, e tra i training set che contengono 25 e 50 notizie per categoria (12, 6 o 3 categorie).

5.1 Training set 1 (12 categorie, 25 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle seguenti categorie:

- Calcio
- Cinema
- Cultura
- Economia
- Internet
- Italia
- Moda
- Mondo
- Musica
- Scienza
- Spettacolo
- Sport

Per ognuna di esse, sono presenti 25 notizie, per un totale di 300 notizie (di cui 267 effettive, in quanto ci sono almeno 3 notizie che appaiono in più di una categoria). Di seguito sono riportati i risultati delle validazioni del classificatore:

- **TF-IDF**

Errore di classificazione minimo=35,21% (K-NN=40).

Analizzando con OpenOffice Calc il contenuto del file 'Grafici.ods', nel foglio di lavoro denominato "*TrainingSet_1_TFIDF*" si può notare che l'errore di classificazione, da un'iniziale 48,31%, tende a scendere all'aumentare di K, fino a raggiungere il minimo (35,21%) per K=40, per poi tendere a risalire.

- **Bhattacharrya**

Errore di classificazione minimo=46,07% (K-NN=8).

Analizzando con OpenOffice Calc il contenuto del file 'Grafici.ods', nel foglio di lavoro denominato "*TrainingSet_1_Bhattacharrya*", si può notare che l'andamento dell'errore di classificazione in relazione al valore di K per il K-NN è simile a quello della metrica TF-IDF, ma il valore minimo risulta maggiore, segno di una minore efficacia di questa metrica nel distanziare documenti dissimili.

5.2 Training set 2 (6 categorie, 25 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle seguenti categorie:

- Cultura
- Economia
- Italia
- Mondo
- Spettacolo
- Sport

Per ognuna di esse, sono presenti 25 notizie, per un totale di 150 notizie (di cui 137 effettive, in quanto ci sono almeno 3 notizie che appaiono in più di una categoria).

Di seguito sono riportati i risultati delle validazioni del classificatore:

- **TF-IDF**

Errore di classificazione minimo=27,74% (K-NN=5).

- **Bhattacharrya**

Errore di classificazione minimo=41,61% (K-NN=1).

5.3 Training set 3 (3 categorie, 25 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle seguenti categorie:

- Italia
- Mondo
- Sport

Per ognuna di esse, sono presenti 25 notizie, per un totale di 75 notizie (in questo caso non ci sono notizie che appartengono contemporaneamente a più di una categoria). Di seguito sono riportati i risultati delle validazioni del classificatore:

- **TF-IDF**

Errore di classificazione minimo=12% (K-NN=20).

- **Bhattacharrya**

Errore di classificazione minimo=20% (K-NN=3).

5.4 Training set 4 (12 categorie, 50 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle stesse categorie del precedente.

Per ognuna di esse, sono presenti 50 notizie, per un totale di 600 notizie (di cui 556 effettive, in quanto ci sono almeno 4 notizie che appaiono in più di una categoria). Di seguito sono riportati i risultati delle validazioni del classificatore:

- **TF-IDF**

Errore di classificazione minimo=30,22% (K-NN=37).

Si può notare che, confrontando i fogli di lavoro "*TrainingSet_4_TFIDF*" e "*TrainingSet_1_TFIDF*", aumentando la dimensione del training set da 25 notizie per categoria a 50 notizie per categoria, e mantenendo la stessa metrica, il valore minimo dell'errore di classificazione diminuisce (35,21% contro 30,22%), ed anche il valore corrispondente a parità di K tende ad essere minore.

- **Bhattacharrya**

Errore di classificazione minimo=40,83% (K-NN=51).

5.5 Training set 5 (6 categorie, 50 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle seguenti categorie:

- Cultura
- Economia
- Italia
- Mondo
- Spettacolo
- Sport

Per ognuna di esse, sono presenti 50 notizie, per un totale di 300 notizie (di cui 279 distinte, in quanto ci sono almeno 4 notizie che appaiono in più di una categoria). Di seguito sono riportati i risultati delle validazioni del classificatore:

- **TF-IDF**

Errore di classificazione minimo=26,52% (K-NN=29).

- **Bhattacharrya**

Errore di classificazione minimo=34,41% (K-NN=8).

5.6 Training set 6 (3 categorie, 50 notizie per categoria)

Le notizie presenti in questo training set sono suddivise nelle seguenti categorie:

- Italia
- Mondo
- Sport

Per ognuna di esse, sono presenti 50 notizie, per un totale di 150 notizie (analogamente al training set 3, anche in questo caso non ci sono notizie che appartengono contemporaneamente a più di una categoria). Di seguito sono riportati i risultati della validazioni del classificatore.

- **TF-IDF**

Errore di classificazione minimo=13,33% (K-NN=9).

- **Bhattacharrya**

Errore di classificazione minimo=22% (K-NN=17).

Si può notare come, rispetto al training set 3, ed alla stessa metrica, in questo caso, pur avendo aumentato il numero di notizie per ogni categorie, l'errore di classificazione minimo è maggiore, mentre in teoria, avendo a che fare con un

numero maggiore di elementi con cui confrontare il documento query corrente, tale errore dovrebbe essere minore. Ciononostante, se si analizza l'andamento di tale errore tra i due training sets, a parità di valore di K , in questo caso esso tende ad essere inferiore, per cui si considera tale caso come un'eccezione.

5.7 Limitazioni della distanza di Bhattacharrya

Dai risultati dei processi di validazione con i sei training sets a disposizione, e dal confronto con la metrica TF-IDF, si può notare che a parità di valori di K , l'errore di classificazione usando la distanza di Bhattacharrya è maggiore, e quindi anche l'errore minimo.

Ciò è dovuto al fatto che tale metrica, nell'assegnare a ciascun termine di ogni documento un peso, utilizza solo informazioni locali, non tenendo conto dell'importanza di tale termine rispetto all'intero training set, e quindi di quanto esso possa essere determinante nel formare il contesto del documento in cui compare.

5.8 Limitazioni della TF-IDF/Vector Space Model

Rispetto alla distanza di Bhattacharrya, in tutti i test tale metrica è risultata migliore, sia rispetto all'errore di classificazione minimo, che rispetto all'errore di classificazione a parità di K .

Ciononostante, è opportuno specificare i seguenti difetti e limitazioni:

1. Ogni volta che si cambia il documento query, è necessario ricalcolare tutti i valori di IDF di tutti i termini del document set, sia perché il nuovo documento query modifica l'insieme di tali termini (aggiungendo nuovi elementi ad esso, e/o eliminandone altri esistenti), che perché comunque la document frequency di ogni termine potrebbe venire modificata.
2. Il modello non prevede un meccanismo che prenda in considerazione non solo la frequenza locale e globale di un termine, ma anche la posizione all'interno del documento in cui esso compare (titolo, 1° paragrafo, corpo, fine, ecc...), che potrebbe anch'essa influire sul peso da assegnare.

5.9 Possibili miglioramenti al programma

1. Riconoscimento sinonimi: Oltre allo stemming ed all'eliminazione delle stopwords, la fase di pre-processing dei documenti dovrebbe prevedere anche l'applicazione di

un dizionario dei sinonimi ai termini, in modo da sostituire termini sinonimi con uno stesso termine (secondo una relazione N:1, per cui più termini, diversi ma sinonimi, vengono sostituiti con un solo termine avente lo stesso significato).

2. Riconoscimento omonimi: Contemporaneamente però, il sistema dovrebbe essere in grado di riconoscere casi di omonimia, nella quale cioè lo stesso termine in documenti diversi ha un significato differente a seconda del contesto in cui si trova (in Italiano, per esempio: la parola “*pesca*” può indicare il frutto, o una voce del verbo pescare; la parola “*legge*” può indicare una disposizione dello stato, oppure una voce del verbo leggere, e così via...). In tali casi infatti, documenti dissimili ottengono un indice di similarità sovrastimato (falsi positivi), dato che il sistema, essendo basato solo su singole parole e non sull'analisi semantica delle frasi, considera termini omonimi uguali.

Per riuscire a fare entrambe le cose, il sistema dovrebbe essere modificato per basarsi non sui singoli termini, ma sui concetti che essi esprimono in un particolare contesto, secondo uno schema (N:M): In fase di pre-processing di ogni documento, ciascun termine dovrebbe essere sostituito con il concetto che esso esprime, basandosi sull'analisi dei termini vicini per determinare il contesto, e consultando un'opposito dizionario delle omonimie nella quale ad ogni termine e ad ogni contesto in cui esso può apparire sia associato il concetto che esso esprime.

■ Riferimenti

1. <http://en.wikipedia.org/wiki/K-NN>
2. <http://www.ansa.it>
3. G. Salton, A. Wong, and C. S. Yang nel 1975 - "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18, nr. 11, pages 613–620
4. Shao, J. (1993), "Linear model selection by cross-validation," J. of the American Statistical Association, 88, 486-494
5. <http://everything2.com/e2node/k-fold%2520cross%2520validation>
6. http://en.wikipedia.org/wiki/Cross-validation#K-fold_cross-validation
7. <http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-12.html>