UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Ingegneria - Dipartimento di Sistemi e Informatica

Tesi di laurea in Ingegneria Informatica Magistrale

Apprendimento automatico per tagging di immagini su base semantica e sociale

Learning to tag images

Candidato
Tiberio Uricchio

Relatori
Prof. Alberto Del Bimbo

Prof. Paolo Frasconi

Correlatori
Ing. Marco Bertini

Ing. Lamberto Ballan

Anno Accademico 2011-2012

Indice

Elenco delle figure

Introduzione

Negli ultimi anni stiamo assistendo ad un aumento esponenziale di contenuti ed informazioni che ogni persona è chiamata a manipolare nella vita quotidiana. Informazioni, foto, musica e video sono presenti nella vita di tutti i giorni essendo diventati molto facili da produrre e alla portata di tutti. Il maggior veicolo dello scambio di informazioni avviene attraverso sistemi informatici, in particolare grazie ai *social network* come ad esempio Facebook[1], Flickr[2] e Twitter[3]. La popolarità di questi strumenti rende possibile l'accumulo di una grande quantità di informazioni e il numero di utenti, sempre più in crescita, non fa altro che aumentarne la visibilità. Pur esistendo così la possibilità di osservare innumerevoli eventi e dati, la mente umana non è in grado di processarne altrettanti e quindi risulta molto importante riuscire a diminuirne il numero. Per i singoli utenti molte informazioni sono inutili se non ripetitive e raggruppabili. Ecco quindi che negli ultimi anni si assiste a un movimento di personalizzazione dei contenuti in relazione ai singoli bisogni e interessi. La ricerca è proseguita di pari passo, allo studio di nuovi sistemi di catalogazione, raggruppamento ed indicizzazione dei contenuti.

Questa tesi si concentra sul problema di categorizzazione o annotazione di immagini. L'obiettivo portante è lo studio di nuovi sistemi per l'analisi e la comprensione dei contenuti visuali, dal livello più basso quali la presenza di colori, bordi o tessiture, fino ad arrivare al livello più alto quale l'analisi semantica, il riconoscimento di oggetti, persone o eventi. In particolare lo sforzo si è concentrato sul problema della categorizzazione, sfruttando

[1]Facebook. `www.facebook.com`

[2]Flickr. `www.flickr.com`

[3]Twitter. `www.twitter.com`

una nuova promettente direzione di ricerca basata sullo sfruttamento delle informazioni presenti nei social network.

I metodi di classificazione delle immagini **allo stato dell'arte** hanno problemi di scalabilità sul numero di concetti considerati in quanto richiedono una procedura di addestramento esteso. I moderni sistemi di *retrieval* utilizzano **descrizioni testuali** per determinare la presenza o meno di un concetto in una data immagine. Purtroppo la maggior parte delle volte non si hanno a disposizione immagini etichettate con descrizioni di alta qualità e l'etichettatura manuale risulta costosa sia in termini di tempo che di risorse. Per cercare di risolvere questo problema si utilizzano tecniche di *machine learning* per cercare di individuare la presenza o meno di concetti semantici sulla base delle *features* visive presenti all'interno delle immagini. L'applicazione pratica della ricerca di concetti semantici in larga scala ha come principale problema la scalabilità delle tecniche di machine learning: per ogni concetto è necessario addestrare un singolo classificatore in grado di individuare la presenza o meno del concetto. A sua volta ogni addestramento necessita di **immagini di esempio etichettate** a dovere. **L'acquisizione di grandi database e di vocabolari di concetti è un processo costoso che necessita di tempo e pone un problema chiave nella scalabilità del sistema.** Per soddisfare la necessità dei dati per ogni addestramento è opportuno valutare nuove fonti di informazioni, ad esempio nei grandi portali di video ed immagini che si sono diffusi a livello mondiale negli ultimi anni. Questi portali costituiscono una risorsa enorme di immagini, informazioni e relazioni tra utenti che potrebbero essere utilizzati per realizzare sistemi completamente automatici, in grado di addestrare classificatori per qualsiasi concetto e valutarne la performance di funzionamento. In questo modo si avrebbe il vantaggio di utilizzare un processo completamente automatizzato e capace di scalare solo sulla base delle risorse computazionali a disposizione. Purtroppo gli utenti che utilizzano questi portali web non sono annotatori professionisti e ognuno possiede un proprio vocabolario di parole. Le annotazioni inseribili sono completamente libere e risultano spesso ambigue, mancanti o troppo personalizzate. A volte vengono utilizzate come annotazioni proprie, per contesti o situazioni non oggettivamente riconducibili

al contenuto dell'immagine. Si assistono anche a fenomeni aggiuntivi quali l'annotazione di periodi temporali, modelli di fotocamere o sigle. Questa mancanza di elementi ha limitato significativamente le applicazioni basate sui tag mentre dall'altro lato rappresenta una nuova sfida per la comunità di ricerca sul multimedia. Si richiede una serie di sforzi di ricerca per processare questi tag incompleti, specialmente nell'utilizzo di tecniche di analisi del contenuto per aumentare il potere descrittivo dei tag rispetto al contenuto dell'immagine. E' quindi importante lo sviluppo di una tecnica automatica di filtraggio, raffinamento e arricchimento per rendere corrette le annotazioni secondo un criterio di valutazione oggettivo.

Il lavoro di tesi ha portato allo sviluppo di un framework completo per il problema del suggerimento e raffinazione di tag, testato su dataset standard. In letteratura, a nostra conoscenza, nessuno ha mai realizzato un framework standardizzato per questo problema. Sono stati inoltre prodotti due contributi scientifici:

- Un confronto di tre tecniche su due dataset rappresentativi delle immagini presenti nei social network.
- Un'analisi qualitativa dell'andamento temporale dei tag nelle immagini con una prima applicazione al problema del Tag Refinement.

La tesi è articolata nel modo seguente:

- il capitolo 1 introduce il problema della classificazione, la relazione con il tagging e formalizza i problemi principali.
- il capitolo 2 descrive le principali tecniche per la descrizione di immagini.
- il capitolo 3 introduce la decomposizione matriciale, utilizzata in alcune tecniche recenti.
- il capitolo 4 offre una descrizione dello stato dell'arte del problema del raffinamento e della valutazione della rilevanza dei tag.

- il capitolo 5 descrive il lavoro compiuto nel corso della tesi, la costruzione del framework analizzando passo dopo passo le decisioni prese ed infine un'analisi dell'utilizzo dei tag nel tempo nei dataset standard.
- il capitolo 6 descrive gli esperimenti compiuti e i risultati quantitativi delle comparazioni.

1

Studio del problema

There's nothing in this universe that can't be explained. Eventually.
– Dr. Greg House, *House MD*

1.1 La classificazione di immagini

La classificazione di immagini è l'operazione di assegnare una classificazione ad un'immagine sulla base del contenuto visuale. Ad esempio se contiene un oggetto oppure no. Esistono diverse applicazioni che rendono questo problema di estrema importanza: un esempio è la ricerca di immagini con un certo contenuto (*retrieval*) oppure il raggruppamento di immagini con contenuti semantici simili (*clustering*). Anche se il problema della classificazione è da molto tempo al centro degli studi nel campo multimediale, solo recentemente è iniziato lo studio dell'applicazione di tecniche su larga scala. La classificazione di immagini è un problema critico sia per gli umani che per i computer, in particolare mentre un umano può riconoscere naturalmente centinaia di migliaia di classi di oggetti e scene, i computer incontrano numerosi problemi. Soltanto recentemente si stanno muovendo i primi passi verso il riconoscimento di un numero di classi elevato, anche se di portata ancora inferiore agli umani. Il primo studio [DBLFF10] sulla classificazione di un numero di categorie a vasta scala risale al 2010 ad opera di Jia Deng. Nello studio vengono utilizzate tecniche basate sull'approccio bag of words

(un modello che conta la presenza o meno di parole "visuali") per realizzare classificatori per 10.000 categorie, utilizzando più di 9 milioni di immagini. Il risultato sono quattro punti importanti:

- i problemi computazionali diventano cruciali nel design degli algoritmi. Basti pensare che nell'utilizzo del classico approccio uno contro tutti è necessario addestrare 10.000 classificatori cioè uno per categoria, senza considerare l'eventuale cross validazione per l'individuazione dei migliori parametri. Infatti, nello studio citato, sono stati utilizzati 66 calcolatori multicore che hanno richiesto in media una settimana di esecuzione a prova.

- le performance misurate negli studi di pochi classificatori su centinaia di immagini non sono indicative quando il numero di categorie aumenta su larga scala.

- esiste una forte relazione tra la struttura tassonomica di WordNet [Fel98] (sviluppata per studiare il linguaggio) e la difficoltà di categorizzazione delle immagini. In base alla specificità dei concetti la difficoltà aumenta. Lo stesso vale per la densità dei concetti.

- la classificazione può essere migliorata utilizzando la gerarchia semantica.

La ricerca è proseguita sviluppando recentemente tecniche basate sempre su bag of words, ma sfruttando la tecnica di *embedding* [BLRF11] per il calcolo efficiente della funzione di matching, in modo da diminuire la complessità computazione insita nell'utilizzo dei metodi kernel. La proiezione di features non lineari, in features lineari ha il vantaggio di permettere l'utilizzo delle recenti ottimizzazioni delle tecniche di apprendimento lineari, quali ad esempio Pegasos [SSSS07] o Liblinear [FCH$^+$08], migliorando notevolmente i tempi di elaborazione a scapito dell'occupazione della memoria e del caricamento dati. Anche se queste tecniche diminuiscono i tempi di apprendimento per singole categorie, rimane lo stesso il problema essenziale del reperimento della *ground truth*: immagini etichettate in modo adeguato utilizzando criteri oggettivi,

condivisi e ripetibili. Ad esempio, solo per l'etichettatura di 500 concetti in LSCOM è stato necessario spendere circa 6000 ore di lavoro [YfCKH07]. Inoltre, in questo modo, il processo di apprendimento di nuovi concetti non è completamente automatizzato.

Una parte della ricerca è passata a sviluppare tecniche per superare il problema dell'annotazione: i principali social network presenti su Internet possiedono enormi quantità di immagini con associati una serie di "tag", ovvero parole chiavi comparabili ad etichette e che potrebbero costituire la soluzione al problema. Se fosse possibile imbrigliare questi dati e renderli utilizzabili, potrebbe essere possibile realizzare classificatori in modo completamente automatico, riducendo il problema soltanto alla disponibilità di risorse computazionali.

1.2 Il tagging e la classificazione

In tempi recenti, grazie all'avvento del Web 2.0, si è assistito all'esplosione della condivisione dei contenuti attraverso grandi portali quali ad esempio Flickr, Youtube o Facebook. Oltre alla semplice fruizione dei contenuti resi disponibili da professionisti, questi portali rendono possibile la condivisione di contenuti personali in maniera semplice ed efficace, portando una trasformazione delle attività del Web. Una delle attività emergenti del Web 2.0 è il cosiddetto *tagging* ovvero l'azione di annotare manualmente i contenuti utilizzando parole chiavi scelte liberamente. Per gli utenti, il tagging costituisce sempre di più un modo per organizzare, indicizzare e ricercare contenuti multimediali; inoltre ne rende possibile il recupero e il raggruppamento. Nonostante l'alta popolarità del tagging, le etichette prodotte dagli utenti di Internet sono lontane da essere soddisfacenti per l'utilizzo di indicizzazione dei contenuti delle immagini.

Si potrebbe pensare di associare il tagging alla classificazione fatta da esperti che viene generalmente compiuta nei lavori di ricerca, tuttavia non è questo il caso. In [SS09], un lavoro di Setz et al. del 2009, viene mostrato che i tag sono spesso ambigui ed altamente personalizzati, a volte contengono errori e generalmente sono mancanti. Una fase di filtraggio e di disambigua-

zione rende comunque possibile un primo miglioramento delle prestazioni di classificazione.

Successivamente Li et al. in [LS09] introducono l'idea di utilizzare le immagini e i tag associati presenti su Flickr per costituire una riserva 'infinita' di esempi negativi da utilizzare al posto dell'annotazione manuale. Sono descritti due esperimenti: il primo con immagini etichettate in modo professionale, sia per quanto riguarda gli esempi positivi che gli esempi negativi; il secondo con immagini etichettate in modo professionale per quanto riguarda gli esempi positivi e in modo automatico per quanto riguarda gli esempi negativi. Reperito un insieme di immagini da Flickr e dato un concetto da apprendere, vengono preparati gli esempi negativi rimuovendo le immagini etichettate con il concetto o con un suo sinonimo. I risultati evidenziano un calo di performance di circa il 4% utilizzando le informazioni sociali.

In [SvZ08] è stato eseguito uno studio volto a rispondere a domande tipo 'come gli utenti etichettano le foto?', 'Che tipo di tag inseriscono?'. L'analisi è stata eseguita su 52 milioni di immagini contenenti 188 milioni di tag in totale (con un dizionario di 3,7 milioni di tag) ed ha evidenziato le seguenti considerazioni:

- La distribuzione della frequenza dei tag è modellata accuratamente utilizzando una legge esponenziale inversa. La probabilità di che un tag abbia frequenza x è proporzionale a $x^{-1.15}$.
- La testa della distribuzione contiene tag generici come ad esempio *2006*, *2005*, *wedding*, *party*, e *2004*.
- La coda finale della distribuzione contiene tag poco frequenti, tipicamente categorizzabili come parole occasionali come errori di battitura o frasi complesse. Ad esempio *ambrose tompkins*, *ambient vector*.
- 1,57 milioni di tag compaiono solo una volta.
- Il 29,8 % delle immagini ha 1 tag, il 32,7 % ha 2-3 tag, il 23,1 % ha 4-6 tag ed infine il restante 14,4 % ha più di 6 tag.

- Non esiste un ordine che ne indichi l'attinenza o la correlazione con il contenuto visivo. Ciò rende equivalenti tutti gli elementi della lista delle etichette e non permette di creare liste di ranking da utilizzare nelle ricerche.

- I tag non vengono inseriti secondo un criterio oggettivo bensì sono influenzati dall'utente che li inserisce. Ogni utente inserisce tag utilizzando un proprio criterio di importanza, preferendo termini legati alla propria esperienza o conoscenza. Questo implica che ogni utente utilizzi un vocabolario proprio, potenzialmente differente da quello di tutti gli altri utenti. Un esempio è l'utilizzo di slang relativi al campo specifico di appartenenza dell'utente: termini militari, geografici, di mestieri particolari o altro.

- I tag sono potenzialmente mancanti in quanto l'attività di tagging è ritenuta onerosa e molto spesso è limitata alle etichette ritenute più importanti all'utente e che possono essere differenti da quelle ritenute importanti da un altro utente. Anche i sinonimi ed i termini espressi in altre lingue sono generalmente mancanti e che invece potrebbero essere utili ai fini della ricerca.

- I tag vengono assegnati a livello di immagini ciò rende difficile il collegamento di aree dell'immagine e termini specificati.

- Le etichette, a volte, descrivono un contesto o una situazione irrilevante ai fini del contenuto visivo. Ad esempio etichette quali 'compleanno', 'in gita', etc.

Un altro problema è il cosiddetto "semantic gap" [vGVSG10], ovvero la presenza di un'elevata distanza concettuale tra i dati di basso livello delle immagini e le informazioni di alto livello delle annotazioni. Un algoritmo può cercare di "capire" un'immagine soltanto utilizzando i pixel, i valori dei singoli punti e la loro posizione. Invece le annotazioni testuali dipendono sempre da una qualche conoscenza, capacità ed espressione di linguaggio dell'annotatore che quindi le rende inaffidabili. Per cercare di riconoscere le

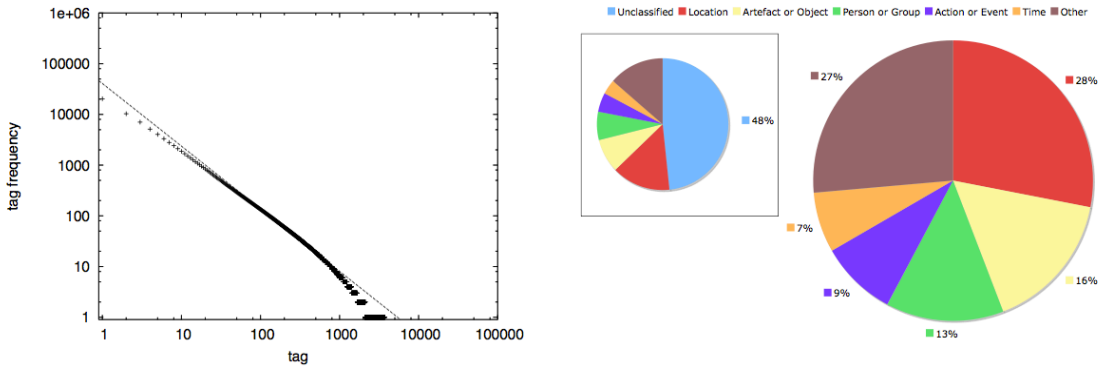

Figura 1.1. Frequenza e distribuzione dei tag risultante nello studio in [SvZ08].

scene, gli algoritmi devono selezionare e manipolare i pixel e soltanto alla fine cercano il collegamento con la descrizione naturale. Anche la più semplice rappresentazione di forme o colore richiede una formalizzazione attraverso metodi matematici, che a loro volta non sono unici.

Sulla base di queste considerazioni dobbiamo pensare ad una fase di pre-processing, necessaria a rendere maggiormente affidabili le informazioni disponibili nei tag. Questo apre una serie di nuovi problemi che devono essere risolti dalla comunità del multimedia e della computer vision:

- **Tag ranking:** per tag ranking, s'intende l'attribuzione di un ordinamento o di un punteggio a ogni etichetta assegnata alle immagini, in modo da poter valorizzare la correlazione tra le etichette e il contenuto visivo.

- **Tag suggestion:** data un'immagine con un insieme di etichette già assegnate, per tag suggestion s'intende l'individuazione di nuove etichette adatte al contenuto in questione. Si assume che le etichette già presenti siano corrette.

- **Tag refinement:** il tag refinement è il problema di migliorare e rifinire i tag inaffidabili inseriti dagli utenti, rimuovendo le etichette non correlate con il contenuto visivo e aggiungendo le etichette mancanti. Si vuole individuare una serie di tag che meglio descrivono il contenuto,

possibilmente mediante l'utilizzo di un vocabolario comune. Generalmente il problema del tag refinement include come sottoproblema il tag ranking. Possiamo definire anche una variante personalizzata con la differenza che le etichette vengono assegnate non secondo un criterio oggettivo generale ma bensì secondo il criterio personale di un utente specifico. È la proiezione del contenuto secondo il punto di vista personalizzato di un utente. Questo lavoro di tesi rientra nella presente categoria.

- **Tag to region e tag to time interval:** Si tratta dell'individuazione delle regioni del contenuto visivo correlate alle etichette specificate.

Definiamo ora formalmente i problema enunciati.

1.3 Formalizzazione dei problemi

Dato un'insieme di immagini $I = \{i_1, i_2, \ldots, i_N\}$, un insieme di utenti $U = \{u_1, u_2, \ldots, u_M\}$ ed infine un vocabolario di tag (parole) $V_T = \{t_1, t_2, \ldots, t_K\}$ definiamo la relazione ternaria:

$$\text{tag}(u, i, t) \subseteq U \times I \times V_T \tag{1.1}$$

ovvero l'utente u, tagga l'immagine i con il tag t. Qualsiasi utente può taggare qualsiasi immagine, non ci sono vincoli per quanto riguarda l'insieme di tag o di immagini. Lo stesso una qualsiasi immagine può essere taggata da qualsiasi utente con qualsiasi tag del vocabolario. Come descritto nella sezione precedente, i tag inseriti dagli utenti non possiedono un ordine. È possibile soltanto definire la presenza o l'assenza. Sulla base della relazione ternaria è possibile definire anche la relazione binaria classica:

$$\text{tag}(i, t) \subseteq I \times V_T := \text{tag(u, i, t)}\, \forall u \in U \tag{1.2}$$

ovvero la relazione di possesso di tag per ogni immagine. Ogni immagine è inserita nel sistema da parte di un utente che ne possiede la proprietà, ovvero definiamo anche la relazione di possesso:

$$\text{owner}(u, i) \subseteq U \times I \tag{1.3}$$

Queste informazioni sono tutte reperibili facilmente sui social network principali.

Denotiamo infine con Φ la terna (U, I, V_T), ovvero un corpus di immagini con i relativi utenti e un vocabolario di tag.

Definizione 1.1. Dato un corpus Φ, un'immagine $i \in I$ e un tag $t \in V_T$, il problema del **tag ranking** è quello di individuare una funzione

$$r(i, t) : (I, V_T) \to \mathbb{R} \tag{1.4}$$

che associa un valore quantitativo alla rilevanza del tag t rispetto al contenuto visivo in i.

L'applicazione di una funzione di ranking ad una immagine singola ed un insieme di tag dovrebbe restituire un punteggio per ogni tag e quindi indurre una lista di ranking, ossia un ordinamento di preferenza per i tag considerati. Il valore numerico assume importanza nel caso in cui si voglia individuare una soglia per tagliare i tag completamente incompatibili, tuttavia l'informazione principale è costituita dall'ordine decrescente del punteggio. Il concetto di rilevanza è prettamente ambiguo in quanto ogni persona potrebbe affermare per ogni tag un valore diverso da quello di un'altra. Un modo per cercare di ridurre questo problema è quello di istituire un'ontologia di concetti "oggettivi" che la maggior parte delle persone è d'accordo sul significato semantico. Nonostante questo esiste ancora una forma di ambiguità visiva. Si pensi, ad esempio, ad un lago fotografato dalla riva verso l'orizzonte: se il lago è abbastanza grande potrebbe essere considerato o scambiato per il mare. E quindi, in tal caso, sarebbe più rilevante il tag 'lago' (un'informazione proveniente dalla conoscenza), oppure il tag 'mare'? La risposta dipende dalla conoscenza e l'esperienza della persona che guarda l'immagine.

Una buona funzione di ranking dovrebbe rispondere correttamente ai seguenti due criteri. Dato un corpus Φ, due immagini $i_1, i_2 \in I$ ed un tag $t \in V_T$ se t è rilevante per i_1, ma non è rilevante per i_2 allora

$$r(i_1, t) > r(i_2, t) \tag{1.5}$$

Dato un corpus Φ, un'immagine $i \in I$ e due tag $t_1, t_2 \in V_T$ se t_1 è rilevante per i e t_2 non è rilevante per i allora

$$r(i, t_1) > r(i, t_2) \tag{1.6}$$

Si può anche definire anche una funzione di ranking personalizzata per utente. In questo caso la definizione della funzione di ranking acquista un parametro aggiuntivo per l'utente, ma rimane simile di significato alla precedente.

Definizione 1.2. Dato un corpus Φ, un'immagine $i \in I$, un tag $t \in V_T$ ed un utente in $u \in U$, il problema del **tag ranking personalizzato** è quello di individuare una funzione

$$r(u, i, t) : (U, I, V_T) \to \mathbb{R} \tag{1.7}$$

che associa, per l'utente u, un valore quantitativo alla rilevanza del tag t rispetto al contenuto visivo in i.

Successivamente al problema del ranking, esiste il problema del suggerimento di nuovi tag (**tag suggestion**).

Definizione 1.3. Dato un corpus Φ, un'immagine $i \in I$ e un'insieme di tag $T_i = \{t_1, t_2, \ldots, t_K\} \in V_T$ rilevanti per i e per i quali vale la relazione $tag(i, t)\ \forall t \in T_i$, il problema della **tag suggestion** è quello di individuare una funzione

$$\text{suggestion}_M(i, T_i) : (I, \mathcal{P}(V_T)) \to \mathcal{P}(V_L) = \{l_1, l_2, \ldots, l_M\} \tag{1.8}$$

che associa un insieme di M tag ordinati e rilevanti al contenuto visivo in i. L'insieme di tag prodotti non può contenere i tag in ingresso. La notazione $\mathcal{P}$ indica l'operatore *power set* e V_L è un insieme di tag che può contenere termini aggiuntivi rispetto al vocabolario utilizzato dagli utenti.

Abbinando la funzione di tag suggestion ad una funzione di ranking è possibile valutare la rilevanza dei tag suggeriti. È importante notare che si assumono tutti i tag originali come rilevanti. Esiste anche la variante in cui

questo non avviene, in tal caso non è comunque contemplata la rimozione dei tag non rilevanti dall'insieme originale. I tag prodotti sono sempre da considerare aggiuntivi.

Oltre ai tag definiamo anche un insieme di etichette $V_L = \{l_1, l_2, \ldots, l_L\}$ denominandole *'label'*, utilizzate per l'output della funzione. Un buon metodo per scegliere un vocabolario delle label generico è quello di utilizzare il vocabolario della lingua in questione. La distinzione tra i due vocabolari è utile per una serie di motivi:

- i tag utilizzati dagli utenti possono essere etichette personali o slang. Definendo due vocabolari possiamo sia indicare un insieme di termini 'oggettivi' che hanno un significato ben preciso per la maggior parte delle persone.
- l'applicazione sotto studio potrebbe aver bisogno di predizioni su un'ontologia specifica.
- ai fini della valutazione delle performance, è troppo costoso e laborioso ottenere i dati di ground truth per tutti i tag della collezione. Si potrebbe selezionare solo un sotto insieme e migliorare le performance avendo a disposizione più dati di input.

Analogamente alla relazione *tag* definiamo anche *label* che svolge la stessa funzione di associare una label ad un'immagine:

$$\text{label}(i, t) \in I \times V_L \tag{1.9}$$

La definizione di label è più vicina al concetto di oggettività enunciato prima, e come tale non include gli utenti.

Diretta conseguenza del problema di **tag suggestion** è il problema di raffinamento dei tag (**tag refinement**), dove cambia l'insieme di tag predicibili:

Definizione 1.4. Dato un corpus Φ, un'immagine $i \in I$ e un insieme di tag $T_i = \{t_1, t_2, \ldots, t_K\} \in V_T$ per i quali vale la relazione $tag(i, t)$ $\forall t \in T_i$, il problema del **tag refinement** è quello di individuare una mappa

$$\text{refine}_M(i, T_i) : (I, \mathcal{P}(V_T)) \to \mathcal{P}(V_L) = \{l_1, l_2, \ldots, l_M\} \tag{1.10}$$

che associa un insieme di M tag ordinati e rilevanti al contenuto visivo in i. $V_L \subseteq V_T$ è un insieme di tag che può contenere termini aggiuntivi rispetto al vocabolario utilizzato dagli utenti.

L'insieme prodotto dalla funzione di raffinamento è inteso come sostituto dell'insieme di tag iniziale. Deve quindi contenere i tag più rilevanti per il contenuto visuale di i, che potrebbero essere contenuti nell'insieme in ingresso. Anche qui vale lo stesso discorso fatto in precedenza per i due dizionari. Si parla anche di raffinamento personalizzato.

Definizione 1.5. Dato un corpus Φ, un'immagine $i \in I$, un utente $uinU$ e un'insieme di tag $T_i = \{t_1, t_2, \ldots, t_K\} \in V_T$ per i quali vale la relazione $tag(i,t)$ $\forall t \in T_i$, il problema del **tag refinement personalizzato** è quello di individuare una mappa

$$\text{refine}_M(u, i, T_i) : (I, \mathcal{P}(V_T)) \to \mathcal{P}(V_L) = \{l_1, l_2, \ldots, l_M\} \tag{1.11}$$

che associa un insieme di M tag ordinati e rilevanti al contenuto visivo di i, secondo u. $V_L \subseteq V_T$ è un insieme di tag che può contenere termini aggiuntivi rispetto al vocabolario utilizzato dagli utenti.

Infine l'individuazione di regioni di immagine relative ad un tag della stessa è il problema della **tag to region**:

Definizione 1.6. Dato un corpus Φ, un'immagine $i \in I$, la sua matrice dati $M_i \in \mathbb{R}^{M \times N}$ e un tag $t \in V_T$ per il quale vale la relazione $tag(i,t)$, il problema del **tag to region** è quello di individuare una mappa

$$\text{region}(M_i, T_i) : (\mathbb{R}^{M \times N}, \mathcal{P}(V_T)) \to \mathbb{R}^{M \times N} \tag{1.12}$$

che associa per il tag t un valore di rilevanza al contenuto visivo in i ad ogni pixel dell'immagine.

Questo problema è fortemente connesso al problema della segmentazione di un'immagine in quanto una buona segmentazione semantica dipende dalle conoscenze della persona (o macchina) che la esegue.

2

Tecniche di analisi e descrizione delle immagini

A picture is a poem without words.

– Cornificius, *Anet. ad Her., 4. 28.*

La ricerca di concetti all'interno di un'immagine passa necessariamente attraverso la rappresentazione grafica su un calcolatore. Come descritto in precedenza, un computer può soltanto analizzare le informazioni di basso livello di un'immagine. I dati grezzi disponibili sono generalmente una serie di matrici di pixel, che descrivono i valori di luminosità di ogni punto in un qualche spazio di colore (o scala di grigi). La ricerca si è da sempre prodigata nella formulazione di tecniche che cerchino di descrivere in modo più alto i contenuti dell'immagine. Anche se esiste ancora il cosiddetto "semantic gap" tra i descrittori più moderni e le informazioni di altissimo livello come i tag, gli avanzamenti più recenti riescono a fornire strumenti e metodi per individuare immagini simili rispetto ad alcune caratteristiche. Possiamo distinguere le tecniche in due gruppi:

- Tecniche che estraggono *features* di tipo **globale**, basate generalmente sullo studio di istogrammi di immagini.
- Tecniche che estraggono *features* di tipo **locale**, basate generalmente sullo studio di punti salienti (o *keypoints*).

Le features di tipo globale sono state in passato oggetto di ricerca fin dall'antico problema del retrieval e del matching. Si tratta generalmente di tecniche che studiano colori e tessiture, due delle più importanti qualità di un'immagine. Viceversa, le features di tipo locale, fanno la loro apparizione in tempi più recenti e si basano sull'individuazione e descrizione di punti salienti. Questi elementi di interesse si riferiscono a caratteristiche di tipo locale ed offrono una certa robustezza ad alcune trasformazioni quali l'ingrandimento o scala, la rotazione, il cambiamento del punto di osservazione, e così via.

2.1 Features di tipo globale

Le features di tipo globale rappresentano l'immagine nella sua interezza, sono generalmente più semplici di quelle locali e più facilmente computabili. Sono categorizzabili in features basate sul colore e features basate sulle *textures*.

2.1.1 Features basate sul colore

Le features basate sul colore sono utilizzate spesso nell'indicizzazione e nel retrieval di immagini, proprio per la loro semplicità e velocità di computazione. Utilizzano poche dimensioni per descrivere l'immagine. La prima cosa da definire è lo spazio di colore su cui sono definite: i più usati sono **RGB** (Red Green Blue), **HSV** (Hue Saturation Value) e **LAB** (Luminance a,b opponents). Mentre RGB è lo spazio di colore predefinito per la cattura e la visualizzazione delle immagini, sia HSV che LAB isolano importanti caratteristiche che non sono evidenti in RGB. Ad esempio l'HSV memorizza la quantità di luce nel canale *Value* mentre LAB cerca di descrivere nel canale *Luminance* l'intensità di luminosità percepita dagli umani.

La feature più semplice è prendere dei dati grezzi dell'immagine. Si possono campionare dei pixel oppure eseguire medie o valutazioni ponderate. L'importante è mantenere l'ordinamento per far corrispondere i dati. Si tratta di un subset dell'immagine e quindi poco rappresentativa dell'intero contenuto.

Sicuramente migliore è l'utilizzo dell'istogramma di un'immagine: si contano semplicemente i valori di ogni singolo pixel e si costruisce l'istogramma definendo il numero di bin voluti. Un modo per ottenere le features è considerare un numero ben preciso di bin richiesti per canale: maggiore è il numero di bin, più precisa è la rappresentazione dell'istogramma a scapito dell'occupazione di memoria. I valori di ogni bin così ottenuti per ogni canale vengono concatenati per ottenere infine le features finali.

È possibile inserire una semplice forma di informazione spaziale calcolando l'istogramma su parti dell'immagine: invece di utilizzare l'intera immagine per il calcolo dell'istogramma, si dividere l'immagine in sotto parti e si calcola un istogramma per ogni parte. Ad esempio è possibile dividere l'immagine in tre parti in verticale ed in altre tre parti in orizzontale. Per semplicità di gestione è poi possibile concatenare tutte le features così ottenute oppure tenerle sempre separate. La scelta potrebbe anche cambiare in parte le performance in base alla presenza e al tipo della normalizzazione.

2.1.2 Features basate sulle texture

Le features basate sulle texture fanno generalmente utilizzo di metodi di analisi di segnali come matrici di co-occorrenza o filtraggi a frequenza per individuare i pattern dell'immagine (ricordiamo che le texture sono definite solo su regioni di immagine). Le features più utilizzate sono quelle di Tamura [TMY78] (basate su esperimenti psicologici), Gabor (basate su una trasformazione analoga a quella di Fourier) e infine GIST [OT06] di Oliva e Torralba che è diventato popolare negli ultimi anni. Vediamo GIST che, al momento, è considerato tra questi il descrittore più importante.

GIST

L'idea alla base del descrittore GIST è inversa rispetto all'approccio che viene generalmente adottato nella classificazione delle scene. Invece di inferire il contenuto delle immagini attraverso gli oggetti individuati, l'approccio tenta di utilizzare il layout spaziale, partendo così dalla stima della struttura globale e dalle relazioni spaziali tra le componenti. Successivamente vengono

analizzati i dettagli locali dell'immagine. Da un punto di vista percettivo è stato dimostrato che l'osservazione di immagini riprodotte a bassa frequenza è sufficiente per percepire l'essenza (che conia il termine *gist*) del contenuto. Inoltre è stato dimostrato che generalmente le persone, per un task di classificazione, utilizzano in un primo tempo di osservazione le componenti a bassa frequenza e solo successivamente impiegano componenti ad alta frequenza. Questo tipo di studi è stato eseguito esponendo (per tempi molto brevi) soggetti a immagini artificiali che fondono frequenze basse ed alte di due immagini diverse.

Mettendo insieme le attività di detector di feature a basso livello su larghe regioni del campo visivo è possibile ricostruire la struttura di una scena senza ricorrere alla segmentazione oppure alla ricerca di oggetti e feature locali, con evidente vantaggio in termini computazionali. Una naturale estensione di questo principio consiste nell'utilizzo non di un'unica feature globale, ma di una collezione di feature globali in grado di catturare la variabilità dei layout e dei punti di osservazione che una scena presenta nel mondo reale.

Il descrittore GIST utilizza un insieme di features globali scelte per l'analisi delle scene, le quali vengono calcolate sulla base dell'output ottenuto attraverso la convoluzione delle immagini con un insieme di filtri orientati a più scale. Scelto un numero N di campioni per il filtro in ogni dimensione, un numero K di orientazioni e scale, la dimensione del descrittore per un'immagine in scala di grigi diventa $N \times N \times K$, dove $N \times N$ è l'insieme dei valori in cui viene salvata la risposta del filtro per ciascun valore di scala ed orientazione. Un'analisi di questo tipo descrive le proprietà di quello che è definito come *Spatial Envelope* della scena, che descrive la naturalezza (*naturalness*), l'apertura (*openness*) e la ruvidezza (*roughness*). Il descrittore GIST, essendo globale, raccoglie le informazioni di un'intera immagine.

Utilizzando i valori suggeriti nell'articolo originale, 20 valori per le orientazioni, 16 elementi su 3 canali per ciascun filtro, il descrittore finale risulta di dimensioni $D = 20 \times 16 \times 3 = 960$.

2.2 Features di tipo locale

Le features di tipo locale rivestono una considerevole importanza principalmente nel problema del matching di immagini e nella categorizzazione. Sono per lo più legate alle informazioni locali di un'immagine, riuscendo ad avere un notevole vantaggio sul problema delle occlusioni. L'utilizzo di features locali è generalmente risultato positivo in termini di performance rispetto alle features globali. Tuttavia il loro costo in termini computazionali è maggiore, sia di tempo di elaborazione che di occupazione di spazio. Per le features di tipo locale dobbiamo distinguere due attività:

- La ricerca di punti interessanti.
- La descrizione delle informazioni rilevate.

2.2.1 Ricerca di punti salienti

Le strategie di ricerca dei punti salienti possono essere divise in due categorie: tecniche di campionamento **sparso** o **denso**. La prima categoria si avvale di metodi di detection basati prettamente sul contenuto dell'immagine, mentre la seconda seleziona punti con canoni stabiliti basate soltanto sulle dimensioni fisiche dell'immagine. Generalmente vogliamo estrarre la "giusta" quantità di informazione, che si traduce in un numero di punti locali sufficiente a descrivere la specificità delle informazioni presenti. Troppi pochi punti non descrivono efficacemente il contenuto informativo, viceversa troppi punti tendono a sprecare risorse.

Il vantaggio principale che deriva dall'uso di una tecnica di ricerca di tipo sparso è che permette di individuare le regioni delle immagini a maggior contenuto visivo o informativo. È necessario tuttavia porre attenzione al tipo o alla risoluzione dell'immagine per individuare la giusta quantità di punti. Una tecnica densa tende invece a coprire l'intera immagine in quanto non ricerca alcun pattern particolare. Questo si traduce in una visione più globale dell'immagine, ma non permette di descrivere accuratamente zone che avrebbero bisogno di più punti.

Pensiamo ad esempio ad un'immagine completamente bianca con soltanto un piccolo disegno al centro. Una tecnica di tipo sparso potrebbe riuscire ad individuare gli elementi del piccolo logo ed essere in grado di descriverlo; invece una tecnica di tipo denso potrebbe selezionare punti qua e là, senza catturare efficacemente il logo centrale, ma valorizzando la grande assenza di elementi.

Campionamento sparso

Negli ultimi anni sono state sviluppate decine di algoritmi di ricerca per feature locali salienti in immagini: una buona analisi è disponibile in [MTS+05]. Due delle tecniche più usate che offrono le migliori performance per la detection sono sicuramente la **Harris-affine** [HS88] e la **Hessian-affine** [Low04] (utilizzata nel SIFT introdotto da Lowe). Entrambe le tecniche individuano punti di interesse nell'analisi *scale-space* e poi determinano una regione ellittica per ogni punto. I punti di interesse sono individuati con un Harris detector oppure un detector basato sulla matrice Hessiana. In entrambi i casi la selezione della scala è basata sull'operatore laplaciano e la forma della regione ellittica è determinata in base al momento del secondo ordine del gradiente dell'intensità. La matrice del momento del secondo ordine, denominata anche matrice di autocorrelazione, è utilizzata spesso per la detection di feature o per la descrizione di strutture locali presenti nell'immagine. La matrice è così definita:

$$M = \mu(\mathbf{x}, \sigma_1, \sigma_D) = \begin{bmatrix} \mu_{1,1} & \mu_{1,2} \\ \mu_{2,1} & \mu_{2,2} \end{bmatrix} = \\ = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{y}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{y}, \sigma_D) \end{bmatrix}$$

Le derivate locali sono calcolate utilizzando una serie di kernel gaussiani a diversa scala σ_D (detta anche rappresentazione piramidale). Gli autovalori di questa matrice rappresentano le due principali variazioni del segnale in un vicinato del punto. Questa proprietà ci permette l'estrazione dei punti per i quali è importante sia la curvatura che il cambiamento del segnale, secondo

direzioni ortogonali. I punti così trovati sono stabili rispetto a cambiamenti arbitrati di luminosità e sono rappresentativi per un'immagine. Un'idea simile è utilizzata nei detector che utilizzando la matrice Hessiana:

$$H = H(\mathbf{x}, \sigma_D) = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma_D) & I_{xy}(\mathbf{x}, \sigma_D) \\ I_{yx}(\mathbf{x}, \sigma_D) & I_{yy}(\mathbf{x}, \sigma_D) \end{bmatrix} \tag{2.1}$$

La derivata seconda, utilizzata in questa matrice, genera risposte forti sui *blobs* (macchie) e sui contorni. Le regioni sono molto simili a quelle rilevate da un'operatore laplaciano tuttavia l'utilizzo di una funzione basata sul determinate della matrice Hessiana rischia di penalizzare le strutture molto lunghe, per le quali una derivata seconda verso una particolare orientazione è molto piccola. Il massimo locale del determinate indica la presenza di una struttura a blob. L'idea è di individuare la scala per la quale si ha un massimo per l'operatore laplaciano, una variante della tecnica di Lindeberg [Lin98].

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.2}$$

dove $I(x, y)$ è l'intensità dell'immagine in esame al punto (x, y),

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{\sigma^2}} \tag{2.3}$$

è la funzione gaussiana bivariata con scala pari a σ e $L(x, y, \sigma)$ è l'immagine risultante. Per questioni di performance, l'operatore laplaciano di gaussiane, viene generalmente approssimato utilizzando le differenze di gaussiane nello scale-space:

$$D(x, y, \sigma) = \frac{L(x, y, k\sigma) - L(x, y, \sigma)}{k - 1} \tag{2.4}$$

Per ogni livello, ciascun pixel viene confrontato con i suoi 8 vicini alla stessa scala e con i 9 vicini individuati sulle scale adiacenti (inferiore e superiore). Soltanto i pixel che risultano essere estremi locali (massimi o minimi) vengono scelti come *keypoints*. Il valore della scala σ dove viene individuato il keypoint è indicato come scala caratteristica del punto.

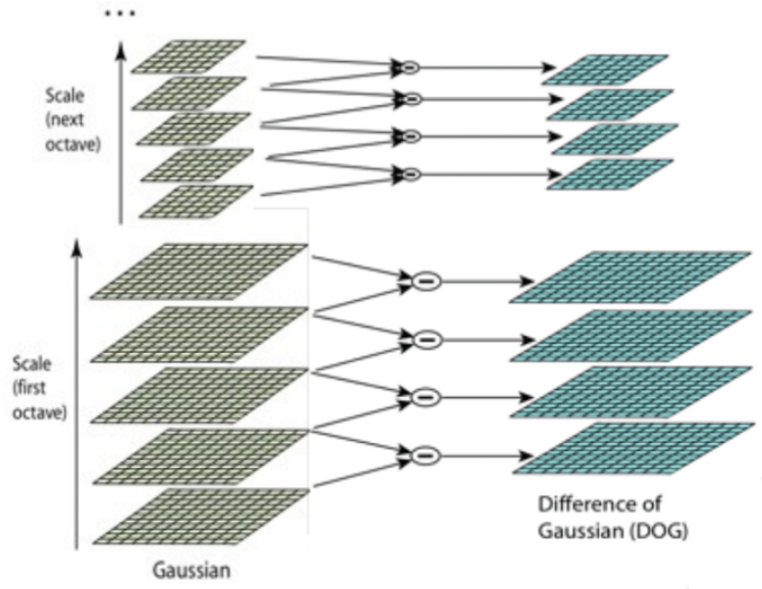

Figura 2.1. L'operatore differenza di gaussiane nella rappresentazione piramidale di un'immagine.

Campionamento denso

Il campionamento denso considera generalmente una serie di punti fissi dipendenti da parametri conosciuti a priori, non influenzati dal contenuto visuale dell'immagine. Esistono principalmente due tecniche:

1. **campionamento a griglia**: si fissa a priori un parametro K che indica il passo di campionamento, ossia la distanza di campionamento da un punto ed un altro in orizzontale ed in verticale. Il risultato è il campionamento di una griglia più o meno densa in base al parametro. L'idea alla base è la ricerca di una sorta di rappresentazione globale, indipendentemente dal contenuto dell'immagine.

2. **campionamento random**: si fissa a priori un numero di punti richiesti N e si esegue una generazione random di N coordinate dell'immagine con estrazioni senza ripetizioni.

Recentemente alcuni studi [Ff05] hanno mostrato che le tecniche di campionamento denso a griglia godono di prestazioni superiori in applicazioni di

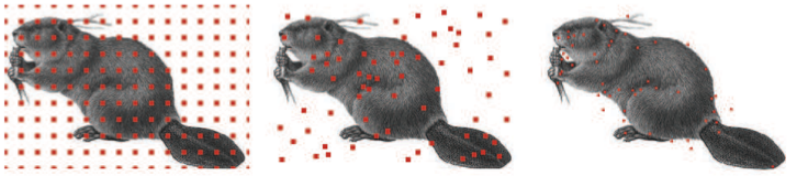

Figura 2.2. I tre tipi di campionamento illustrati. Da sinistra: campionamento denso a griglia, campionamento denso random, campionamento sparso con operatore Hessian

classificazione basate su bag of words di scene naturali.

2.2.2 Descrizione di punti salienti

Individuati i punti salienti è conveniente descrivere le informazioni utili della zona in modo conciso, preferibilmente in descrittori con un numero di dimensioni ridotte ma pur sempre adatte a contenerle. In letteratura ne esistono di svariati tipi: SIFT (Scale Invariant Feature Transform) [Low04], SURF [BTVG06], Color SIFT [vdSGS10], Robust Hue Descriptor o Hue-SIFT [vdWS06], HMAX [RPS99], ASIFT [MY09] e molti altri. Tra di essi, i più popolari sono sicuramente SIFT e i suoi derivati con l'aggiunta delle informazioni sul colore ColorSIFT.

Descrittore SIFT

Fissata una scala caratteristica per un keypoint, il calcolo del descrittore SIFT procede secondo due fasi:

1. calcolo dell'orientazione canonica del keypoint
2. calcolo del vettore di feature

Prima di tutto si determina l'immagine $L(x, y, \sigma)$ alla scala assegnata al keypoint. Presa una finestra 4x4 pixel intorno al keypoint in questa immagine, si accumulano in un istogramma a 8 dimensioni le direzioni del gradiente. Il picco registrato nell'istogramma corrisponde alla **direzione dominante** del gradiente locale per il punto corrente. Il descrittore viene calcolato considerando una finestra di 16x16 locazioni nell'intorno del keypoint alla scala

ad esso assegnata, prendendo come sistema di riferimento quello individuato dall'orientazione canonica. In questo modo si rende il descrittore invariante a rotazioni. Tali locazioni vengono raggruppate in sottoregioni 4x4, per ciascuna delle quali è calcolato un istogramma a 8 dimensioni delle orientazioni del gradiente locale, sempre in riferimento alla direzione dominante. I valori di ogni bin dell'istogramma equivalgono alla somma delle ampiezze dei vettori gradiente con direzione nel range del bin considerato. Il risultato finale è un vettore a 128 dimensioni: 16 (4x4) x 8. Infine il descrittore del keypoint è normalizzato per renderlo invariante a cambi di intensità.

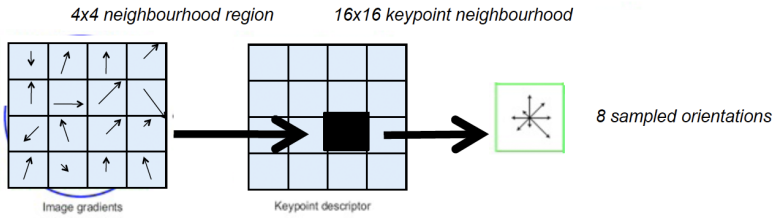

Figura 2.3. Procedimento per il calcolo di un descrittore SIFT.

Descrittore ColorSIFT

Il descrittore SIFT si basa soltanto sulla variazione di luminosità, non considerando il colore. Un'estensione interessante, il descrittore ColorSIFT, analizza anche le informazioni di colore. In [vdSGS10], Van de Sande *et al.* presentano uno studio delle proprietà di invarianza e del grado di distintività di descrittori basati su SIFT ed estesi alla rappresentazione di informazioni di colore. Le diverse soluzioni si differenziano in base allo spazio di colore utilizzato per la rappresentazione dell'immagine ed al tipo di approccio utilizzato per la descrizione SIFT. L'autore ha reso disponibile un software per la descrizione di immagini che consente di utilizzare sia la versione a scale di grigio "classica" di Lowe che le possibili varianti che impiegano l'immagine a colori.

HSV-SIFT: il descrittore SIFT viene calcolato sui tre canali dello spazio di colore HSV (Hue Saturation Value), che corrispondono singolarmente

ad un'immagine a scala di grigi dell'implementazione originaria. Il risultato è un descrittore di dimensione complessiva $3 \times 128 = 384$.

HueSIFT: introdotto in [vdWS06], deriva dalla concatenazione dell'istogramma di colore HSV normalizzato rispetto alla tonalità con il corrispondente descrittore SIFT, calcolato su patch locali; dato che lo hue histogram viene rappresentato su 37 bin, la dimensione finale del descrittore è 165.

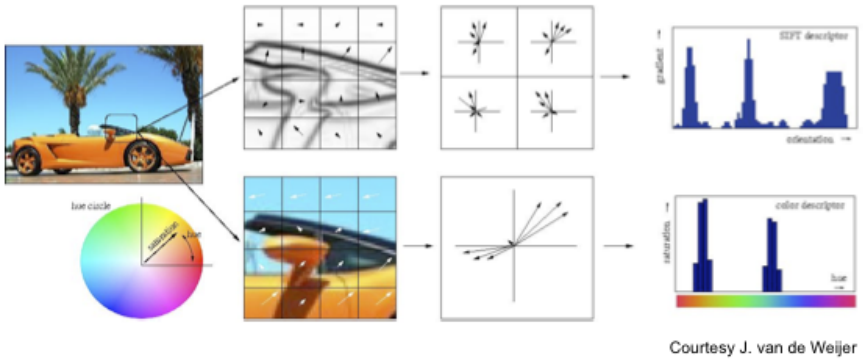

Figura 2.4. Creazione del descrittore HueSIFT

OpponentSIFT: Descrive tutti i canali dello spazio di colore *opponent* con SIFT:

$$\begin{pmatrix} O_1 \\ O_2 \\ O_3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

Il canale O_3 fornisce informazioni sulla luminanza, O_1 ed O_2 sul colore. Anche in questo caso si ottiene un descrittore finale di dimensione $3 \times 128 = 384$ per ogni keypoint.

C-SIFT: dato che i canali O_1 e O_2 dello spazio di colore opponent contengono comunque informazioni di luminanza, viene introdotto anche un descrittore ad "invariante di colore" C-SIFT che consiste nel normalizzare i due canali rispetto alla terza componente: O1/O3, O2/O3; il descrittore risultante ha le stesse dimensioni di OpponentSIFT (384).

rgSIFT: Il modello RGB normalizzato viene calcolato come segue:

$$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} \frac{R}{R+G+B} \\ \frac{G}{R+G+B} \\ \frac{B}{R+G+B} \end{pmatrix}$$

Nel descrittore rgSIFT vengono aggiunti al SIFT classico i descrittori per le componenti cromatiche r e g di tale modello (dim. 384).

tcSIFT: la *transformed color distribution* deriva da RGB ma a differenza di questo risulta invariante a modifiche di illuminazione:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R-\mu_R}{\sigma_R} \\ \frac{G-\mu_G}{\sigma_G} \\ \frac{B-\mu_B}{\sigma_B} \end{pmatrix}$$

dove μ_C e σ_C rappresentano media e deviazione standard della distribuzione nel canale C calcolate nell'area di interesse per il descrittore (patch locale). Il descrittore SIFT viene applicato ai tre canali RGB normalizzati così ottenuti (dim. 384).

RGB-SIFT: descrittore SIFT applicato ad ognuno dei tre canali R, G e B separatamente (dim. 384).

2.3 Misurazioni di distanza visuale

Dopo aver definito le principali features visuali, è opportuno valutare distanze che permettano di confrontare in modo efficace la similarità tra immagini.

Definizione 2.1. Date due immagini I_1, $I_2 \in I$, definiamo una metrica tra immagini:

$$\mu_V(I_1, I_2) : I \times I \to \mathbb{R} \tag{2.5}$$

dove valgono le proprietà $\forall x, y, z \in I$

$$\mu_V(x, y) \geq 0 \tag{2.6}$$

$$\mu_V(x, y) = 0 \Leftrightarrow x = y \tag{2.7}$$

$$\mu_V(x, y) = \mu_V(y, x) \tag{2.8}$$

$$\mu_V(x, y) \leq \mu_V(x, z) + \mu_V(z, y) \tag{2.9}$$

La distanza μ_V si basa sulle features visuali estratte dalle immagini. Data un'immagine I_1, si estraggono K features di diverso tipo $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_K\}$ con $\mathbf{X}_i \in \mathbb{R}^{K_i \times D_i}$. K_i sono il numero di feature del singolo tipo, D_i è la dimensione della singola feature di quel tipo. Ogni tipo di feature deve essere normalizzata secondo una norma a propria scelta (ad esempio la norma di Frobenius) per poter essere comparabile con le altre. Date due insiemi di features $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{D_i}$, le distanze più utilizzate per i singoli tipi di features sono:

- la distanza Manhattan (L1):

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.10}$$

- la distanza euclidea (L2):

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.11}$$

- la distanza χ^2, largamente usata in statistica:

$$\chi^2(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \frac{1}{2} \sum_i \frac{(x_i - y_i)^2}{x_i + y_i} \tag{2.12}$$

  dove

$$\bar{\mathbf{x}} = [\bar{x_1}, \ldots, \bar{x_n}] \quad \bar{x_i} = \frac{x_i}{\sum_{k=1}^{n} x_k}$$

Avendo K tipi di features, abbiamo potenzialmente K metriche diverse. Per fonderle insieme e realizzare infine la distanza fra immagini μ_V è possibile utilizzare un metodo semplice denominato JEC (Joint Equal Contribution) dove ogni metrica contribuisce in modo eguale alle altre:

$$\mu_V(I_1, I_2) = \frac{\sum_{k=1}^{K} d(\mathbf{x}, \mathbf{y})}{K} \tag{2.13}$$

È importante che ogni features sia stata prima normalizzata (ovvero le features devono avere norma 1) in modo che abbiano tutte lo stesso contributo.

3

Fattorizzazione di matrici e tensori

The Matrix is everywhere. It is all around us. Even now, in this very room.

– Morpheus, *Matrix*

La fattorizzazione di matrici è un argomento importante ed unificante che ha trovato numerose applicazioni in molte aree scientifiche [DLJ10] [DLPP06] [KZWS07]. L'idea principale è quella di riuscire a decomporre una matrice (o generalmente un tensore a più vie) di dati in un modello additivo o moltiplicativo lineare. Tensori e loro decomposizioni sono state largamente utilizzati in misurazioni chimiche e più recentemente sono stati applicati al *data mining* e a problemi di *machine learning*, ad esempio per modellare effetti temporali in social networks. Nel campo dell'apprendimento relazionale sono emersi recentemente come approccio alternativo ai modelli grafici. Alcune tecniche allo stato dell'arte nel raffinamento dei tag fanno uso di tecniche di questo tipo: procediamo quindi ad una breve descrizione degli strumenti necessari ad utilizzare questo tipo di tecniche.

Dal punto di vista della modellazione sono apprezzati per la semplicità, per la scalabilità (in quanto modelli lineari) e flessibilità sui vincoli. E' possibile modellare relazioni n-arie semplicemente come tensore di ordine superiore ed inoltre non è necessaria alcuna conoscenza a priori della struttura del

problema, differentemente rispetto ai modelli a reti bayesiane oppure le reti Markov. La decomposizione tensoriale ha mostrato buoni risultati in domini ad alta dimensionalità e sparsità, elementi presenti ad esempio nell'analisi della relazione ternaria tra utenti, immagini e tag descritta in precedenza. Per contro è generalmente difficile modellare l'incertezza (come probabilità) nei dati [NTK11] utilizzando questo genere di modelli.

Le principali tecniche di decomposizione sono:

- Decomposizione LU
- Decomposizione QR
- Decomposizione Cholesky
- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)
- Nonnegative Matrix Factorization (NMF) e le sue estensioni quali multilayer NMF, NMF sparsa, NMF simmetrica, NMF ortogonale, etc.
- PARAFAC [Har70]
- Tucker Decomposition [Tuc66] e le sue estensioni quali Nonnegative Tucker Decomposition, Constrained Tucker Decomposition, etc.

Probabilmente le fattorizzazione più comprese sono le prime cinque, tuttavia lo studio dei vincoli di nonnegatività e di sparsità stanno eseguendo un ruolo chiave nel cercare di comprendere e migliorare l'apprendimento di dati a larga scala. I modelli PARAFAC e Tucker, nonostante siano stati ideati decenni fa, sono recentemente riportati alla ribalta grazie alla possibilità di utilizzare computer più potenti che ne rendono possibile la modellazione in tempi accettabili. Questi due modelli, rispetto a quelli descritti sopra, sono considerati generalizzazioni di ordine maggiore di 2: chiaramente una visione piatta del mondo a sole due dimensioni risulta insufficiente per modellare i dati in certe applicazioni dove sono presenti soggetti multipli, più sorgenti o più relazioni. L'analisi a più vie risulta una scelta naturale in applicazioni multicanale con

spazio, tempo e frequenza, rappresentazioni sparse, estrazioni di features, clustering a più vie e così via.

Uno dei problemi che la neuroscienza si è trovata ad affrontare è stato quello della separazione cieca di segnali, ovvero la ricostruzione di segnali che non hanno alcuna possibilità di essere controllati direttamente. Un esempio tipico è quello della ricostruzione di segnali cerebrali a partire dalle misurazioni con elettrodi applicati al di fuori della scatola cranica: i segnali da misurare sono generati all'interno della scatola, sono fusi con disturbi ed interferenze e poi infine misurati agli elettrodi. Non c'è modo di ricostruire esattamente il segnale originario, ma grazie all'applicazione della fattorizzazione di matrici, è possibile averne una stima pur senza aver accesso ai dati corretti. Si tratta quindi di poter separare il rumore dal segnale senza conoscere alcun parametro o processo di filtraggio in atto. In effetti senza conoscere alcuna informazione **a priori** non è possibile stimare i segnali in modo **univoco**. In ogni caso è possibile eseguirne una stima avendo comunque una certa indeterminatezza, tipo ad esempio una permutazione dei segnali o un'amplificazione diversa. Nonostante possa sembrare limitante, in un gran numero di applicazioni non è importante in quanto le forme rappresentate rimangono comunque informative. Date una serie di osservazioni discrete $\mathbf{Y} = [y_{i,j}] = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T] \in \mathbb{R}^{I \times T}$ derivanti da una serie di sorgenti di input $\mathbf{x}_j$ con $j = 1, 2, \ldots, J$, si vuole eseguire la fattorizzazione in modo da ottenere:

$$\mathbf{Y} = \mathbf{AX} + \mathbf{E} \tag{3.1}$$

dove $A \in \mathbb{R}^{I \times J}$ rappresenta le basi (non conosciute) dello spazio o la matrice di filtraggio; $E^{I \times J}$ rappresenta l'errore della ricostruzione e $X \in \mathbb{R}^{I \times J}$ rappresenta il segnale latente o nascosto ed indica il contributo di ogni base. T è la dimensionalità del singolo campione, I è il numero di osservazioni e J è il numero di sorgenti o componenti. Generalmente i fattori latenti rappresentano i segnali sorgente sconosciuti, con una qualche proprietà statistica. Il numero J è sconosciuto e può essere sia più piccolo, uguale o più grande rispetto al numero di osservazioni. Definire il numero di componenti è ancora un problema aperto.

A partire da questa definizione base, i vari modelli enunciati carcano di ricostruire i segnali sulla base di qualche proprietà statistica, ad esempio PCA cerca una rappresentazione a bassa dimensionalità catturando la maggior parte della varianza mentre NMF cerca di spiegare i dati usando una mistura di componenti additive localizzate.

Per quanto riguarda lo studio delle decomposizioni classiche rimandiamo alla letteratura. Procediamo invece con lo studio dei modelli che rivestono maggior importanza in tempi recenti, ovvero SVD, NMF, PARAFAC e Tucker.

3.1 Singular Value Decomposition (SVD)

La singular value decomposition (SVD) è una delle decomposizioni più importanti del calcolo matriciale. Utilizzandola opportunamente è possibile risolvere moltissimi problemi, sia in algebra lineare, sia in molti altri ambiti. Alcuni esempi sono la determinazione pratica del rango di una matrice, la risoluzione del problema lineare dei minimi quadrati, la ricostruzione di immagini sfuocate ed affette da rumore.

Data una qualsiasi matrice rettangolare con coefficienti in $\mathbb{C}$ la decomposizione SVD esiste sempre.

Teorema 3.1. *Sia $A \in \mathbb{C}^{m\times n}$. Allora esistono una matrice unitaria $U \in \mathbb{C}^{m\times m}$ e una matrice unitaria $V \in \mathbb{C}^{n\times n}$ tali che:*

$$A = U\Sigma V^H \tag{3.2}$$

dove la matrice $\Sigma \in \mathbb{C}^{m\times m}$ ha elementi $\sigma_{i,j}$ nulli per $i \neq j$ e per $i = j$ ha elementi $\sigma_{i,i} = \sigma_i$ con

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0 \qquad p = \min\{m, n\} \tag{3.3}$$

La dimostrazione (che rimandiamo alla letteratura) è per induzione su n e si articola in tre passi: dimostrazione della tesi per $n = 1$, induzione di una prima decomposizione non a valori singolari con due particolari matrici

unitarie ed infine induzione utilizzando il precedente passo per dimostrare la decomposizione a valori singolari.

La decomposizione a valori singolari è quindi definita:

$$U^H AV = \Lambda^{1/2} = diag[\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_m}] \tag{3.4}$$

ovvero è una trasformazione che diagonalizza la matrice A. Limitandosi al dominio dei numeri reali, basta semplicemente sostituire l'operatore hermitiano con l'operatore trasposto:

$$U^T AV = \Lambda^{1/2} = diag[\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_m}] \tag{3.5}$$

È inoltre possibile applicare la decomposizione SVD come trasformata, di particolare utilità nella compressione delle immagini. Partendo dal presupposto di avere una matrice non a rango massimo, è possibile eseguire una decomposizione 'economica', oppure eseguire una compressione *lossy* dei dati conservando soltanto una parte dei valori singolari in base al loro valore energetico.

Sia $A = [a_{i,j}] \in \mathbb{R}^{M \times N}$ una matrice bidimensionale con rango $R \leq M \leq N$. Consideriamo il seguente problema di autovalori per $A^T A$ e AA^T:

$$AA^T u_i = \lambda_i u_i$$
$$A^T A u_i = \lambda_i v_i$$

dove λ_i con $(i = 1, 2, \ldots, R)$ sono gli autovalori sia di AA^T che di $A^T A$, u_i e v_i sono gli autovettori di AA^T e di $A^T T$ rispettivamente. Poiché le matrici in considerazione sono simmetriche, i solo autovettori sono ortogonali:

$$u_i^T u_j = v_i^t v_j = \delta_{i,j}$$

e quindi formano due matrici ortogonali U e V:

$$U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_M] \in \mathbb{R}^{M \times M}$$
$$V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N] \in \mathbb{R}^{N \times N}$$

$$UU^T = U^T U = P$$
$$VV^T = V^T V = P$$

Poiché esistono solo R autovalori diversi da 0, sia U che V hanno alcune colonne di valori pari a 0 e la matrice P ha solo R valori diversi da 0 sulla diagonale. Quindi le matrici U e V diagonalizzano AA^T e A^TA rispettivamente:

$$U^T(AA^T)U = \Lambda_{M\times M} = diag[\lambda_1, \lambda_2, \cdots, \lambda_R]$$
$$V^T(A^TA)V = \Lambda_{N\times N} = diag[\lambda_1, \lambda_2, \cdots, \lambda_R]$$

La trasformazione SVD è quindi definita come

$$U^TAV = \Lambda^{1/2} = diag[\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_R}] \tag{3.6}$$

e la sua forma inversa

$$A = U\Lambda^{1/2}V^T = \sum_{i=1}^{R} \sqrt{\lambda_i}[u_i v_i^T] \tag{3.7}$$

La trasformazione SVD ha avuto impiego positivo nella compressione delle immagini e risulta utile anche per valutare l'apporto energetico dei singoli autovettori in relazione alla matrice descritta (come ad esempio l'energia contenuta in un'immagine).

3.2 Nonnegative Matrix Factorization

Una grande quantità di dati in applicazioni reali è generalmente non negativa e le sue componenti hanno un significato fisico tangibile solo nel caso in cui siano non negative. Molto spesso le proprietà di nonnegatività e di sparsità sono altamente desiderabili o necessarie in quanto le componenti latenti hanno un'interpretazione fisica. Per esempio i segnali delle immagini sono generalmente non negativi in quanto non possono essere generati segnali che ne annullano altri. In questo caso la decomposizione potrebbe essere la divisione in parti delle immagini rilevanti (le basi), in modo da poter estrarre una qualche forma o oggetto. L'importanza di avere una rappresentazione sparsa è importante in quanto per alcune applicazioni è necessario avere un numero di componenti attive ristretto. Ciò è utile anche per spiegare i dati. In *machine learning* il problema della nonnegatività è comune a quello relativo alle distribuzioni di probabilità ed il problema della sparsità è relativo

alla selezione delle features. Ovviamente cercare una soluzione con questi vincoli può risultare in una soluzione che meno spiega la varianza dei dati che incontriamo. Questo indica che esiste un trade-off tra la possibilità di spiegare la soluzione e la possibilità di ricostruire efficacemente i segnali di uscita modellati. Magari ci accontentiamo di spiegare una quantità di dati ma non tutti e ciò dipende fortemente dalle applicazioni. Quando è imposto il vincolo di nonnegatività, le basi contano come elementi additivi e i fattori latenti ne regolano la quantità. Non esistendo la possibilità di sottrarre, le basi o componenti sono presenti (e in una certa quantità) oppure no. Ad esempio nel caso di dati facciali, il modello NMF con vincolo di sparsità è stato in grado di individuare componenti come barba, naso, occhi, capelli e di miscelarne le quantità [Hoy04].

3.2.1 Definizione

La tecnica di fattorizzazione di matrici nonnegativa è stata investigata in passato da molti ricercatori, ma ha ottenuto popolarità dal lavoro di Lee e Seung [LS00] dove gli autori propongono un semplice algoritmo per individuare la forma non negativa da applicare alle immagini. Il problema è così formulato: data una matrice di dati non negativa $\mathbf{Y} \in \mathbb{R}_{+}^{I \times T}$ dove $y_{i,j} \geq 0 \ \forall (i,j) \in I \times T$, con $rank(Y) \leq \min(I,T)$, trovare due matrici non negative $\mathbf{A} \in \mathbb{R}^{I \times J}$ e $\mathbf{B} \in \mathbb{R}^{J \times T}$ tali che fattorizzano Y al meglio possibile. La forma ricercata è quindi:

$$\mathbf{Y} = \mathbf{AB} + \mathbf{E} \tag{3.8}$$

dove la matrice $\mathbf{E}$ è l'errore di ricostruzione o di approssimazione. Il significato che possiamo attribuire ad $\mathbf{A}$ e a $\mathbf{B}$ dipende generalmente dall'applicazione. Ad una delle due (generalmente $\mathbf{A}$) è attribuito il significato di base, come ad esempio segnali di base, pattern di base e così via. Nel caso in cui l'errore $\mathbf{E} = \mathbf{0}$ allora si parla di **nonnegative rank factorization** (NRF) a significare la ricostruzione esatta della matrice originale. Il più piccolo intero J per il quale è possibile ottenere una rappresentazione nonnegative rank per i dati specifici è chiamato **nonnegative rank** della matrice $\mathbf{Y}$ ed è indicato

con $rank_+(\mathbf{Y})$. Una decomposizione NMF non è quindi necessariamente una NRF. Si tratta in genere di un'approssimazione. Esiste anche un vincolo sui ranghi [KP07]:

$$rank(\mathbf{Y}) \leq rank_+(\mathbf{Y}) \leq \min(I, T) \tag{3.9}$$

ovvero il rango necessario alla ricostruzione completa non negativa di una matrice è pari o superiore al rango della matrice stessa. Nel caso peggiore potrebbe essere il rango massimo per le dimensioni della matrice.

3.2.2 NMF Simmetrica

Nel caso in cui la matrice $\mathbf{A} = \mathbf{B} \in \mathbb{R}_+^{I \times J}$, la decomposizione è detta simmetrica (NMF simmetrica) ed è data da

$$\mathbf{Y} = \mathbf{A}\mathbf{A}^T + \mathbf{E} \tag{3.10}$$

Questo modello è considerato equivalente all'algoritmo di kernel K-means e al clustering spettrale [JQ07].

3.2.3 Tri-NMF

Nel caso di tre fattori, si parla di fattorizzazione nonnegativa a tre fattori (tri-NMF) e prende come forma generale

$$\mathbf{Y} = \mathbf{A}\mathbf{S}\mathbf{X} + \mathbf{E} \tag{3.11}$$

con $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{S} \in \mathbb{R}^{J \times R}$ e $\mathbf{X} \in \mathbb{R}^{R \times T}$. La forma a tre fattori è riconducibile alla forma NMF standard sostituendo $\mathbf{A}$ al posto di $\mathbf{AS}$ oppure $\mathbf{X}$ al posto di $\mathbf{SX}$, purché non vengano aggiunti alcun vincolo o condizione speciale.

3.2.4 Algoritmo ALS

L'algoritmo più semplice per calcolare la fattorizzazione è denominato ALS (Alternating Least Squares) e calcola la forma ricercata iterando il procedimento di fissare tutte le matrici a parte una alla volta e risolvendo il sotto problema. Prima di tutto è necessario definire una misura della dissimilarità

(definita anche distanza, divergenza) tra la matrice ricostruita e la matrice originale. La scelta della funzione dipende dalla distribuzione di probabilità delle componenti stimate oppure sulla struttura dei dati o il tipo di rumore presente. La più semplice e più utilizzata è la misura di distanza basata sulla norma di Frobenius o distanza euclidea:

$$D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}||\mathbf{Y} - \mathbf{AX}||_F^2 \tag{3.12}$$

È importante notare che la misura qui enunciata è convessa rispetto alle singoli matrici $\mathbf{A}$ e $\mathbf{X}$ ma non rispetto ad entrambe. Ciò si traduce in un problema di ottimizzazione non convesso nel caso sia $\mathbf{A}$ e sia $\mathbf{X}$ siano variabili libere. Invece la singola ottimizzazione di una matrice alla volta, fissando l'altra è un problema convesso. A partire da questa affermazione possiamo definire l'algoritmo di ottimizzazione ALS:

1. Inizializzare $\mathbf{A}$ e $\mathbf{X}$ utilizzando una qualsiasi strategia.
2. Stimare $\mathbf{X}$ risolvendo $\min_{\mathbf{X}} D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}||\mathbf{Y}-\mathbf{AX}||_F^2$ con $\mathbf{A}$ fissata.
3. Porre tutti gli elementi di $\mathbf{X}$ a 0 o a qualche piccolo valore positivo ϵ.
4. Stimare $\mathbf{A}$ risolvendo $\min_{\mathbf{A}} D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2}||\mathbf{Y}^T - \mathbf{X}^T\mathbf{A}^T||_F^2$ con $\mathbf{X}$ fissata.
5. Porre tutti gli elementi di $\mathbf{A}$ a 0 o a qualche piccolo valore positivo ϵ.

Questo algoritmo, pur essendo semplice, non è molto accurato e non è garantito che trovi il minimo globale, né tantomeno un punto stazionario. È in grado di trovare soltanto una soluzione dove il costo smette di decrescere. Viene quindi considerato solo per la semplicità di programmazione. È interessante osservare che la sola imposizione dei valori positivi dei punti 3 e 5 è sufficiente a garantire il vincolo della nonnegatività.

3.3 Fattorizzazione di tensori

Le fattorizzazioni discusse finora sono facilmente estendibili e generalizzati ai vettori a multipla entrata, chiamati matrici multi dimensionali o più semplicemente tensori. Si parla quindi di fattorizzazione tensoriale. Si noti che

la definizione di tensore qui riportata differisce dalla definizione di tensore di campo generalmente utilizzati in fisica.

3.3.1 Definizione e operatori di base

Un tensore di ordine p è una matrice multidimensionale di p dimensioni, chiamati anche vie o modi. La definizione formale è:

Definizione 3.1. Siano $I_1, I_2, \ldots, I_N \in \mathbb{N}$ i limiti superiori per ogni via. Un **tensore** $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ di ordine N è un vettore a N entrate dove gli elementi $y_{i_1,i_2,\ldots,i_N}$ sono indicizzati da $i_n \in \{1, 2, \ldots, I_N\}$ per $1 \leq n \leq N$.

Si tratta quindi di una generalizzazione di vettori e di matrici ad ordini maggiori. Un tensore di ordine 0 è uno scalare, un tensore di ordine 1 è un vettore ed infine un tensore di ordine 2 è una matrice. I tensori di ordine maggiore a 2 sono chiamati semplicemente tensori o tensori di ordine superiore.

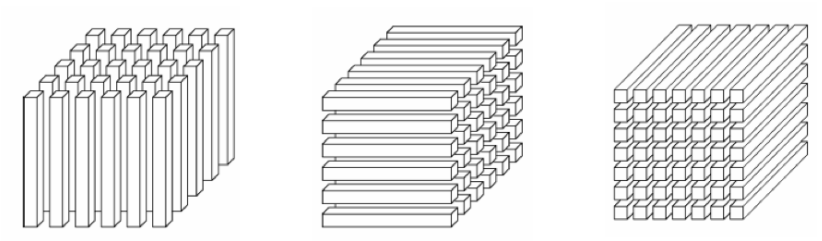

Figura 3.1. Esempi di fibre di un tensore di ordine 3. In questo caso si usano anche le definizioni *fibra*, *riga* o *tubo* in base alla dimensione libera.

Se fissiamo uno o più indici possiamo selezionare un sottoinsieme di elementi chiamati rispettivamente fibre, fette e sottotensori, in base al numero di indici liberi. Sono il rispettivo tensoriale delle righe e colonne delle matrici. Generalmente si utilizza la notazione di MATLAB, i due punti, per indicare un indice libero. Ad esempio in una matrice $\mathbf{A}$ (un tensore di ordine 2), per indicare gli elementi della 3 riga si può utilizzare la sintassi $\mathbf{A}_{3,:}$, mentre per indicare la seconda colonna si può utilizzare $\mathbf{A}_{:,2}$. Lo stesso, in un tensore $\underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_J}$ per indicare gli elementi con il k esimo indice fissato, possiamo utilizzare la notazione $\underline{\mathbf{B}}_{:,k}$.

Definizione 3.2. Una **fibra** di un tensore è un frammento unidimensionale ottenuto fissando tutti gli indici eccetto uno.

Definizione 3.3. Una **fetta** di un tensore è una sezione bidimensionale ottenuto fissando tutti gli indici eccetto due.

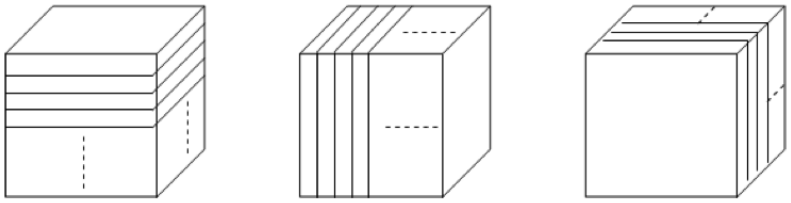

Figura 3.2. Esempi di fette di un tensore di ordine 3.

Un'altra operazione frequentemente utilizzata è la matricizzazione (*unfolding*) di un tensore, ovvero la rappresentazione di un tensore utilizzando soltanto matrici. Ci sono diversi modi per ordinare le fibre di un tensore e quindi il processo di *unfolding* non è unico. Per un tensore di ordine 3, è possibile eseguire la matricizzazione utilizzando fette frontali, orizzontali o laterali in base alla divisione per profondità, righe o colonne. Anche l'ordine delle colonne non è univoco.

Definizione 3.4. La matricizzazione di un tensore $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_J}$ secondo il modo n è l'ordinamento delle fibre del modo n nelle colonne di una matrice. L'operazione è denotata con $\mathbf{A}_{(n)}$. Dato un'indice $(i_1, i_2, \ldots, i_J)$, l'elemento del tensore corrispondente viene mappato nella matrice all'indice (i_n, j) dove

$$j = 1 + \sum_{p \neq n} (i_p - 1) K_p \tag{3.13}$$

dove

$$K_p = \begin{cases} 1 & \text{se } p = 1, \text{ o } p = 2 \text{ e } n = 1 \\ \prod_{m \neq n}^{p-1} I_m & \text{altrimenti} \end{cases} \tag{3.14}$$

Analoga all'operazione di unfolding, si può eseguire un'operazione di vettorizzazione ossia la rappresentazione di un tensore utilizzando vettori.

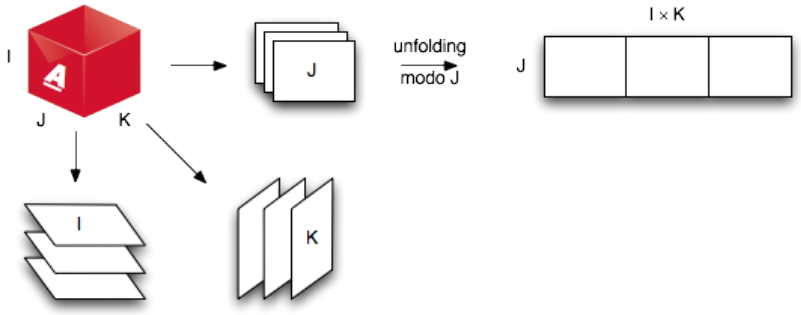

Figura 3.3. Esempio di unfolding di un tensore di ordine 3 rispetto al modo J.

Definizione 3.5. Data una matrice $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_J] \in \mathbb{R}^{I \times J}$, la vettorizzazione è definita come

$$\mathbf{Y} = \text{vec}(\mathbf{Y}) = [\mathbf{y}_1^T, \mathbf{y}_2^T, \ldots, \mathbf{y}_J^T]^T \in \mathbb{R}^{IJ} \tag{3.15}$$

L'operatore **vec** così definito permette di rappresentare una matrice ordinando gli elementi della matrice in un vettore secondo l'ordinamento a colonne. Analogamente è possibile definire un operatore **vec** anche per tensori passando per la forma matricizzata.

Definizione 3.6. Dato un tensore $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_J}$, la vettorizzazione per il modo n è definita come

$$\text{vec}(\underline{\mathbf{Y}}) = \text{vec}(\underline{\mathbf{Y}}_{(n)}) = \left[\text{vec}(\mathbf{Y}_{:1})^T, \text{vec}(\mathbf{Y}_{:2})^T, \ldots, \text{vec}(\mathbf{Y}_{:I_n})^T\right]^T \in \mathbb{R}^{I_1 \cdot I_2 \cdots I_J} \tag{3.16}$$

L'operatore **vec** è molto importante ed è spesso utilizzato sia per semplificare la programmazione dei modelli che per ottenere forme semplificate delle decomposizioni.

3.3.2 Altri operatori importanti

Una serie di operatori speciali sono molto importanti ai fini delle decomposizioni e ricostruzioni.

Definizione 3.7. Il **prodotto vettoriale** (outer product) di un tensore $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ con un altro $\underline{\mathbf{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ è dato da:

$$\underline{\mathbf{C}} = \underline{\mathbf{A}} \circ \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times J_2 \cdots \times J_M} \tag{3.17}$$

dove gli elementi di $\underline{\mathbf{C}}$ sono

$$c_{i_1,i_2,\ldots,i_n,j_1,j_2,\ldots,j_m} = a_{i_1,i_2,\ldots,i_n} b_{j_1,j_2,\ldots,j_m} \tag{3.18}$$

Il prodotto vettoriale tra due tensori produce tutte le possibili moltiplicazioni possibili tra gli elementi.

Definizione 3.8. Il **prodotto scalare** (inner product) di due tensori $\underline{\mathbf{A}}$,$\underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ dello stesso ordine è calcolato come somma del prodotto elemento per elemento su tutti gli indici.

$$c = < \underline{\mathbf{A}}, \underline{\mathbf{B}} > = \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} b_{i_1,i_2,\ldots,i_N} a_{i_1,i_2,\ldots,i_N} \in \mathbb{R} \tag{3.19}$$

La definizione di prodotto scalare ci permette di definire la norma p per tensori. Dato un tensore $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$:

$$||\underline{\mathbf{A}}||_p = \sqrt[p]{< \underline{\mathbf{A}}, \underline{\mathbf{A}} >} = \sqrt[p]{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} a^p_{i_1,i_2,\ldots,i_N}} \tag{3.20}$$

Definizione 3.9. Il **prodotto di Kronecker** tra due matrici $\mathbf{A} \in \mathbb{R}^{I \times J}$ e $\mathbf{B} \in \mathbb{R}^{T \times R}$ è:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{3.21}$$

Definizione 3.10. Il **prodotto di Hadamard** tra due matrici $\mathbf{A} \in \mathbb{R}^{I \times J}$ e $\mathbf{B} \in \mathbb{R}^{I \times J}$ è:

$$\mathbf{A} \circledast \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix} \tag{3.22}$$

Infine il prodotto che viene utilizzato nelle decomposizioni successive è quello tra un tensore ed una matrice e quello tra un tensore un vettore. Per eseguire la moltiplicazione è necessario indicare il modo su cui la si esegue.

Definizione 3.11. Il **prodotto di modo-n** di un tensore $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ per una matrice $\mathbf{A} \in \mathbb{R}^{I_n \times J_n}$ è il tensore:

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_n \mathbf{A} \in \mathbb{R}^{J_1 \times J_2, \times \cdots \times J_{n-1} \times I_n \times J_{n+1} \times \cdots \times J_n} \tag{3.23}$$

dove i singoli elementi di $\underline{\mathbf{Y}}$ sono:

$$y_{j_1, j_2, \ldots, j_{n-1}, i_n, j_{n+1}, \ldots, j_N} = \sum_{j_n=1}^{J_n} g_{j_1, j_2, \ldots, j_N} a_{i_n, j_n} \tag{3.24}$$

Il tensore risultante ha le dimensioni di tutti i modi invariati eccetto quello su cui si esegue la moltiplicazione. Il prodotto può essere eseguito anche su più modi contemporaneamente ed è commutativo. Nel caso in cui si voglia eseguire la moltiplicazione lungo un modo n di più di una matrice, è equivalente ad eseguire la moltiplicazione tra matrici ed infine il prodotto con il tensore.

$$(\underline{\mathbf{G}} \times_n \mathbf{A}) \times_m \mathbf{B} = (\underline{\mathbf{G}} \times_m \mathbf{B}) \times_n \mathbf{A} \qquad (m \neq n) \tag{3.25}$$

$$(\underline{\mathbf{G}} \times_n \mathbf{A}) \times_n \mathbf{B} = \underline{\mathbf{G}} \times_n \mathbf{AB} \tag{3.26}$$

Anche per quanto riguarda il prodotto di un tensore con un vettore è necessario scegliere il modo su cui avviene la moltiplicazione. La definizione deriva direttamente da quella della moltiplicazione con una matrice. La differenza è che il vettore ha solo una dimensione e quindi si ha uno 'schiacciamento' del tensore lungo il modo scelto.

Definizione 3.12. Il **prodotto di modo-n** di un tensore $\underline{\mathbf{G}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ per un vettore $\mathbf{a} \in \mathbb{R}^{J_n}$ è il tensore:

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_n \mathbf{a} \in \mathbb{R}^{J_1 \times J_2, \times \cdots \times J_{n-1} \times J_{n+1} \times \cdots \times J_n} \tag{3.27}$$

dove i singoli elementi di $\underline{\mathbf{Y}}$ sono:

$$y_{j_1, j_2, \ldots, j_{n-1}, j_{n+1}, \ldots, j_N} = \sum_{j_n=1}^{J_n} g_{j_1, j_2, \ldots, j_N} a_{j_n} \tag{3.28}$$

Anche qui valgono le stesse affermazioni riguardo alla moltiplicazione su più modi che già valevano per le matrici. Se si esegue una moltiplicazione del tensore per un vettore in ogni modo, si ottiene alla fine uno scalare.

3.3.3 Modello PARAFAC o CANDECOMP

Il modello PARAFAC (chiamato anche CANDECOMP), insieme al modello di Tucker, risulta essere una delle decomposizioni per modelli tensoriali a N vie più popolari. Il modello PARAFAC può essere così formulato: dato un tensore $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ ed un numero di variabili positivo intero J, si ricercano tre matrici $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_J,] \in \mathbb{R}^{I \times J}$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_J,] \in \mathbb{R}^{T \times J}$ e $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_J,] \in \mathbb{R}^{Q \times J}$ che rappresentano le componenti di carico. Dove $\mathbf{a}_j = [a_{i,j}] \in \mathbb{R}^I$, $\mathbf{b}_j = [b_{t,j}] \in \mathbb{R}^T$ e $\mathbf{c}_j = [c_{q,j}] \in \mathbb{R}^Q$. Il modello PARAFAC è un'approssimazione così definita:

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j + \underline{\mathbf{E}} \tag{3.29}$$

ovvero ogni elemento di $\underline{\mathbf{Y}}$ è definito come:

$$y_{itq} = \sum_{j=1}^{J} a_{i,j} b_{t,j} c_{q,j} + e_{itq} \tag{3.30}$$

PARAFAC decompone un tensore in una somma di termini trilineari in modo analogo alla decomposizione matriciale bilineare. Si tratta di una somma di tre tensori di rango 1. Il modello di per sé non impone alcun vincolo, nemmeno di ortogonalità come nella decomposizione SVD. È possibile aggiungere i vincoli di sparsità o di nonnegatività: in tal caso si parla di modello PARAFAC nonnegativo. La motivazione dietro a PARAFAC è di ottenere una soluzione unica in modo che le matrici delle componenti siano determinate univocamente a meno di una permutazione e di fattore di scala. La proprietà di unicità di PARAFAC, lo rende una tecnica popolare in vari campi. Purtroppo non esiste un metodo sicuro per individuare il numero di componenti J per dimensionare il modello. In letteratura sono proposti vari metodi, come ad esempio l'utilizzo della decomposizione SVD e una misurazione di

energia per la ricostruzione [AY09], ma spesso è preferibile utilizzare tecniche di diagnostica e testing manuale. Un'altra forma equivalente del modello

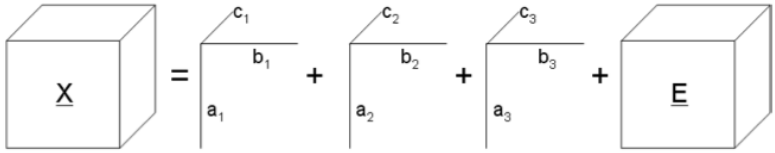

Figura 3.4. Il modello PARAFAC definito come somma di tre tensori di rango 1.

PARAFAC è dato dalla decomposizione:

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} + \underline{\mathbf{E}} \tag{3.31}$$

dove $\underline{\mathbf{G}} \in \mathbb{R}^{J \times J \times J}$ è un tensore diagonale con elementi λ_j sulla superdiagonale. Denominato anche modello di Harshman, è un caso speciale del modello di Tucker a 3 vie dove il tensore centrale è un tensore super identità. Il fatto di poter esprimere il modello con un tensore centrale, indica il tipo di interazioni possibili tra le componenti.

3.3.4 Modello Tucker3

Similmente a PARAFAC, anche Tucker3 è un'estensione di ordine superiore dell'analisi a fattori bilineari. In genere viene chiamata anche approssimazione a miglior rango in quanto emula il funzionamento di PCA in tensori di ordine maggiore di 2. Dato un tensore $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ e tre indici positivi J,R,P, tutti minori rispettivamente delle dimensioni I,T,Q del tensore da decomporre, si vuole calcolare un tensore centrale (core) $\underline{\mathbf{G}} = [g_{j,r,p}] \in \mathbb{R}^{J \times R \times P}$ e tre matrici $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_J] \in \mathbb{R}^{I \times J}$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_R] \in \mathbb{R}^{T \times R}$, $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_P] \in \mathbb{R}^{Q \times P}$ chiamate fattori latenti o matrici di carico. $\mathbf{a}_j \in \mathbb{R}^I$, $\mathbf{b}_j \in \mathbb{R}^T$ e $\mathbf{c}_j \in \mathbb{R}^Q$ La forma della decomposizione è:

$$\underline{\mathbf{Y}} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{j,r,p} (\mathbf{a}_j \circ \mathbf{b}_r \circ \mathbf{c}_p) + \underline{\mathbf{E}} \tag{3.32}$$

dove i singoli elementi di $\underline{\mathbf{Y}}$ sono:

$$y_{i,t,q} = \sum_{j=1}^{J} \sum_{r=1}^{R} \sum_{p=1}^{P} g_{j,r,p} a_{i,j} b_{t,r} c_{q,p} + e_{i,t,q} \tag{3.33}$$

dove gli elementi $g_{j,r,p}$ sono i fattori di scala presenti nel tensore centrale $\underline{\mathbf{G}} = [g_{j,r,p}] \in \mathbb{R}^{J \times R \times P}$. Una forma compatta del modello Tucker3 è esprimibile utilizzando la moltiplicazione in modo-n:

$$\underline{\mathbf{Y}} = \underline{\mathbf{G}} \times_1 \mathbf{A} \times_2 \underline{\mathbf{B}} \times_3 \underline{\mathbf{C}} + \underline{\mathbf{E}} \tag{3.34}$$

Rispetto a PARAFAC, il modello di Tucker risulta più flessibile grazie all'e-

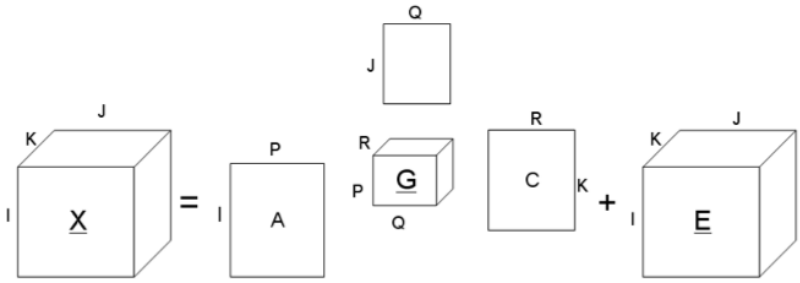

Figura 3.5. Il modello Tucker3: un tensore è decomposto in un tensore centrale di dimensioni inferiori e tre matrici contenenti i fattori latenti ricostruiti.

sistenza del tensore centrale $\underline{\mathbf{G}}$ che modella le interazioni tra tutti i fattori contemporaneamente. Tuttavia questa flessibilità è pagata come indeterminatezza alla rotazione del modello: non esiste un'unica decomposizione di Tucker in quanto applicando una matrice di rotazione ad una qualsiasi matrice dei fattori e l'inversa della matrice di rotazione al tensore nucleo si ottiene nuovamente una decomposizione equivalente. Un modello di Tucker3 può quindi individuare le matrici delle componenti a meno di una rotazione. Inoltre l'interpretazione di un modello di Tucker è più complessa rispetto al modello PARAFAC. Anche nel modello di Tucker è difficile individuare il numero di componenti latenti da utilizzare per ogni matrice: oltre ad utilizzare la decomposizione SVD sul tensore matricizzato, è consigliabile utilizzare una procedura di *model selection* quale ad esempio la *cross validazione*.

4

Stato dell'arte del tag refinement

Art, like morality, consists in drawing the line somewhere.
– G. K. Chesterton, *Arts magazine: Vol. 1 (1926)*

Questa tesi poggia le basi sulla ricerca più recente, basata sul contenuto visuale e non soltanto sull'analisi statistica della frequenza dei tag. In particolare si fa riferimento all'analisi modale e multimodale derivata dall'analisi del contenuto visivo dagli approcci basati su analisi dell'istogramma e bag of words, dall'analisi del significato semantico delle etichette, fino ad arrivare all'analisi delle relazioni presenti tra utenti, immagini e tag. Analizziamo dapprima i principali dataset disponibili in letteratura.

4.1 Datasets

La trasformazione data dal mondo social network è stata avvertita in tempi recenti anche nei confronti dei dataset. Per poter avere benchmark e studi attendibili per i problemi di categorizzazione e di raffinamento dei tag, anche i dataset sono passati ad includere le informazioni sociali. Analizziamo un dataset leggermente più anziano denominato "ESP Game" [Ahn06] basato su generiche immagini e successivamente due dataset di tempi più recenti "MIRFLICKR-25k" [HL08] (con la sua estensione "MIRFLICKR-

1M" [MJHL10]) e "NUSWIDE" [CTH+09]. Questi ultimi due basati sulle immagini reperite dal social network preferito dai fotografi: Flickr.

4.1.1 ESP Game

L'idea alla base di "ESP Game" è l'utilizzo di un videogioco online[1] per la creazione di un dataset a larga scala. Ogni anno moltissime persone passano del tempo a giocare al computer. Se si riuscisse in qualche modo a sfruttare le potenzialità di risoluzione di problemi di queste persone, probabilmente avremmo a disposizione una sorta di sistema "computerizzato" formato da umani in grado di risolvere task molto più complessi di quelli per computer. Un esempio famoso è il videogioco **FoldIt** [CKT+10], che riesce a predire la struttura delle proteine utilizzano i dati immessi dagli utenti mentre giocano.

ESP Game cerca di ottenere le etichette di un dataset attraverso l'utilizzo di un videogioco: data un'immagine contenente un solo oggetto, due utenti che non possono comunicare in alcun modo a parte per il gioco, si sfidano a cercare di indovinarlo. Entrambi devono indicare una serie di termini che meglio descrivono l'oggetto. L'obiettivo è introdurre lo stesso termine dell'altro utente e la ricompensa è un punto in caso venga inserito almeno un termine in comune. La stessa sfida si ripete per altri utenti e così via. Nel caso in cui i termini vincenti siano i soliti, ripetuti almeno un certo numero di volte, vengono aggiunti ai tag del dataset. Il punto di forza sta nel controllo successivo da parte di altri utenti, sotto forma sempre di inserimento di termini per quella immagine, che garantisce una maggior oggettività dei termini. È un po' come se gli utenti votassero un termine per ogni immagine e vincessero alla fine i termini più ripetuti.

Le immagini del dataset sono state recuperate da internet scegliendole in modo completamente casuale. Si tratta di 67.796 immagini di dimensione variabile ma generalmente al disotto della risoluzione di 320×240. Sono presenti sia immagini di paesaggi, oggetti o persone, sia immagini che sono elementi grafici di pagine web come banner, pubblicità e grafica vettoriale.

[1] ESP Game. `www.espgame.org`

Non rappresenta realmente le immagini inserite da utenti che fotografano, bensì materiale grafico qualsiasi presente sul web.

Il dizionario dei termini comprende 269 termini ed ogni immagine è associata massimo a 15 termini. In media le immagini hanno 4,6 tag.

4.1.2 MIRFLICKR-25k

Un dataset più recente, realizzato utilizzando le immagini rilasciate sotto licenza Creative Commons[2] su Flickr e costituito da un numero inferiore di immagini, ma con maggiore variabilità in termini di tag, denominato MIR Flickr. L'obiettivo degli autori era realizzare un dataset secondo alcuni criteri:

- **rappresentativo** per l'area di appartenenza. Essendo un dataset per uno studio dei *social network*, dovrebbe essere realizzato a partire dal contributo degli utenti in un social network reale
- disponibilità della **ground truth** completa realizzata da professionisti etichettatori
- facilmente **accessibile** e liberamente **condivisibile**
- **baseline standardizzate** come punto di partenza per le performance.

Il dataset è composto da 25.000 immagini reperite dal social network Flickr giornalmente per un periodo compreso tra marzo 2007 e giugno 2008. Le immagini sono recuperate da quelle che Flickr specifica essere del gruppo "interesting" e catturano in parte l'essenza del periodo e buona qualità di esposizione. La dimensione scelta per le immagini è la *medium* di Flickr ovvero sono ridimensionate in modo tale da avere 500 pixel sul lato più lungo. Le immagini recuperate sono state inserite da 9.862 utenti dei quali 5.566 sono rappresentati da una sola immagine. Il numero medio di tag è 8,94 per immagine, con 1.386 tag che compaiono in almeno 20 immagini. I tag più frequenti sono elencati in tabella 4.1. Inoltre per ogni immagine sono forniti i dati EXIF della fotocamera.

[2]Le licenze Creative Commons sono descritte in `www.creativecommons.org`

Figura 4.1. Un campione di immagini del dataset "ESP Game".

General topic	Subtopics
sky	clouds
water	sea/ocean, river, lake
people	portrait, boy/man, girl/woman, baby
night	
plant life*	tree, flower
animals	dog, bird
man-built structures*	architecture, building, house, city/urban, bridge, road/street
sunset	
indoor	
transport*	car

Figura 4.2. I concetti con groundtruth in MIRFLICKR-25K.

Tag	Numero di immagini
sky	845
water	641
portrait	623
night	621
nature	596
sunset	585
clouds	558
flower	510
beach	407
landscape	385

Tabella 4.1. I 10 tag più frequenti in MIRFLICKR-25K.

Creare la ground truth di un dataset di 25.000 immagini non è certo economico, sia in termini di tempo che di denaro: per questo sono state etichettate solo 27 classi di oggetti. 11 classi sono elementi di alto livello: *sky*, *water*, *people*, *night*, *plant life*, *animals*, *man-built structures*, *sunset*, *indoor*, *trasport*, *food*.

Il dataset è stato consigliato dagli autori per affrontare problemi di riconoscimento di concetti visuali basandosi solo sulle informazioni visive e la groundtruth, tag propagation basandosi sulle informazioni visuali, i tag e la groundtruth, e infine tag suggestion basandosi sulle informazioni visuali e i tag.

MIRFLICKR-1M

Successivamente il dataset MIRFLICKR è stato espanso portando il numero di immagini a 1 milione. Le prime 25.000 sono le stesse immagini del precedente dataset. In più sono presenti immagini di alta qualità derivate utilizzando il punteggio *interestingness* di Flickr insieme ai tag e ai dati EXIF. Sono forniti anche una serie di descrittori precalcolati per avere una baseline comune per quanto riguarda le informazioni visuali. Notare però che, a causa della grandezza del dataset, non è disponibile una groundtruth

Figura 4.3. Un campione di immagini del dataset "MIRFLICKR-25K".

fatta a mano. Il dataset MIRFLICKR esteso a 1 milione di immagini è stato utilizzato nell'ImageCLEF 2011 workshop.

4.1.3 NUSWIDE

Un altro dataset molto popolare negli ultimi 2 anni è NUSWIDE di Chua et al. [CTH+09]. Il dataset è stato realizzato recuperando da Flickr 269.648 immagini con i relativi tag associati. Si tratta esattamente di 5.018 tag unici, ottenuti da un insieme di 425.059 dopo aver rimosso i tag con meno di 100 apparizioni e i tag che non trovano corrispondenza in WordNet. I tag più frequenti sono elencati in tabella 4.2 e il grafico della distribuzione in figura 4.4. Notare che i tag sono testo, e come tali assumono una distribuzione

Tag	Numero di immagini
nature	20142
sunset	10962
sky	18935
light	10869
blue	17822
white	10082
water	17646
people	9324
clouds	14201
sea	9016

Tabella 4.2. I 10 tag più frequenti in NUSWIDE.

caratterizzata dalla cosiddetta legge di Zipf pubblicata dal linguista di Harvard George Kingsley Zipf. NUSWIDE non include le immagini ma bensì sei tipi diversi di features a basso livello estratte dalle immagini stesse:

- istogrammi di colore a 64 dimensioni
- correlogrammi di colore a 144 dimensioni
- istogrammi delle direzioni degli edge a 73 dimensioni

- tessiture wavelet a 128 dimensioni
- momenti di colore calcolati a blocchi di 225 dimensioni
- bag of words con vocabolario a 500 dimensioni basate su descrittore SIFT.

La fornitura delle features è per standardizzare i task pubblicizzati e, allo stesso tempo, per evitare la distribuzione delle immagini, in quanto non tutte sono libere da copyright per la diffusione libera.

É presente una groundtruth di 81 concetti ottenuta attraverso un processo semi manuale basato sull'annotazione manuale di tutto il dataset. Oltre a questo sono disponibili anche risultati standardizzati che valutano la precision, la recall, l'F_1 e la quantità di errore presente per ogni tag.

Rispetto a MIRFLICKR, il dataset NUSWIDE è più rappresentativo delle immagini presenti sui social network: il metodo con cui sono state recuperate non è basato sul punteggio di *interestingness* di Flickr, ma soltanto su parole chiave prestabilite. Mentre MIRFLICKR è più vario come genere di immagini, NUSWIDE tende ad avere più immagini degli stessi argomenti. Numero di immagini a parte, MIRFLICKR dovrebbe essere più facile da apprendere rispetto a NUSWIDE.

4.2 Tecniche principali

Per quanto riguarda il problema della categorizzazione delle immagini, i problemi del tag ranking e del tag refinement sono quelli direttamente collegati. Analizziamo la situazione in letteratura dei due problemi in questione.

4.2.1 Tag Ranking

Come descritto nel capitolo 1, il tag ranking è l'azione di associare un peso a ogni etichetta secondo la correlazione presente verso il contenuto visivo dell'immagine. Si tratta di un requisito importante per la ricerca di immagini [JLM03] [JB08] e permette a un motore di ricerca di selezionare immagini con maggior rilievo rispetto alle etichette di ricerca. La quasi totalità dei social

network (come ad esempio Flickr) non ha la possibilità di specificare l'ordine di importanza dei tag. La mancanza di un ordine di rilevanza ha portato a una limitazione delle applicazioni: basti osservare che non è possibile eseguire una ricerca di immagini presenti in Flickr sulla base di un criterio di rilevanza rispetto alle etichette ricercate, ma solo in base alla presenza o meno dei termini. Ad esempio vediamo in figura 4.5 un'immagine recuperata da Flickr, con un numero elevato di tag associati.

È possibile osservare che solo alcuni tag sono più rilevanti. I tag 'Lavandino' e 'gatto certosino' sono sicuramente i più importanti mentre altri quali 'iphone4' o 'Mastrobiggo' non sono assolutamente collegati con il contenuto visivo, anche se in realtà indicano la fotocamera usata e il soprannome dell'autore. Non è possibile stabilire questi valori soltanto dalla lista dei tag. Questo fenomeno è osservato frequentemente difatti in uno studio eseguito in [LHZ11] è stato misurato in modo quantitativo la frequenza di quale posizione occupa il tag più rilevante. Selezionate 1200 immagini a caso da Flickr, è stato chiesto a cinque volontari di selezionare il tag più rilevante di ogni immagine. Il risultato, visibile nel grafico in figura 4.6, evidenzia una distribuzione quasi uniforme lungo la posizione della lista e verifica l'impossibilità di affidarsi alla lista naturale come ordine valido di rilevanza.

Al momento della scrittura, esistono due categorie di tecniche che cercano di risolvere il problema del tag ranking:

- tecniche parametriche
- tecniche non parametriche, basate sulla propagazione di etichette.

Tecniche parametriche

Il primo lavoro che ha affrontato il problema del tag ranking è di D. Liu et al. [LHY+09], dove viene proposto il calcolo di un punteggio di rilevanza sulla base della *kernel density estimation*, e rifinita attraverso l'utilizzo di random walk. Data un'immagine e i suoi tag associati, il primo passo cerca di stimare il punteggio di rilevanza di ogni tag attraverso un approccio probabilistico dove viene considerata simultaneamente la probabilità di un tag data l'immagine e la sua abilità descrittiva nell'intero insieme delle immagini. I punteggi

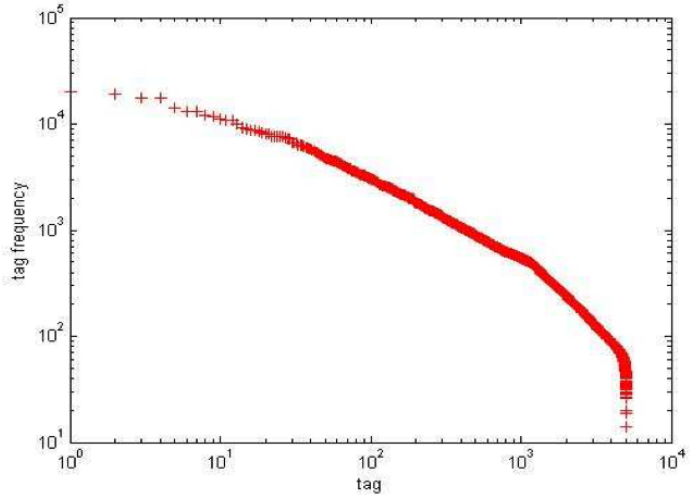

Figura 4.4. Frequenza dei tag in NUSWIDE.

Figura 4.5. Esempio di immagine con annessi tag recuperata da Flickr. Cortesia di Massimo Regonati, licenza CC BY-NC-SA 2.0.

ottenuti in questo modo riflettono la rilevanza dei tag, tuttavia utilizzando un approccio di raffinamento basato sulla random walk è possibile sfruttare le relazioni tra i tag ed ottenere un risultato migliore. La figura 4.7 illustra il processo.

Il vantaggio di utilizzare la kernel density estimation e il successivo passo di raffinamento con la random walk è l'assenza di imposizioni sulla struttura dei dati che rende possibile la modellizzazione di qualsiasi tipo di contenuto e quindi adatta a essere utilizzata su Internet. Inoltre significativo utilizzo delle correlazioni tra tag che risultano utili ai fini delle performance. Tuttavia il metodo proposto valuta soltanto la similarità visuale tra coppie di immagini utilizzando features visuali quali colore e tessiture senza considerare la presenza o assenza dei tag come elemento contestuale.

Partendo da questo approccio Wang et al. [WFZY10] propongono un modello di apprendimento semi supervisionato basato sull'apprendimento di proiezioni di valori di ranking dei tag a partire dalla distribuzione delle parole visuali. Feng et al [FLX10] combinano un modello di attenzione visuale a un modello di apprendimento *multi-instance*.

Tecniche non parametriche

Questa categoria di tecniche ha come elemento comune l'utilizzo di tecniche basate sul vicinato di un'immagine: dopo aver estratto una serie di features in uno spazio metrico da tutte le immagini si possono individuare i *nearest neighbours* per inferire nuove informazioni sull'elemento in analisi. Un primo lavoro che utilizza una tecnica di apprendimento *lazy* è di Makadia et al. [MPK10] dove viene introdotto l'utilizzo del vicinato e del trasferimento delle etichette. Da ogni immagine vengono estratte una serie di features basate sul colore (RGB, HSV, LAB) con una semplice divisione spaziale ed una serie basate sulle texture (Haar, HaarQ, Gabor, GaborQ). Una volta combinate secondo un contributo eguale o secondo una penalizzazione lasso l_1, procedono al calcolo delle distanze. Il trasferimento delle etichette è eseguito secondo un semplice algoritmo che individua l'immagine più simile ed emette i suoi tag come predizione. Se sono richiesti un numero maggiore di tag rispetto

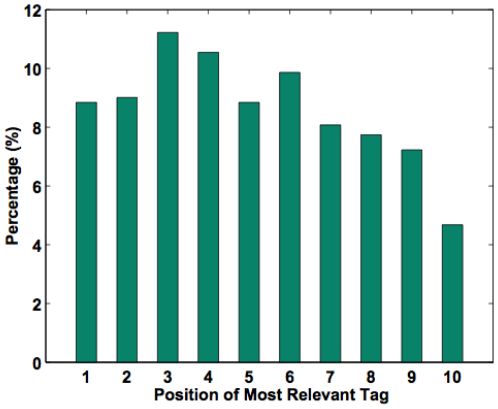

Figura 4.6. Percentuale di immagini che hanno il tag più rilevante nella posizione n-esima della lista con $n = 1, 2, \ldots, 10$.

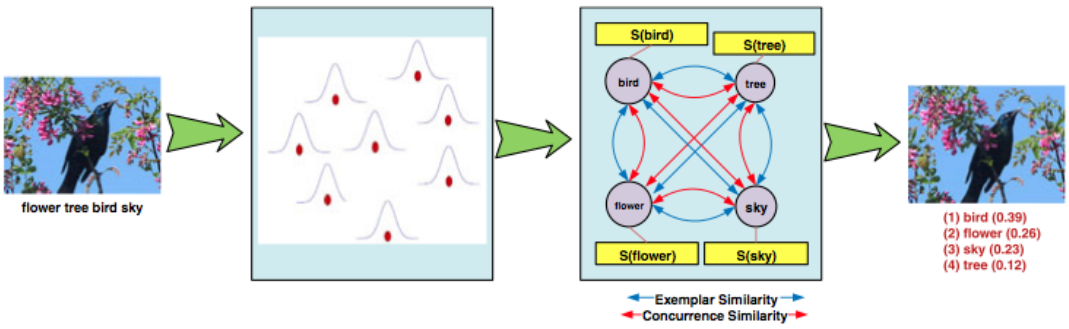

Figura 4.7. Il processo descritto in [LHY+09]: viene eseguita una stima probabilistica mediante kernel density estimation ed infine un processo di random walk per raffinare i risultati.

a quelli presenti nell'immagine più vicina, vengono considerati altri K vicini in ordine di distanza con la relativa distribuzione dei tag. I tag necessari vengono scelti tra quelli con frequenza più alta e maggior correlazione con i tag precedentemente scelti. I risultati riportano un miglioramento delle performance rispetto a tutte le tecniche precedenti, anche molto più complesse rispetto a questo semplice algoritmo.

Uno sviluppo successivo è il lavoro di Guillaumin et al. [GMVS09], dove è fatto uso di un modello probabilistico per selezionare un peso da associare a ogni tipo di feature e a ogni immagine in relazione alle altre. L'approccio permette "l'apprendimento" della metrica da utilizzare per le distanze, evidenziando che l'utilizzo della funzione logistica è in grado di migliorare le performance comprimendo le probabilità dei tag troppo frequenti ed espandendo le probabilità di quelli troppo rari.

Una svolta verso algoritmi più performanti è stata con il lavoro di Li X. et al. [LSW08] dove viene calcolato un punteggio di rilevanza per ogni etichetta, per ogni immagine, accumulando voti da immagini vicine. Viene introdotta una misura di rilevanza denominata *tag relevance*, basata sulla distribuzione dei tag nel vicinato in rapporto alla distribuzione dei tag a livello globale sull'intero dataset. La misura assume che il vicinato sia una distribuzione locale più precisa rispetto a quella globale e stima la probabilità che un tag sia adatto all'immagine in base a quanto la distribuzione locale differisce rispetto a quella globale. Per evitare il bias derivato da utenti con troppe immagini e per cercare di rendere più generale la distribuzione locale, non possono essere scelte più di una immagine per utente. La tecnica in questione rappresenta la base di molti metodi successivi e possiamo considerarla rappresentativa di questo filone di ricerca. Le applicazioni suggerite dagli autori riguardano sia l'image retrieval che il tag refinement per immagini pre etichettate e non.

Successivamente la tecnica è stata espansa in Li X. et al. [LSW10], per lavorare in spazi visuali multipli. Vengono calcolati i punteggi di rilevanza dei tag in differenti spazi di features dopodiché i risultati sono aggregati secondo un metodo di fusione. Sono investigati diversi metodi di aggregazione come per esempio la fusione a punteggio medio, il Borda count e il RankBoost. I risultati mostrano che una semplice fusione media è in grado di avere

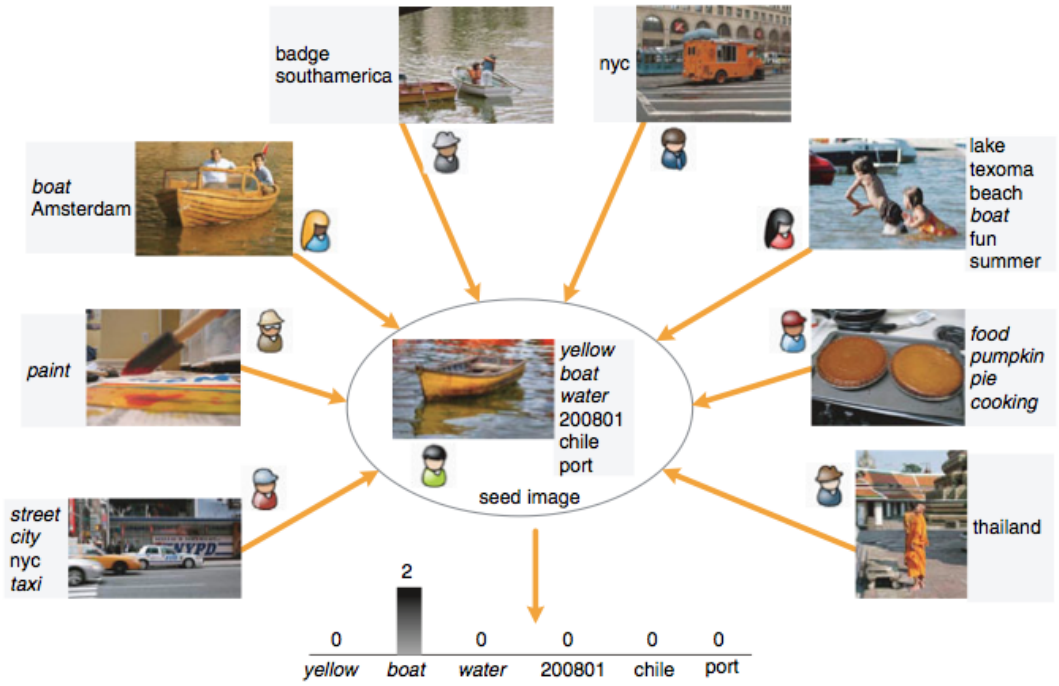

Figura 4.8. La tecnica della misura *tag relevance*. Ogni immagine del vicinato vota i tag dell'immagine di test sulla base dei propri.

performance paragonabili a metodi supervisionati come RankBoost.

La tecnica di Li X. et al. [LSW08] è stata applicata con successo ai video nel lavoro di Ballan et al. [BBDB+10], dove si cerca di applicare tag a parti di sequenze di video. Dato un filmato taggato a livello di video, l'algoritmo utilizza i suoi tag per eseguire ricerche su Flickr e individuare immagini attinenti da utilizzare successivamente nella predizione dei tag a livello di shot. Vengono estratte una serie di features da ogni keyframe per individuare tra le immagini di Flickr scaricate i vicini visuali, su cui viene applicata la misura tag relevance per la predizione.

4.2.2 Tag Refinement

Dei problemi descritti, quello del tag refinement, è sicuramente il più studiato negli ultimi anni. L'obiettivo del tag refinement è quello di migliorare i tag già esistenti delle immagini, per produrne una serie che non sia affetta dai tipici problemi di soggettività ed incompletezza di quelli prodotti dagli utenti. Il processo generalmente sfrutta sia i tag esistenti, sia una serie di informazioni e di conoscenza di fonti eterogenee come ad esempio il vocabolario della lingua, la similarità semantica tra tag, la similarità tra le immagini, le relazioni in essere e così via. Un qualsiasi tag presente in Flickr ha soltanto il 50% di probabilità che questo sia rilevante [SvZ08] e più della metà delle immagini ha soltanto 3 o meno tag. Generalmente non sono sufficienti per descrivere completamente il contenuto di un'immagine. Sono state sviluppate diverse tecniche per cercare di migliorare i tag e generalmente si basano sulle seguenti assunzioni:

- La similarità visuale tra le immagini dovrebbe riflettersi nella similarità tra gli insiemi di tag annessi.
- Tag di significato semantico simile dovrebbero avere un'alta correlazione quindi un'alta probabilità di apparire insieme.
- I tag inseriti dagli utenti non dovrebbero essere cambiati troppo, questo per cercare di regolarizzare il processo.

Uno degli sforzi iniziali è di Liu et al. [LHWZ10] dove propone un sistema generale per migliorare la qualità dei tag. L'assunzione utilizzata è che deve essere imposta la consistenza tra similarità visuale e similarità semantica. L'organizzazione del processo è in tre fasi successive:

1. **Filtraggio:** i tag vengono sottoposti ad un filtraggio utilizza una qualche ontologia o vocabolario per escludere i tag di cui non è conosciuto il significato semantico. Nel lavoro in questione viene utilizzata WordNet [Fel98] e quindi limitando lo studio alla lingua inglese.

2. **Rifinitura:** basandosi sull'assunzione descritta in precedenza, il problema è formulato come un framework che cerca di massimizzare la consistenza mentre minimizza la differenza dai tag forniti dagli utenti. Questo permette l'inserimento di informazioni da canali differenti in modo collettivo. L'assunzione di consistenza è applicabile principalmente ai tag di contenuti relativi ovvero quelli che hanno un'alta probabilità di descrivere il contenuto visuale delle immagini. I tag con contenuti non relativi peggiorano le performance dell'algoritmo, ma dovrebbero essere diminuiti dall'utilizzo del precedente passo di filtraggio.

3. **Arricchimento:** una volta che le etichette individuate sono più 'sicure', è possibile tentare di arricchirne il valore aggiungendo elementi quali sinonimi e peronimi sulla base di ontologie come ad esempio WordNet. L'arricchimento, pur favorendo la recall, di per sé rappresenta una possibile fonte di informazioni scorrelate e potenzialmente errate.

I risultati mostrano l'efficacia del procedimento e il possibile miglioramento delle performance di image retrieval ed annotazione di nuove immagini. Il problema principale di questa tecnica è che funziona soltanto su un'insieme di immagini e tag fissati, senza una strategia per gestire le immagini fuori dal campione analizzato. Questo limita ovviamente l'applicazione al web in quanto le comunità sociali sono prettamente dinamiche e si evolvono in tempo reale.

Basandosi su questo lavoro, Zhu et al. [ZYM10] propone un nuovo metodo di raffinamento dei tag basandosi su quattro osservazioni effettuate su un grande volume di immagini acquisite dai social network:

1. Lo spazio semantico delle informazioni testuali è generalmente approssimabile utilizzando un piccolo sottoinsieme di parole salienti derivate dallo spazio originale. Essendo pur sempre informazioni testuali, anche i tag delle immagini sono soggetti alle stesse proprietà come ad esempio la possibilità di applicare l'approssimazione low-rank.

2. Immagini con contenuti visuali simili, sono tipicamente annotati con tag simili. Possiamo indicare questo fatto come proprietà di consistenza del contenuto.

3. I tag non appaiono isolati gli uni dagli altri, ma generalmente in gruppo, in base alle correlazioni e alle interazioni nel livello semantico.

4. I tag risultati per ogni immagine sono accurati fino ad un certo punto e questo porta ad avere sparsità nella matrice errore dei tag ed immagini.

Sulla base di queste osservazioni, Zhu et al. utilizza la norma traccia e la norma l_1 per modellare una decomposizione della matrice dei tag forniti dagli utenti in matrici di basso rango. In particolare la matrice degli utenti è ricostruita come somma di una matrice a basso rango ed una matrice di errori. Viene formulato un problema di ottimizzazione convessa che simultaneamente minimizza il rango della matrice, la consistenza e la sparsità nella matrice degli errori. Una procedura iterativa procede alla risoluzione del problema. Le performance sono migliori rispetto al metodo in [LHWZ10], tuttavia il problema dell'applicazioni su immagini fuori dal campione scelto rimane. Inoltre l'ottimizzazione della forma nucleare è eseguita utilizzando la decomposizione SVD (singular value decomposition) che ne limita la scalabilità nel problemi a grande scala.

In Bracamonte et al. [BP11] viene proposto un metodo per la propagazione delle informazioni al fine di eseguire il tagging automatico di immagini. Il sistema riutilizza i grafi visuali semantici [PBMB10] per rappresentare le

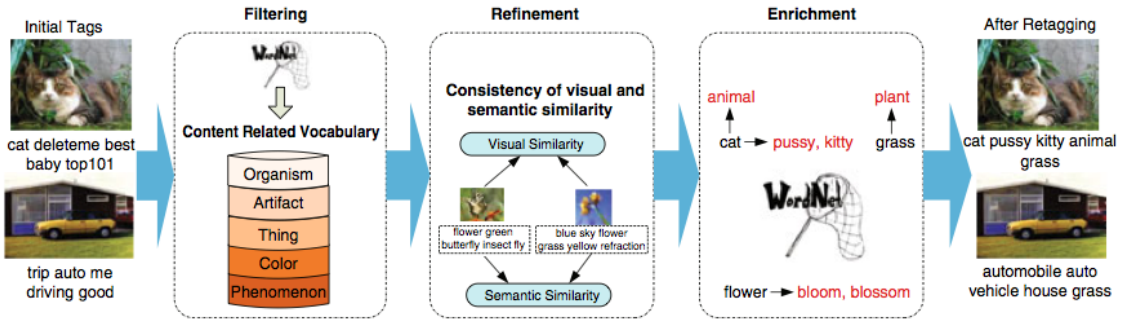

Figura 4.9. Schema del processo di raffinamento dei tag descritto in [LHWZ10]. I tag vengono filtrati, rifiniti ed infine arricchiti da nuove fonti esterne ai soli tag degli utenti.

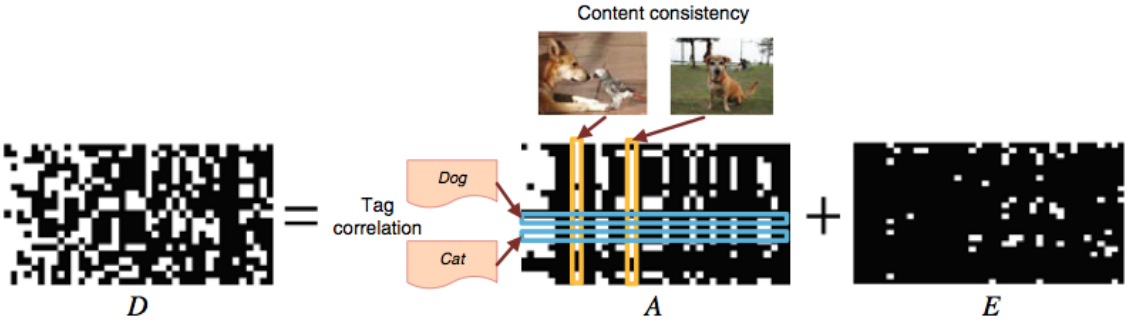

Figura 4.10. La matrice dei tag originali è decomposta in una somma di due matrici: una matrice low-rank ed una relativa agli errori di ricostruzione.

relazioni tra la semantica e la similarità visuale nelle preferenze di ricerca degli utenti. È quindi necessario avere a disposizione i click eseguiti sulle query di motori di ricerca e le relative immagini. Viene evidenziata l'esistenza di cosiddette 'stop-images' che sono paragonabili alle stop-words utilizzate nei sistemi di indicizzazione testuali: ovvero si tratta di immagini che possiedono innumerevoli collegamenti e che dominano il flusso di trasferimento di informazioni nei grafi. La rimozione di questi 'super-nodi' migliora le prestazioni. Il contributo chiave del lavoro è un procedimento innovativo per il trasferimento delle etichette e una serie di euristiche per il pruning dei grafi su insiemi di immagini a larga scala.

Un approccio diverso, adatto ad essere utilizzato su dati a larga scala, è quello di Tsai et al. [TJL+11]. Si tratta di uno dei primi lavori su milioni di immagini e migliaia di annotazioni. Gli autori descrivono il concetto di *'visual-synset'* prendendo spunto dai synset semantici, come ad esempio quelli impiegati in WordNet. Un visual-synset è un insieme di immagini che sono visivamente simili o semanticamente correlate. Ogni visual-synset rappresenta un prototipo di concetto visuale con associate una serie di annotazioni pesate e vengono definiti collezionando immagini e tag in modo completamente automatico attraverso query su motori di ricerca di immagini (come Google Image Search). Per ogni visual synset così reperito, è addestrata una SVM lineare, in grado di definire l'inclusione o meno di nuove immagini nel synset. La predizione avviene mediante votazione di ogni synset, sommando i pesi sui tag associati dei synset risultati positivi. È inoltre ribadita l'efficacia dell'utilizzo della relazione tra similarità visuale e similarità semantica, dimostrando il potere discriminativo dei visual synset.

Infine la direzione più recente è costituita dall'applicazione di tecniche di raccomandazione e di filtraggio collaborativo, dagli autori Sang et al. in [SXL12] evoluzione del lavoro di Rendle et al. in [RMNSt09], applicato ad oggetti generici. Sempre sulla scia dell'inclusione di informazioni aggiuntive, Sang et al. considerano l'aggiunta di maggior informazioni relazionali. Invece di utilizzare soltanto le informazioni tra immagini e tag, il modello considerato analizza una semantica ternaria includendo gli utenti. Il problema è ricondotto alla fattorizzazione tensoriale a tre vie, utilizzando la decomposi-

zione di Tucker [AY09] e imponendo vincoli soft sulla similarità dei fattori latenti generati. Distinguendo una natura ternaria dei tag come positivi sicuri, negativi mancanti od incerti e negativi sicuri, il modello impone per ogni immagine una lista di rank basata sulle informazioni sicure. I risultati condotti indicano performance promettenti, simili allo stato dell'arte, beneficiando di una performance superiore per i tag che possono trarre beneficio dal background degli utenti.

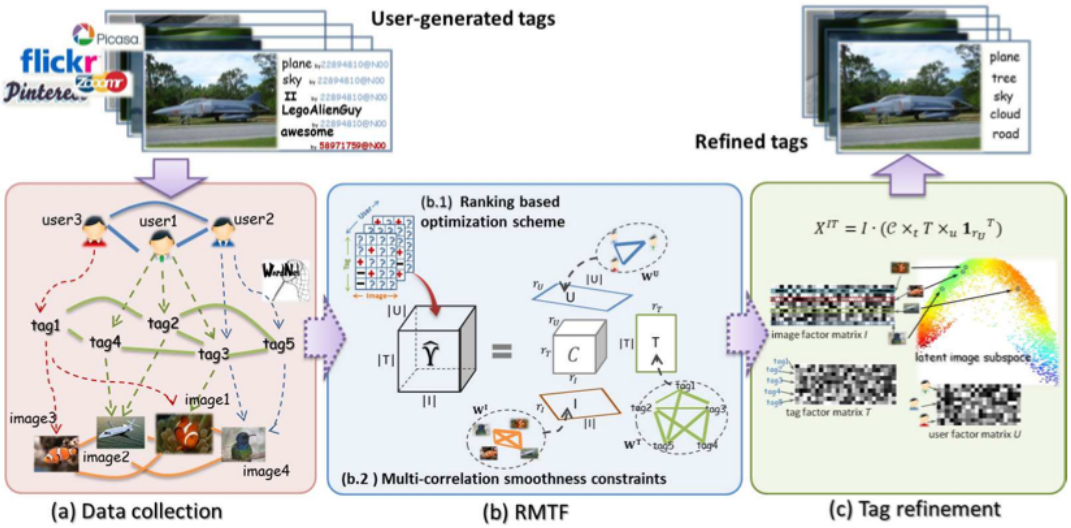

Figura 4.11. Il framework di lavoro proposto in [SXL12].

5

Contributo del lavoro di tesi

No problem is insurmountable. With a little courage, teamwork and determination a person can overcome anything.

– B. Dodge

Ci concentriamo adesso soltanto sul problema del tag refinement, cercando di analizzare e possibilmente migliorare lo stato dell'arte. Dato quindi un corpus Φ con un insieme di immagini I, un insieme di utenti U, un vocabolario per i tag V_T ed un vocabolario di etichette V_L, vogliamo individuare una funzione che assegna le migliori N etichette di V_L ad ogni immagine.

Osservando gli ultimi principali lavori vediamo che è difficile compararli tra di loro, in particolare è difficile poter affermare che un lavoro è superiore agli altri. La letteratura indica con certezza che una semplice analisi della densità di probabilità dei tag non è sufficiente a ottenere buoni risultati. I tag rari non emergono e c'è bisogno di un ulteriore rafforzamento (ad esempio con random walk [LHWZ10]). L'approccio non parametrico con nearest neighbours risulta pratico e al momento costituisce uno degli approcci migliori. È possibile applicarlo a dizionari scalabili e, risolto il problema di calcolare le distanze, è un classificatore lazy (non ha bisogno di addestramento). Dall'altra parte, invece, gli approcci parametrici dei lavori di Zhu et al. [ZYM10] e di Tsai et al. [TJL+11] hanno la possibilità di studiare il modello appreso per eseguire *data mining*, ossia estrarre informazioni rilevanti sugli utenti. Il costo è concentrato nella ricerca dei parametri del modello: una volta ap-

preso, il costo della predizione è minore rispetto alle tecniche con nearest neighbours.

Purtroppo lavorando in un ambito di ricerca dove la semantica del linguaggio è importante, è difficile anche stabilire quale tecnica sia superiore in quanto è difficile quantificare i risultati. Sicuramente una buona parte della valutazione è incentrata sulla qualità che sulla quantità ed entra in gioco il fattore soggettivo dell'utente. Questo ci porta ad un problema nella valutazione.

5.1 Il problema della valutazione

Un problema difficile quando si lavora con i tag è sicuramente quello della valutazione. In letteratura non è stato definito un metodo di comune accordo per valutare le performance di un sistema per il raffinamento di tag. Ci sono una serie di valutazioni che rendono difficile trovare una soluzione buona per ogni caso, rendendo il problema mal definito:

- **Il numero di tag**: il numero dei tag da assegnare a un'immagine non sono definiti. Non è possibile esprimere quantitativamente il numero di tag necessari per descrivere un'immagine. I risultati precedenti suggeriscono che sono necessari in media circa 3-5 tag per immagine. Tuttavia, essendo una media, ci sono sicuramente immagini che hanno bisogno di più o meno di 3-5 tag. La scelta del numero di tag da assegnare è ancora un problema aperto e probabilmente di difficile soluzione senza interpretare completamente la semantica dell'immagine.

- **Vocabolario:** quali tag utilizzare? Un sistema automatico si può limitare nell'uso di termini di oggetti presenti nell'immagine, ma per altri utenti potrebbero non essere i migliori da utilizzare. La semantica dei termini è diversa da utente ad utente: gli stessi termini potrebbero essere interpretati in modi diversi oppure potrebbero essere troppo generici per alcuni o troppo specifici per altri.

- **Ground Truth:** la costruzione della ground truth è difficoltosa e mal definita. Prima di tutto per poter costruire una ground truth è necessario scegliere i termini da utilizzare ovvero l'ontologia dei termini. Una volta definiti i termini, è opportuno valutare la semantica. Da qui i problemi: nessuna ontologia potrà mai descrivere tutti i possibili termini in tutti i loro possibili significati. Oltre a considerare tutte le possibili aree specialistiche, ognuno di noi potrebbe attribuire ad un termine un significato diverso e al momento non esiste un modo oggettivo per fare il match dei termini con esattezza. Assumendo comunque di riuscire a realizzare una buona ontologia, c'è anche da spendere tempo per realizzare l'etichettatura: su una quantità di immagini come quella dei dataset MIRFLICKR e NUSWIDE realizzarla è un costo molto alto.

Purtroppo la valutazione completa di un sistema che lavora su un numero di immagini alto, come nel nostro caso, pone difficoltà nel confronto tra le tecniche. I dataset descritti in precedenza possiedono una ground truth parziale: soltanto pochi concetti sono rappresentati nel vocabolario delle etichette. La valutazione verrà fatta soltanto su questi e ovviamente non è completamente rappresentativa del caso generale. Purtroppo, come visto in [DBLFF10], l'aumentare del numero delle etichette considerate potrebbe cambiare le performance di riferimento. Per le misurazioni, aggiungiamo quindi un vincolo sul vocabolario di output del sistema, limitandolo al vocabolario della groundtruth.

Un altro problema che rende difficoltosa la comparazione del nostro sistema con i precedenti è che generalmente non è specificato esattamente il pre processing applicato ai tag. Ogni lavoro utilizza un proprio sistema e non sempre è specificato completamente lasciando sottinteso parti che sono delicate ai fini della comparazione. Ciò è visibile ad esempio nella valutazione dell'F_1 sui tag originali: nessun lavoro precedente ha in comune i valori con altri.

In letteratura, i recenti lavori che affrontano il tag refinement [LHWZ10] [ZYM10] [TJL+11], utilizzano la metrica F_1 analizzando sia la performance a livello di etichetta che a livello globale. Date M categorie, la precision per

categoria i è definita come:

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \tag{5.1}$$

e la recall per categoria come:

$$\rho_i = \frac{TP_i}{TP_i + FN_i} \tag{5.2}$$

dove TP_i sono i *true positive* (le immagini correttamente assegnate alla categoria), FP_i sono i *false positive* (le immagini erroneamente assegnate alla categoria) e FN_i sono i *false negative* (le immagini della categoria erroneamente non assegnate).

La metrica F_1 è definita in $[0, 1]$ e valori maggiori corrispondono a una classificazione di qualità maggiore. Per ogni concetto è definita come:

$$F_{1,i} = \frac{2\pi_i\rho_i}{\pi_i + \rho_i} \tag{5.3}$$

Esistono anche misure F_β con valori di $\beta \neq 1$. La formula generale è:

$$F_\beta = (1 + \beta^2)\frac{\pi_i}{(\beta^2\pi_i) + \rho_i} \tag{5.4}$$

Le misure più popolari oltre la F_1 sono la $F_{0,5}$ che dà più peso alla precision e la F_2 che dà più peso alla recall.

Per quanto riguarda invece la misura F_1 relativa all'intero problema di classificazione esistono due tipi di media: la *micro-average* e la *macro-average* [OOG05].

La misura **micro-average** calcola l'F_1 sull'intero corpus delle immagini su tutte le categorie. La precision e la recall sono ottenute sommando i contributi delle singole immagini:

$$\pi = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M}(TP_i + FP_i)} \tag{5.5}$$

$$\rho = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M}(TP_i + FN_i)} \tag{5.6}$$

dove M è il numero delle categorie. L'F_1 micro-average è quindi ottenuta calcolando:

$$F_{1,\text{micro}} = \frac{2\pi\rho}{\pi + \rho} \tag{5.7}$$

La micro-average considera ogni immagine con lo stesso peso ed è quindi considerata una media su tutte le coppie documento-categoria. Tende a essere dominata dalle performance dei classificatori sulle poche categorie frequenti.

La misura **macro-average**, invece, calcola l'F_1 localmente per ogni categoria ed infine ne esegue una media. La precision e la recall vengono calcolate per ogni categoria utilizzando rispettivamente le equazioni 5.1 e 5.2. Lo stesso la F_1 per ogni categoria utilizzando la 5.3. A questo punto l'F_1 globale dell'intero sistema è la media dell'F_1 di ogni categoria:

$$F_{1,\text{macro}} = \frac{\sum_{i=1}^{M} F_{1,i}}{M} \tag{5.8}$$

La macro-average considera ogni categoria con lo stesso peso, indipendentemente dalla loro frequenza. Tende ad essere dominata dalla performance sulle categorie rare. Decidiamo di utilizzare la micro-average per le valutazioni principali in quanto sono presenti un numero abbondante di categorie poco frequenti.

Il numero dei tag predetto dai metodi analizzati è generalmente 5 o 10 per immagine. Pensiamo che sia interessante valutare anche un numero inferiore, dove si dovrebbe evidenziare l'insufficienza a descrivere l'intero contenuto visuale. Inoltre è desiderabile riuscire ad avere un algoritmo che predice in meno tag corretti possibili il contenuto.

Infine il vocabolario dei concetti della ground truth del dataset diventa il vocabolario delle etichette di output del sistema V_L. In questo modo il modello in analisi può solo predire tag valutabili.

5.2 Scelta delle tecniche

Vista la difficoltà di dimostrare e verificare in modo quantitativo la superiorità di un approccio rispetto ad un altro e vista la possibilità di utilizzare diversi modi di valutazione (non sempre chiari nei lavori di riferimento), decidiamo di comparare tre tecniche rappresentative. Utilizziamo principalmente i tre dataset descritti in precedenza: ESP Game, MIRFLICKR, NUSWIDE.

Dall'osservazione dei tre dataset notiamo alcune particolarità:

- **ESP Game** possiede immagini totalmente casuali recuperate su pagine internet casuali. Si può trovare di tutto, da banner pubblicitari a bordi utilizzati nel disegno delle pagine web; da piccolissime immagini con clipart a immagini di dimensioni discrete con panorami. Generalmente le immagini non sono di buona qualità, sono piccole ed alcune hanno anche palette di soli 8 colori. I concetti descritti sono facilmente interpretabili da un umano. La ground truth è stata realizzata con l'ausilio di un programma per computer e potrebbe contenere errori. Infine non ci sono i tag degli utenti, ma solo le annotazioni.

- **MIRFLICKR** possiede immagini derivanti dalle *interestingness* di Flickr. Tutte le immagini del dataset sono di alta qualità, ma possiamo arguire che non si trattano di foto semplici. Le foto interesting sono una nicchia di tutte le possibili foto presenti sul sito: Flickr è composto da utenti di diverse tipologie, ma non tutti producono foto interessanti. Alcune persone postano foto scattate con macchine fotografiche compatte economiche, altre con reflex, altre fanno fotoritocchi e così via. Le foto interessanti seguono un po' la qualità preferita dal periodo e dalle persone che ci sono ed è naturale avere una selezione di foto particolari: a nessuno piacciono le foto "piatte". Una foto deve esprimere creatività, elementi artistici o sentimenti. Sono tutte caratteristiche difficili da interpretare e che non sono comprese nei dati visuali in sé perché sono sensazioni che cambiano da persona a persona. Si potrebbe obiettare che un computer deve essere in grado di capirle, ma per gli obiettivi scelti in precedenza (etichettare secondo gli elementi compresi nell'immagine) sono immagini più difficili. Un altro fattore interessante sono il numero di tag che generalmente è più alto rispetto alla media generica di Flickr e più variegato.

- **NUSWIDE** invece è stato realizzato utilizzando parole chiavi ricercate su Flickr. Generalmente le immagini sembrano ben categorizzabili in quanto ci sono molti elementi ripetuti in tante immagini: le immagini del mare hanno molte similarità, ci sono diverse istanze dei soliti animali e così via. A parte questa considerazione, le immagini sembra-

no ben variegate in termini di stile: si va da foto tipiche "snapshot" di situazioni, fino ad immagini creative e ritoccate. I tag sembrano ricalcare la normale distribuzione descritta in precedenza.

Quindi NUSWIDE è più rappresentativo della realtà dei media sociali rispetto a MIRFLICKR. Sia NUSWIDE che MIRFLICKR costituiscono comunque un insieme di foto che è possibile trovare sui social network, a differenza di ESP Game che ne contiene un campione poco rappresentativo.

Per quanto riguarda le tecniche seguiamo alcune considerazioni frutto del lavoro di studio della letteratura:

- Assumiamo che la vicinanza visuale implica vicinanza semantica e viceversa.
- Ogni utente inserisce tag solo sulle proprie foto. Su Flickr questa assunzione è generalmente vera: gli utenti inseriscono tag sulle altre immagini solo per segnalarle come appartenenti ad un gruppo specifico. In genere il tag inserito è univoco e non è una parola di uso comune.
- L'apporto informativo dell'utente che inserisce le immagini è utile.
- La probabilità che un tag inserito da un utente sia corretto è maggiore della probabilità che non lo sia.

Osservati quindi i dataset in gioco, decidiamo di prendere una tecnica rappresentativa per le tecniche parametriche e una per le non parametriche:

- Per le tecniche non parametriche scegliamo l'approccio di Li X. et al. [LSW10], la tecnica denominata **TagRelevance** applicata al tag refinement. Inoltre valutiamo anche la performance di una tecnica molto simile di Makadia et al. [MPK10] che denominiamo **TagSimilar**. Entrambe le tecniche si basano sulle assunzioni elencate (a parte Makadia et al. che non utilizza gli utenti).
- Per le tecniche parametriche scegliamo il lavoro più recente che modella le informazioni ternarie tag, immagini e utenti di Sang et al. [SXL12].

In letteratura, a nostra conoscenza, nessuno ha mai confrontato prima i due approcci. Descriviamo gli approcci che utilizziamo.

5.2.1 TagSimilar

Data un'immagine I_T, e un numero intero n di etichette che vogliamo assegnare a I_T, si utilizza un approccio basato sulle immagini vicine visualmente. Per calcolare la distanza tra due immagini si utilizza una metrica basata su K tipi di feature diverse. Ai fini del confronto delle tecniche le feature estratte devono essere le stesse, in modo da evidenziare i diversi risultati a parità di informazioni di input. Trattiamo successivamente la scelta delle features.

Data l'immagine di input, utilizzando la metrica, possiamo individuare un insieme di R immagini in ordine di distanza crescente:

$$I_{T,1}, I_{T,2}, \ldots, I_{T,R} \tag{5.9}$$

Dove l'immagine $I_{T,1}$ è la più simile visualmente. Denotiamo con $|I_{T_i}|$ il numero di tag associati all'immagine vicina I_{T_i}. L'algoritmo di assegnazione delle immagini è così definito:

1. Valuta la rilevanza di ogni tag di $I_{T,1}$, l'immagine più simile, utilizzando la frequenza del training set.

2. Trasferisci n etichette di $I_{T,1}$ a I_T in ordine di rilevanza a partire da quelle con il valore più alto. Se $|I_{T,1}| \geq n$ l'algoritmo termina, altrimenti prosegue con il passo successivo.

3. Considera le successive $I_{T,2}, I_{T,3}, \ldots, I_{T,R}$ immagini e i loro tag. Valutiamo tutti i tag usati nell'insieme (quindi il vocabolario) in base alla co-occorrenza delle parole trasferite al passo 2 con il training set e alla frequenza locale del gruppo delle immagini vicine. Si trasferiscono le $n - |I_{T,1}|$ etichette con il maggior punteggio di rilevanza così ottenuto.

L'algoritmo, per quanto semplice, è generalmente diverso dalle scelte più ovvie, tipo la selezione simultanea di immagini dall'intero vicinato. Trattare le immagini allo stesso modo, secondo quanto riportato in [MPK10], non è una buona scelta.

5.2.2 TagRelevance

Anche TagRelevance, similmente a TagSimilar, utilizza un approccio basato sulle immagini vicine visualmente. Data un'immagine I_T e un numero intero di n di etichette che vogliamo assegnare a I_T, utilizziamo una metrica basata su K tipi di feature diverse. La funzionalità dell'algoritmo di TagRelevance si basa su due assunzioni considerate in precedenza:

1. In un database a larga scala di immagini, la probabilità che un tag inserito dall'utente sia corretto è maggiore della probabilità che non lo sia. Le etichette inserite dagli utenti non sono casuali la maggior parte delle volte (almeno per gli oggetti visibili).

2. Una ricerca basata sul contenuto utilizzando feature visuali è migliore rispetto ad un campionamento casuale.

La misura di rilevanza TagRelevance è definita come:

$$\text{tagRelevance}(t, I, k) := n_t[N_f(I, k)] - Prior(t, k) \tag{5.10}$$

dove t è il tag di cui vogliamo la rilevanza per l'immagine I, k il numero di vicini da considerare; $n_t[N_f(I, k)]$ è il numero di volte che il tag t appare nel vicinato di I con k immagini, ovvero la probabilità di trovare il tag considerato nel vicinato. Il termine $Prior(t, k)$ invece è la probabilità di trovare il tag su una qualsiasi immagine del corpus considerato. Può essere approssimato misurando la probabilità frequentista dei tag sull'intero corpus:

$$Prior(t, k) \approx k\frac{|L^T|}{|\Phi|} \tag{5.11}$$

La misura di rilevanza misura la differenza tra la probabilità di individuare il tag nel vicinato rispetto alla probabilità di trovarlo in una qualsiasi immagine del corpus. Questo meccanismo è basato esplicitamente sulla seconda assunzione.

Sulla base della definizione è introdotto il seguente algoritmo:

1. Data l'immagine I_T di test, si individuano K vicini visuali con il vincolo che ogni utente può votare con un'immagine soltanto (la più vicina). Ogni utente ha soltanto un'immagine nel vicinato.

2. Definiamo ed inizializziamo il punteggio tagRelevance per ogni tag t da considerare a 0, ovvero poniamo $\text{tagRelevance}(t, I, k) = 0$.

3. Per ogni immagine J del vicinato di I_T, per ogni tag $t \in J$

   (a) $\text{tagRelevance}(t, I, k) = \text{tagRelevance}(t, I, k) + 1$

4. Dopo aver iterato su tutti i tag, si sottrae il prior secondo la definizione: $\text{tagRelevance}(t, I, k) = \text{tagRelevance}(t, I, k) - Prior(t, k)$

5. Si pone $\text{tagRelevance}(t, I, k) = \max(\text{tagRelevance}(t, I, k), 1)$

L'informazione dell'utente viene utilizzata in fase di selezione del vicinato in modo da evitare l'imparzialità di giudizio relativa agli utenti. Accade spesso che un utente inserisca un insieme di immagini molto simili visivamente di un evento (ad esempio una partita di basket in una palestra) e che costituiscano un cluster denso. Le immagini sono generalmente sparse relativamente alla distanza tra di loro e quindi porterebbero in caso di selezione di un'immagine del cluster a selezionare anche le altre in quanto vicinissime. Si tratta di una deviazione di giudizio rispetto a un singolo utente che viene evitato imponendo il vincolo della singola immagine per utente. Si suppone che l'insieme di immagini sia abbastanza ampio da colmare la riduzione delle immagini simili.

La misura di tagRelevance dà un valore alla rilevanza di un tag rispetto a un'immagine, rimane quindi da applicarlo al problema del tag ranking. Un modo semplice è quello suggerito dagli autori dell'articolo originale: ipotizzando che l'immagine di test non abbia etichette, si può valutare la rilevanza di tutte le etichette del dizionario V_L e procedere a selezionare le prime n etichette che hanno il valore più alto. Ci riferiremo a questa modalità con il termine generico **TagRelevance**.

Supponendo che i tag filtrati siano corretti, possiamo pensare di utilizzare la misura tagRelevance per aggiungere tag, lasciando i tag esistenti. Data un'immagine di test I_T con i suoi tag filtrati $T_T = \{t_1, t_2, \ldots, t_M\} \in V_L$ associati dagli utenti, valutiamo la rilevanza sull'insieme $V_L \setminus T_T$, cioè tutte le etichette del vocabolario eccetto i tag già utilizzati. In questa modalità di

funzionamento i tag originali vengono conservati e viene soltanto aggiunta informazione. L'approccio si basa sull'idea che la maggior parte dei tag sono mancanti e infatti sono affetti da bassa recall. Denominiamo questa modalità **TagRelevancePlus**.

5.2.3 RankingDecomposition

La relazione tag(u, i, t) descritta nella sezione 1.3 può essere rappresentata sotto forma di tensore a 3 modi. Un modo è dedicato ad enumerare le immagini, uno ad enumerare gli utenti e l'ultimo ad enumerare i tag. Costruiamo il tensore $\underline{\mathbf{Y}} \in \mathbb{R}^{|U|\times|I|\times|T|}$ per la relazione tag, definendo:

$$y_{u,i,t} = \begin{cases} 1 & \text{se } \exists \text{ tag}(u,i,t) \\ 0 & \text{altrimenti} \end{cases} \tag{5.12}$$

L'idea è di applicare la decomposizione Tucker3 per cercare i fattori latenti alla base dei tre modi. Vogliamo individuare un segnale per ogni immagine, uno per ogni utente e uno per ogni tag sempre facendo corrispondere i dati originali. Si suppone che due utenti simili abbiano in comune i fattori latenti o almeno poco distanti in norma tra di loro. Lo stesso per le immagini e i tag. Una volta scelte in qualche modo il numero di componenti latenti per ogni modo $r_u \in \mathbb{N}$ per gli utenti, $r_i \in \mathbb{N}$ per le immagini, $r_t \in \mathbb{N}$ per i tag, la decomposizione di Tucker3 è definita:

$$\underline{\mathbf{Y}} = \underline{\hat{\mathbf{C}}} \times_u \hat{\mathbf{U}} \times_i \hat{\mathbf{I}} \times_t \hat{\mathbf{T}} + \underline{\mathbf{E}} \tag{5.13}$$

con $\underline{\hat{\mathbf{C}}} \in \mathbb{R}^{r_u \times r_i \times r_t}$ il tensore nucleo, $\hat{U} \in \mathbb{R}^{r_u \times |U|}$ la matrice dei fattori latenti degli utenti, $\hat{I} \in \mathbb{R}^{r_i \times |I|}$ la matrice dei fattori latenti delle immagini, $\hat{T} \in \mathbb{R}^{r_t \times |T|}$ la matrice dei fattori latenti dei tag ed infine $\underline{\mathbf{E}} \in \mathbb{R}^{|U|\times|I|\times|T|}$ il tensore dell'errore di ricostruzione. Possiamo anche indicare la relazione elemento per elemento:

$$y_{u,i,t} = \sum_{\tilde{u}} \sum_{\tilde{i}} \sum_{\tilde{t}} c_{\tilde{u},\tilde{i},\tilde{t}} \cdot u_{u,\tilde{u}} \cdot i_{i,\tilde{i}} \cdot t_{t,\tilde{t}} \tag{5.14}$$

Usiamo la notazione con la tilde per riferirsi alle dimensioni dei fattori latenti. Ad esempio la notazione $i_{i,\tilde{i}}$ indica la riga i che appartiene all'immagine I_i e $\tilde{i} \in [1, r_i]$ indica la componente latente, cioè la colonna.

Il tensore nucleo contiene le interazioni tra i tre modi mentre le matrici di modo a basso rango rispondono a un solo modo. Definiamo per comodità $\hat{\theta} = \{\hat{C}, \hat{U}, \hat{I}, \hat{T}\}$. Le 4 componenti sono derivate dal tensore originale $\underline{\mathbf{Y}}$ della relazione tag, imponendo un problema di minimizzazione:

$$\underset{\hat{\theta}}{\operatorname{argmin}} \sum_{(u,i,t)\in U\times I\times T} (\hat{y}_{u,i,t} - y_{u,i,t})^2 \tag{5.15}$$

Assumendo che i dati siano densi, è conveniente utilizzare la distanza euclidea. Definiamo questo tipo di ottimizzazione con il nome **schema 0/1**, ovvero si cerca di imparare direttamente i valori 0 e 1 del tensore originale.

Tuttavia l'assunzione che i dati siano densi non è corretta per questo problema: la relazione tag è sparsa e si riflette nel tensore. Inoltre la decomposizione è eseguita con matrici e tensori reali quando invece si cerca di ricostruire due soli valori interi 0 e 1. Risulta più conveniente valutare direttamente la lista di ranking finale e di imporre l'ordinamento utilizzando il training set. Data la lista finale di ranking dei tag per un'immagine, l'idea è che tutti i tag di un'immagine devono apparire prima dei tag assenti. Conviene quindi cambiare l'obiettivo in:

$$\underset{\hat{\theta}}{\operatorname{argmin}} \sum_{(u,i)\in U\times I} \frac{1}{|T^+_{u,i}||T^-_{u,i}|} \sum_{t^+\in T^+_{u,i}} \sum_{t^-\in T^-_{u,i}} H_{0.5}(\hat{y}_{u,i,t^-} - \hat{y}_{u,i,t^+}) \tag{5.16}$$

dove $t^+ \in T^+$ sono gli indici dei tag indicati come positivi cioè presenti, $t^- \in T^+$ sono gli indici dei tag indicati come negativi cioè assenti. H_α è la funzione gradino di Heaviside:

$$H_\alpha(x) = \begin{cases} 0 & \text{se } x < 0 \\ \alpha & \text{se } x = 0 \\ 1 & \text{se } x > 0 \end{cases} \tag{5.17}$$

In questo modo l'addestramento è fatto utilizzando soltanto i tag osservati e si ottimizza direttamente per l'obiettivo finale cioè la lista di ranking. Denominiamo questo approccio con **schema ranking**. In questo modo stiamo affermando che per un'immagine, se i tag positivi non stanno sopra ai tag negativi nella lista finale, paghiamo un costo pari ad 1.

Dato un numero sufficiente di componenti, è possibile ottenere una buona ricostruzione delle liste di ranking del training set. Tuttavia non siamo interessati ad ottenere buone performance rispetto al training in quanto vogliamo generalizzare su nuove immagini e tag ovvero si potrebbe verificare il problema dell'*overfit*. È opportuno aggiungere termini di regolarizzazione all'obiettivo:

$$\begin{aligned}\underset{\hat{\theta}}{\operatorname{argmin}} \sum_{(u,i)\in U\times I} \frac{1}{|T^+_{u,i}||T^-_{u,i}|} \sum_{t^+\in T^+_{u,i}} \sum_{t^-\in T^-_{u,i}} H_{0.5}(\hat{y}_{u,i,t^-} - \hat{y}_{u,i,t^+}) \\ + \gamma_C||\hat{C}||^2_F + \gamma(||\hat{U}||^2_F + ||\hat{I}||^2_F + ||\hat{T}||^2_F)\end{aligned} \tag{5.18}$$

dove $\gamma_C, \gamma \in \mathbb{R}$ sono i parametri per la regolarizzazione e $||\cdot||_F$ è la norma di Frobenius.

Il problema dei tag mancanti

Rimangono da definire gli insiemi dei tag positivi e dei tag negativi per ogni immagine. Si può utilizzare uno **schema binario** dove i tag presenti sono i positivi ed i tag assenti sono i negativi. Per quanto riguarda i tag assegnati dagli utenti abbiamo detto che generalmente sono corretti, quindi è accettabile assegnarli tutti ai positivi. Per quanto riguarda i negativi invece possiamo individuare due casi:

- Il tag è effettivamente non rilevante per l'immagine e in quanto tale non dovrebbe esservi assegnato.
- Il tag è rilevante per l'immagine ma l'utente non lo ha inserito. Forse era un termine sconosciuto per l'utente specifico oppure è una dimenticanza, o ancora potrebbe essere stato considerato non rilevante. Si tratta del problema dei tag mancanti.

Da questa osservazione possiamo costruire uno **schema ternario**: i tag presenti rientrano nei tag positivi mentre non tutti i tag assenti sono tag negativi. Generalmente i tag positivi non si attivano singolarmente, sono spesso gruppi di tag [DBLFF10]. Per scegliere i tag da considerare negativi possiamo individuare i tag simili ai positivi tra tutti quelli mancanti. Inoltre possiamo

considerare anche quelli che compaiono insieme nel training set. L'insieme dei tag negativi è quindi formato da tutti i tag che non sono troppo simili ai positivi e che inoltre non compaiono spesso insieme. I tag rimanenti sono considerati neutri per quell'immagine e lasciamo che sia il modello appreso a completarli successivamente.

Decidiamo di utilizzare lo schema ternario per la nostra prova in quanto ha dimostrato di avere prestazioni superiori agli altri schemi [SXL12].

Regolarizzazione dei fattori latenti

Lo schema fin qui descritto è puramente relazionale: il modello si basa soltanto sulle relazioni tag, owner e il ranking imposto. È possibile utilizzare una serie di vincoli *soft* per imporre la conoscenza sulle correlazioni dei tre modi considerati: due immagini simili dovrebbero avere fattori latenti simili, lo stesso per gli utenti e i tag. Introduciamo allora tre matrici di similarità $W^U \in \mathbb{R}^{|U|\times|U|}$ per gli utenti, $W^I \in \mathbb{R}^{|I|\times|I|}$ per le immagini e $W^T \in \mathbb{R}^{|T|\times|T|}$ per i tag. Ogni elemento $w_{i,j}$ di ognuna di queste tre matrici è un valore in $[0, 1]$ che indica la similarità tra la coppia di elementi i e j. Una coppia di elementi che risulta molto simile avrà un valore vicino ad 1 e vogliamo quindi che i loro fattori latenti siano poco distanti. Viceversa se una coppia di elementi ha un valore di similarità vicino a 0, non si influenzano tra di loro e possono trovarsi in qualsiasi punto dello spazio latente. Questo si traduce nell'imporre i seguenti tre termini di regolarizzazione:

$$\sum_{m=1}^{|U|}\sum_{n=1}^{|U|} W^U_{m,n}||\mathbf{u}_m - \mathbf{u}_n||^2 \tag{5.19}$$

$$\sum_{m=1}^{|I|}\sum_{n=1}^{|I|} W^I_{m,n}||\mathbf{i}_m - \mathbf{i}_n||^2 \tag{5.20}$$

$$\sum_{m=1}^{|T|}\sum_{n=1}^{|T|} W^T_{m,n}||\mathbf{t}_m - \mathbf{t}_n||^2 \tag{5.21}$$

dove i termini $\mathbf{u}_i$, $\mathbf{i}_i$ e $\mathbf{t}_i$ sono le righe dell'elemento i-esimo rispettivamente delle matrici $\mathbf{U}, \mathbf{I}, \mathbf{T}$ ovvero i fattori latenti per l'elemento i. Utilizziamo la distanza euclidea per calcolare la distanza tra i fattori dei due elementi

e imponiamo un costo direttamente proporzionale alla distanza pesata dal valore di similarità. I valori di similarità tra 2 elementi assumono quindi il significato di peso per il costo che impone la loro distanza nello spazio latente.

Imponendo soltanto dei vincoli soft si ha il vantaggio che il modello ha la possibilità di andare contro la conoscenza pregressa e quindi essere potenzialmente più robusto al rumore introdotto da una non perfetta similarità. Questo impone anche dei lati negativi, principalmente computazionali in quanto dobbiamo calcolare obbligatoriamente la similarità tra un numero potenzialmente alto di elementi. Il numero di elementi delle matrici di similarità cresce al quadrato rispetto al numero di oggetti del modello, portando presto a problemi computazionali di calcolo e di spazio.

Un altro modo di vedere le matrici di similarità è di considerare un grafo completo per gli utenti, uno per le immagini e uno per i tag. In questo caso, utilizzando il laplaciano del grafo, si possono regolarizzare i fattori latenti.

Data la matrice A di adiacenza di un grafo, il laplaciano è definito:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{5.22}$$

dove $D = diag(A)$. Data una funzione f qualsiasi a valori reali sui vertici di un grafo, si può realizzare un fattore di regolarizzazione:

$$< f, Lf >= f^T Lf = -\frac{1}{2} \sum_{i \sim j} (f_i - f_j)^2 \tag{5.23}$$

Nel nostro caso possiamo utilizzare una forma compatta calcolando la traccia del prodotto scalare. I tre termini di regolarizzazione diventano:

$$tr(\hat{U}^T L^U \hat{U}) \tag{5.24}$$

$$tr(\hat{I}^T L^I \hat{I}) \tag{5.25}$$

$$tr(\hat{T}^T L^T \hat{T}) \tag{5.26}$$

dove tr è l'operatore traccia e L^U, L^I, L^T sono le matrici laplaciane del grafo degli utenti, delle immagini e dei tag. Questo tipo di regolarizzazione funge sia da fattore regolarizzatore in norma 2 sia come guida per lo spazio dei fattori latenti. Inoltre esercita un controllo sull'uscita del sistema e allevia in parte il problema della sparsità.

Considerando la regolarizzazione sulla similarità, l'obiettivo finale diventa:

$$\begin{aligned}\underset{\hat{\theta}}{\operatorname{argmin}} & \sum_{(u,i)\in U\times I} \frac{1}{|T^+_{u,i}||T^-_{u,i}|} \sum_{t^+\in T^+_{u,i}} \sum_{t^-\in T^-_{u,i}} H_{0.5}(\hat{y}_{u,i,t^-} - \hat{y}_{u,i,t^+}) \\ & + \gamma_C ||\hat{C}||^2_F + \gamma_\beta (||\hat{U}||^2_F + ||\hat{I}||^2_F + ||\hat{T}||^2_F) \\ & + \gamma_{\alpha,1} tr(\hat{U}^T L^U \hat{U}) + \gamma_{\alpha,2} tr(\hat{I}^T L^I \hat{I}) + \gamma_{\alpha,3} tr(\hat{T}^T L^T \hat{T})\end{aligned} \tag{5.27}$$

Costruzione dei grafi di similarità

Abbiamo bisogno di un grafo di similarità per ogni modo ed essendo un grafo completo, abbiamo bisogno di una matrice di similarità con valori in $[0, 1]$.

Per quanto riguarda la similarità tra utenti dobbiamo misurare una sorta di affinità basandosi sugli interessi e sulle attività passate. Un modo semplice per confrontare utenti è quello di esaminare i gruppi di Flickr a cui appartengono: ogni utente può iscriversi liberamente o su invito a gruppi di persone che condividono il luogo, la lingua, la passione per la fotografia di certi tipi di oggetti, stili della tecnica e così via. Possiamo misurare l'affinità di una persona con un altra valutando il numero di gruppi che hanno in comune:

$$W^U_{i,j} = \frac{n(u_i, u_j)}{n(u_i) + n(u_j)} \tag{5.28}$$

dove $n(u_i, u_j)$ è il numero di gruppi che gli utenti u_i, u_j hanno in comune e $n(u_i)$ è il numero di utenti dell'utente u_i.

Per quanto riguarda la similarità tra tag possiamo utilizzare un criterio basato sulla similarità semantica e uno basato sulla esistenza contestuale. Utilizziamo l'approccio di Liu et al. [LHWZ10] per misurare la similarità semantica utilizzando la Lin Similarity in WordNet:

$$T^s_{i,j} = \frac{2 \cdot IC(lcs(t_i, t_j))}{IC(t_i) + IC(t_j)} \tag{5.29}$$

dove $IC(\cdot)$ è l'*Information Content*, *lcs* è il *Least Common Subsumer* (il progenitore più lontano in comune). Il risultato è dipendente dal testo utilizzato per la generazione del dizionario. Decidiamo di utilizzare il corpus

Brown [FK79]. Per misurare l'esistenza contestuale utilizziamo una semplice co-occorrenza dei tag nelle immagini:

$$T^c_{i,j} = \frac{n(t_i, t_j)}{n(t_i) + n(t_j)} \tag{5.30}$$

dove $n(t_i, t_j)$ è il numero di immagini che hanno entrambi i tag t_i, t_j e $n(t_i)$ è il numero di immagini che ha il tag t_i. La similarità e la contestualità tra tag vengono uniti nel valore finale di similarità tra i due tag secondo un peso complementare:

$$W^T_{i,j} = \lambda \cdot T^c_{i,j} + (1 - \lambda) \cdot T^s_{i,j} \tag{5.31}$$

Infine, per quanto riguarda la similarità tra immagini utilizziamo una distanza tra immagini usando le features visuali e basata sul kernel gaussiano:

$$W^I_{i,j} = e^{-\frac{\mu(I_i, I_j)}{\sigma^2}} \tag{5.32}$$

dove la distanza μ è diversa a seconda del dataset utilizzato ed è la stessa utilizzata nelle precedenti tecniche TagSimilar e TagRelevance per individuare i vicini visuali.

Algoritmo di apprendimento

Ottimizzare direttamente l'equazione 5.27 non è fattibile. Utilizziamo un algoritmo di discesa del gradiente per minimizzare la funzione obiettivo. La forma non è differenziabile per via della funzione di Heaviside, la sostituiamo quindi con la funzione logistica:

$$s(x) = \frac{1}{1 + e^{-x}} \tag{5.33}$$

La funzione obiettivo così realizzata non è una funzione convessa, tuttavia lo è se consideriamo il tensore $\underline{\mathbf{C}}$ e le matrici U, I, T uno alla volta e teniamo fisse le altre tre. Possiamo realizzare un algoritmo ALS tipo quello descritto per la decomposizione non negativa di matrici 3.2.4 oppure un algoritmo di tipo stocastico. Utilizzare una forma di gradiente batch è infattibile per un problema a larga scala con tanti esempi ed è conveniente passare ad un algoritmo di tipo stocastico [BBZ08].

Tra l'utilizzo di un algoritmo ALS oppure uno con gradiente stocastico puro, si preferisce la seconda per via della velocità di convergenza: ogni volta che si trova un minimo per uno dei sotto problemi, ci si allontana dai minimi degli altri sotto problemi. Globalmente ci avviciniamo sempre di più a un minimo locale del problema generale, ma ogni volta paghiamo diverse iterazioni per avvicinarsi molto lentamente ai minimi dei sotto problemi, per poi ripartire e andare da un'altra parte per il sotto problema seguente.

Decidiamo quindi di utilizzare un gradiente stocastico puro. Calcoliamo le derivate per ogni parametro del modello, una parte per volta. Concentriamoci sulla prima parte dell'obiettivo, ovvero la decomposizione. Per ogni immagine abbiamo:

$$\begin{aligned}\frac{\partial}{\partial x}g &= \frac{\partial}{\partial x}\frac{1}{|T^+_{u,i}||T^-_{u,i}|}\sum_{t^+\in T^+_{u,i}}\sum_{t^-\in T^-_{u,i}} s(\hat{y}_{u,i,t^-}-\hat{y}_{u,i,t^+}) \\ &= z\sum_{t^+\in T^+_{u,i}}\sum_{t^-\in T^-_{u,i}} w_{t^+,t^-}\frac{\partial}{\partial x}(\hat{y}_{u,i,t^-}-\hat{y}_{u,i,t^+})\end{aligned} \tag{5.34}$$

dove:

$$z = \frac{1}{|T^+_{u,i}||T^-_{u,i}|} \tag{5.35}$$

$$w_{t^+,t^-} = s(\hat{y}_{u,i,t^-}-\hat{y}_{u,i,t^+})(1-s(\hat{y}_{u,i,t^-}-\hat{y}_{u,i,t^+})) \tag{5.36}$$

$$\hat{y}_{u,i,t^-}-\hat{y}_{u,i,t^+} = \sum_{\tilde{u}}\sum_{\tilde{i}}\sum_{\tilde{t}} c_{\tilde{u},\tilde{i},\tilde{t}}u_{u,\tilde{u}}i_{i,\tilde{i}}(t_{t^-,\tilde{t}}-t_{t^+,\tilde{t}}) \tag{5.37}$$

In questo modo semplifichiamo la notazione. La derivata rispetto al tensore nucleo $\underline{\mathbf{C}}$ è:

$$\frac{\partial g}{\partial c_{u,i,t}} = z\sum_{t^+\in T^+_{u,i}}\sum_{t^-\in T^-_{u,i}} w_{t^+,t^-}u_{u,\tilde{u}}i_{i,\tilde{i}}(t_{t^-,\tilde{t}}-t_{t^+,\tilde{t}}) \tag{5.38}$$

La derivata rispetto alla matrice U:

$$\frac{\partial g}{\partial u_{u,\tilde{u}}} = z\sum_{t^+\in T^+_{u,i}}\sum_{t^-\in T^-_{u,i}}\sum_{\tilde{i}}\sum_{\tilde{t}} w_{t^+,t^-}c_{\tilde{u},\tilde{i},\tilde{t}}i_{i,\tilde{i}}(t_{t^-,\tilde{t}}-t_{t^+,\tilde{t}}) \tag{5.39}$$

La derivata rispetto alla matrice I:

$$\frac{\partial g}{\partial i_{i,\tilde{i}}} = z\sum_{t^+\in T^+_{u,i}}\sum_{t^-\in T^-_{u,i}}\sum_{\tilde{u}}\sum_{\tilde{t}} w_{t^+,t^-}c_{\tilde{u},\tilde{i},\tilde{t}}u_{u,\tilde{u}}(t_{t^-,\tilde{t}}-t_{t^+,\tilde{t}}) \tag{5.40}$$

Infine la derivata rispetto a T, che è diversa in caso il singolo tag sia positivo o negativo:

$$\frac{\partial g}{\partial t_{t^+,\tilde{t}}} = -z \sum_{t^- \in T^-_{u,i}} \sum_{\tilde{u}} \sum_{\tilde{i}} w_{t^+,t^-} c_{\tilde{u},\tilde{i},\tilde{t}} u_{u,\tilde{u}} i_{i,\tilde{i}} \tag{5.41}$$

$$\frac{\partial g}{\partial t_{t^-,\tilde{t}}} = z \sum_{t^+ \in T^+_{u,i}} \sum_{\tilde{u}} \sum_{\tilde{i}} w_{t^+,t^-} c_{\tilde{u},\tilde{i},\tilde{t}} u_{u,\tilde{u}} i_{i,\tilde{i}} \tag{5.42}$$

$$\tag{5.43}$$

Per quanto riguarda i fattori di regolarizzazione in norma di Frobenius:

$$\frac{\partial}{\partial \hat{C}} \gamma_C ||\hat{C}||_F^2 + \gamma(||\hat{U}||_F^2 + ||\hat{I}||_F^2 + ||\hat{T}||_F^2) = 2\gamma_C \hat{C} \tag{5.44}$$

$$\frac{\partial}{\partial \hat{U}} \gamma_C ||\hat{C}||_F^2 + \gamma(||\hat{U}||_F^2 + ||\hat{I}||_F^2 + ||\hat{T}||_F^2) = 2\gamma \hat{U} \tag{5.45}$$

$$\frac{\partial}{\partial \hat{I}} \gamma_C ||\hat{C}||_F^2 + \gamma(||\hat{U}||_F^2 + ||\hat{I}||_F^2 + ||\hat{T}||_F^2) = 2\gamma \hat{I} \tag{5.46}$$

$$\frac{\partial}{\partial \hat{T}} \gamma_C ||\hat{C}||_F^2 + \gamma(||\hat{U}||_F^2 + ||\hat{I}||_F^2 + ||\hat{T}||_F^2) = 2\gamma \hat{T} \tag{5.47}$$

e i fattori di regolarizzazione per la similarità, sfruttando la simmetria della matrice laplaciana:

$$\frac{\partial}{\partial \hat{C}} \gamma_{\alpha,1}\ tr(\hat{U}^T L^U \hat{U}) + \gamma_{\alpha,2}\ tr(\hat{I}^T L^I \hat{I}) + \gamma_{\alpha,3}\ tr(\hat{T}^T L^T \hat{T}) = 0 \tag{5.48}$$

$$\frac{\partial}{\partial \hat{U}} \gamma_{\alpha,1}\ tr(\hat{U}^T L^U \hat{U}) + \gamma_{\alpha,2}\ tr(\hat{I}^T L^I \hat{I}) + \gamma_{\alpha,3}\ tr(\hat{T}^T L^T \hat{T}) = 2\gamma_{\alpha,1} L^U \hat{U} \tag{5.49}$$

$$\frac{\partial}{\partial \hat{I}} \gamma_{\alpha,1}\ tr(\hat{U}^T L^U \hat{U}) + \gamma_{\alpha,2}\ tr(\hat{I}^T L^I \hat{I}) + \gamma_{\alpha,3}\ tr(\hat{T}^T L^T \hat{T}) = 2\gamma_{\alpha,2} L^I \hat{I} \tag{5.50}$$

$$\frac{\partial}{\partial \hat{T}} \gamma_{\alpha,1}\ tr(\hat{U}^T L^U \hat{U}) + \gamma_{\alpha,2}\ tr(\hat{I}^T L^I \hat{I}) + \gamma_{\alpha,3}\ tr(\hat{T}^T L^T \hat{T}) = 2\gamma_{\alpha,3} L^T \hat{T} \tag{5.51}$$

Date le derivate, descriviamo l'algoritmo iterativo per l'apprendimento del modello nel riquadro 1.

5.2.4 Esecuzione del tag refinement

Una volta appreso il modello possiamo rifinire i tag delle immagini del campione scelto. Data un'immagine i di test e l'utente u che la possiede, la

Algorithm 1: Discesa del gradiente stocastica per l'apprendimento di RankingDecomposition.

Input: $\Phi, \varphi, \gamma, \gamma_C, \gamma_{\alpha,1}, \gamma_{\alpha,2}, \gamma_{\alpha,3}$
Output: $\hat{C}, \hat{U}, \hat{I}, \hat{T}$
inizializza $\hat{\theta} = (\hat{C}, \hat{U}, \hat{I}, \hat{T})$;
while *il criterio di stop non è raggiunto* **do**
  for $(u, i) \in U \times I$ **do**
    for $(\tilde{u}, \tilde{i}, \tilde{t}) \in r_u \times r_i \times r_t$ **do**
      $\hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \leftarrow \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} - \varphi(\frac{\partial g}{\partial \hat{c}_{\tilde{u},\tilde{i},\tilde{t}}} + 2\gamma_C \hat{c}_{\tilde{u},\tilde{i},\tilde{t}})$
    end
    for $\tilde{u} \leftarrow 1, \ldots, r_u$ **do**
      $\hat{u}_{u,\tilde{u}} \leftarrow \hat{u}_{u,\tilde{u}} - \varphi(\frac{\partial g}{\partial \hat{u}_{u,\tilde{u}}} + 2\gamma \hat{u}_{u,\tilde{u}} + 2\gamma_{\alpha,1} L^U_{u,:} \hat{u}_{:,\tilde{u}})$
    end
    for $\tilde{i} \leftarrow 1, \ldots, r_i$ **do**
      $\hat{i}_{i,\tilde{i}} \leftarrow \hat{i}_{i,\tilde{i}} - \varphi(\frac{\partial g}{\partial \hat{i}_{i,\tilde{i}}} + 2\gamma \hat{i}_{i,\tilde{i}} + 2\gamma_{\alpha,1} L^I_{i,:} \hat{u}_{:,\tilde{i}})$
    end
    for $t \leftarrow 1, \ldots, |T|$ **do**
      for $\tilde{t} \leftarrow 1, \ldots, r_t$ **do**
        $\hat{t}_{t,\tilde{t}} \leftarrow \hat{t}_{t,\tilde{t}} - \varphi(\frac{\partial g}{\partial \hat{t}_{t,\tilde{t}}} + 2\gamma \hat{t}_{t,\tilde{t}} + 2\gamma_{\alpha,1} L^T_{t,:} \hat{t}_{:,\tilde{t}})$
      end
    end
  end
end
return $\hat{\theta}$

predizione può essere eseguita in modo personalizzato o impersonale, cioè seguendo le preferenze dell'utente oppure no.

- La predizione **personalizzata** si ottiene eseguendo il calcolo diretto della decomposizione di Tucker e selezionando gli n tag richiesti con il maggiore valore di rilevanza:

$$\text{refine}(u,i) := \max_{t \in T}{}^{n} \underline{\mathbf{Y}}_{u,i,:} = \max_{t \in T}{}^{n} C \times_u U_{:,u} \times_i I_{:,i} \times_t T_{:,t} \tag{5.52}$$

- La predizione **impersonale** si ottiene sommando il tensore lungo il modo degli utenti e calcolando il resto della decomposizione:

$$\text{refine}(i) := \max_{t \in T}{}^{n} \sum_{u \in U} \underline{\mathbf{Y}}_{u,i,:} = \max_{t \in T}{}^{n} \sum_{u \in U} C \times_u U_{:,u} \times_i I_{:,i} \times_t T_{:,t} \tag{5.53}$$

5.3 Analisi temporale

In letteratura soltanto un lavoro di Gunhee et al. [KXT10] analizza l'andamento temporale di argomenti in insiemi generali di immagini web. Gli autori hanno selezionato un dataset da Flickr composto da 47 argomenti e utilizzando le immagini di Google Image Search per riferimento fanno interessanti osservazioni sull'andamento temporale degli argomenti. In particolare sostengono che l'utilizzo dell'associazione temporale è in grado di migliorare le performance di riconoscimento.

Nessuno ha mai provato ad applicare un approccio con associazione temporale al problema del tag refinement. Utilizzando le date di scatto delle immagini e l'andamento generale nel tempo dei tag vogliamo cercare di migliorare le performance degli esistenti algoritmi. Eseguiamo prima di tutto un'analisi dei tag nei dataset a disposizione relativo ai tag degli utenti e relativo alla ground truth. Questo tipo di analisi necessita numerosi esempi e dataset estesi.

Introduciamo una mappa temporale. Data un'immagine $i \in I$:

$$\text{time}(i) : I \to \mathbb{T} \tag{5.54}$$

dove $\mathbb{T} = \{x_1, x_2, \ldots, x_K\}$ è l'arco temporale. time(i) collega semplicemente un'immagine al tempo del suo scatto.

Analizziamo i dataset MIRFLICKR nella versione 25K e NUSWIDE per quanto riguarda gli aspetti temporali.

5.3.1 NUSWIDE

Per poter analizzare l'aspetto temporale è ovviamente necessario che il dataset includa un numero sufficiente di immagini per un periodo di tempo preferibilmente più vasto possibile. Se il numero di immagini è basso abbiamo pochi campioni e la statistica risulta inaffidabile. NUSWIDE di per sé non contiene le informazioni temporali di quando le immagini sono state scattate. È fornito però l'identificativo dell'immagine su Flickr. Questo ci permette di eseguire il *crawling* delle informazioni direttamente utilizzando le API di Flickr. Purtroppo, oltre ad essere un processo lento, non tutte le immagini del dataset sono ancora disponibili su Flickr.

Il processo di crawling è durato circa una settimana e sono state recuperate le informazioni di 243.173 immagini sul totale di 269.648, ovvero circa il 90,2% dell'intero dataset. L'arco temporale delle immagini del dataset, visibile in figura 5.1 va dal 01/01/1826 fino al 2101. Ovviamente immagini più recenti rispetto a quelle in cui è stato pubblicato il dataset (08/07/2009) non sono sicuramente valide. Invece le immagini scattate in tempi remoti, dopo una verifica manuale, risultano effettivamente della data riportata: sono state fatte in quel periodo ma sono state digitalizzate. Limitiamo le informazioni al periodo temporale dove è presente il picco, selezionando l'intervallo contiguo di lunghezza maggiore dove sono presenti almeno 15 immagini giornaliere. Il periodo scelto va dal 01/02/2005 al 12/08/2008 contando quindi circa 3 anni. La distribuzione è visibile in figura 5.2.

Vogliamo analizzare l'andamento delle singole etichette nell'arco temporale scelto. Valutiamo dapprima le etichette utilizzando le informazioni della ground truth in modo da avere segnali di riferimento, quindi analizzeremo la distribuzione dei tag. Per motivi di spazio evidenzieremo solo il comportamento di alcune etichette, ma riporteremo scritto il tipo di andamento. Per ogni etichetta è analizzato il grafico dell'andamento raggruppando gli intervalli giorno per giorno, settimana per settimana e mese per mese. Per

processare i dati facciamo uso di Google Refine[1] e per la visualizzazione le API Google Charts Tools[2] per ottenere la possibilità di interagire facilmente.

Analizziamo la distribuzione delle etichette della ground truth: per ogni etichetta costruiamo un grafico dove è riportata lungo l'arco temporale scelto la probabilità di trovare quell'etichetta. Ovvero valutiamo la probabilità frequentistica contando per ogni giorno il numero di immagini che hanno il tag rispetto al totale. Sia t la label, x un giorno dell'arco temporale e T la variabile temporale:

$$p(t|T = x) \approx \frac{|\{i \mid \exists \text{ label}(i, t) \wedge \text{time}(i) = x\}|}{|\{i \mid \text{time}(i) = x|\}} \tag{5.55}$$

Calcoliamo la probabilità di ogni etichetta per ogni giorno dell'arco temporale e realizziamo alla fine tre grafici con i valori ottenuti, uno con granularità giornaliera, uno settimanale ed infine uno mensile. Per studiare le serie temporali è importante visualizzare sempre la forma del segnale e risulta spesso utile raggruppare i giorni per rivelare dati ulteriori. Ogni grafico comprende due insiemi di punti: la probabilità di presenza dell'etichetta e il numero di immagini scattate. Il numero di immagini scattate è un indice per valutare la bontà della probabilità. Un numero di immagini elevato (quindi un campione di immagini elevato) indica una bontà maggiore. Dall'analisi delle 81 etichette si notano comportamenti diversi:

- etichette generalmente **stabili** quali *vehicle*, *animal*, *buildings* che esibiscono un comportamento tendenzialmente costante o con oscillazioni lievi. Sono etichette presenti generalmente sempre senza un periodo temporale più probabile.

- etichette generalmente **con trend** quali *boats*, *cat* che esibiscono un comportamento con trend ovvero un aumento o una diminuzione lineare lungo l'arco temporale.

- etichette generalmente **sporadiche** quali *book*, *protest*, *earthquake*, *fire* che seguono un comportamento randomico ma con picchi sparsi in giornate sporadiche.

[1]Google Refine. `http://code.google.com/p/google-refine/`
[2]Google Charts Tools `https://developers.google.com/chart/`

- etichette generalmente **stagionali** quali *snow*, *frost*, *tree* che mantengono un andamento oscillatorio o stagionale con periodo annuale, mensile, settimanale e così via.

- etichette che hanno una combinazione dei comportamenti elencati come ad esempio *flowers* che risulta stagionale e possiede un trend positivo.

I segnali sono ben visibili ad occhio e ciò indica che probabilmente è possibile sfruttarli facilmente per migliorare le performance di categorizzazione. Il dataset NUSWIDE possiede quindi un insieme di immagini di cui alcune etichette sono effettivamente correlate temporalmente.

Viene naturale cercare di interpretare i grafici secondo qualche logica. Ad esempio:

- l'etichetta *snow* esibisce un andamento stagionale annuale. L'andamento del tag riflette l'utilizzo che ne fa la popolazione di Flickr, ovvero persone generalmente in paesi industrializzati, situati nell'emisfero nord. La neve è presente generalmente d'inverno e infatti, durante i periodi invernali, si assiste a un aumento del numero di foto con la neve.

- l'etichetta *animal* esibisce un comportamento stabile durante l'anno. Le persone fotografano animali durante tutto il periodo dell'anno e nel tempo negli ultimi anni non si è assistito a nessun cambiamento dell'abitudine.

- l'etichetta *earthquake* appare in modo sparso alcuni giorni dell'anno. Da un'esame delle giornate in questione utilizzando Google News, i picchi sono avvenuti proprio in giornate con terremoti.

È possibile inferire diverse informazioni soltanto analizzando questi grafici e viene naturale pensare ad un possibile impiego nello studio della profilazione degli utenti o nell'analisi dei trend.

Analizziamo anche i tag degli utenti per vedere se questi riflettono effettivamente un andamento temporale e quanto si avvicinano all'andamento corretto della ground truth. Allo stesso modo della ground truth, realizziamo

un grafico per ogni tag. Considerando che gli utenti hanno la possibilità di utilizzare una funzione di *batch tagging* (tagging automatico di un insieme di immagini con gli stessi tag), se si analizzano direttamente i dati grezzi si potrebbero riscontrare dei giorni dove alcuni tag sono sovra utilizzati. Per rimuovere il fenomeno decidiamo che ogni utente, per ogni giorno può avere soltanto una combinazione di tag. Anche se viene ripetuta su diverse immagini, questa conta soltanto una volta nel conteggio. Analizziamo i dati risultati confrontando i tag con le etichette della groundtruth e vediamo che effettivamente gli andamenti sono molto simili a quelli delle etichette corrispondenti. Le etichette che avevano andamento stagionale o con trend hanno un andamento simile. Si trae la conclusione che il segnale dei tag stagionali e con trend si riflettono effettivamente nei tag degli utenti. Per quanto riguarda invece i tag con andamento sporadico sulla base di eventi, abbiamo effettivamente lo stesso tipo di fenomeno tuttavia sono generalmente parecchio affetti da rumore. È possibile vedere chiaramente il fenomeno per il tag *earthquake* che, oltre ad avere picchi di presenza nei giorni dei terremoti, possiede anche altri picchi sparsi, probabilmente perché gli utenti inseriscono foto con gli effetti dei terremoti nei giorni successivi. Per ogni tag è possibile inferire facilmente motivi per la presenza o assenza.

Altri esempi di tag con informazioni interessanti:

- il tag *canon* e *nikon* seguono un andamento con trend positivo: corrisponde all'aumento del numero di immagini inviate giornalmente su twitter. Tra i due, il tag *nikon* ha avuto una crescita superiore rispetto a *canon*, segno che nel tempo il numero di utenti che utilizzano macchine fotografiche Nikon sono aumentati più di quelli che usano macchine Canon.

- il tag *democrat* non ha un andamento facile da spiegare matematicamente, ma è possibile osservare momenti di maggior uso durante convention o eventi legati ai democratici in america.

- il tag *obama* ha un uso sporadico fino a quando non è apparso sulle scene internazionali.

- i tag dei mesi *january*, *february*, e così via, appaiono sempre nel mese citato. È presente anche rumore negli altri mesi. La stessa inferenza è valida per i giorni della settimana (*monday*, *tuesday*, etc.) e per i tag annuali (*2006*, *2007*, etc.).

- il tag *apple*, esibisce un comportamento stabile (una componente probabilmente legata per la maggior parte al significato comune di mela) con picchi durante eventi quali presentazione di nuovi prodotti a marchio Apple.

L'andamento temporale è probabilmente scindibile in componenti sulla base dei significati semantici che vi sono attribuiti. In successivi studi potrebbe essere interessante l'utilizzo delle tecniche di matrix factorization per l'analisi.

Un altro collegamento molto interessante è quello con Google Trends[3], le statistiche delle ricerche delle parole chiave su Google in relazione alle news pubblicate. I tag che abbiamo "scoperto" essere periodici, lo sono anche nei volumi di ricerca di google. Lo stesso i tag con trend o quelli con i comportamenti sporadici a picchi.

[3] `http://www.google.com/trends/`

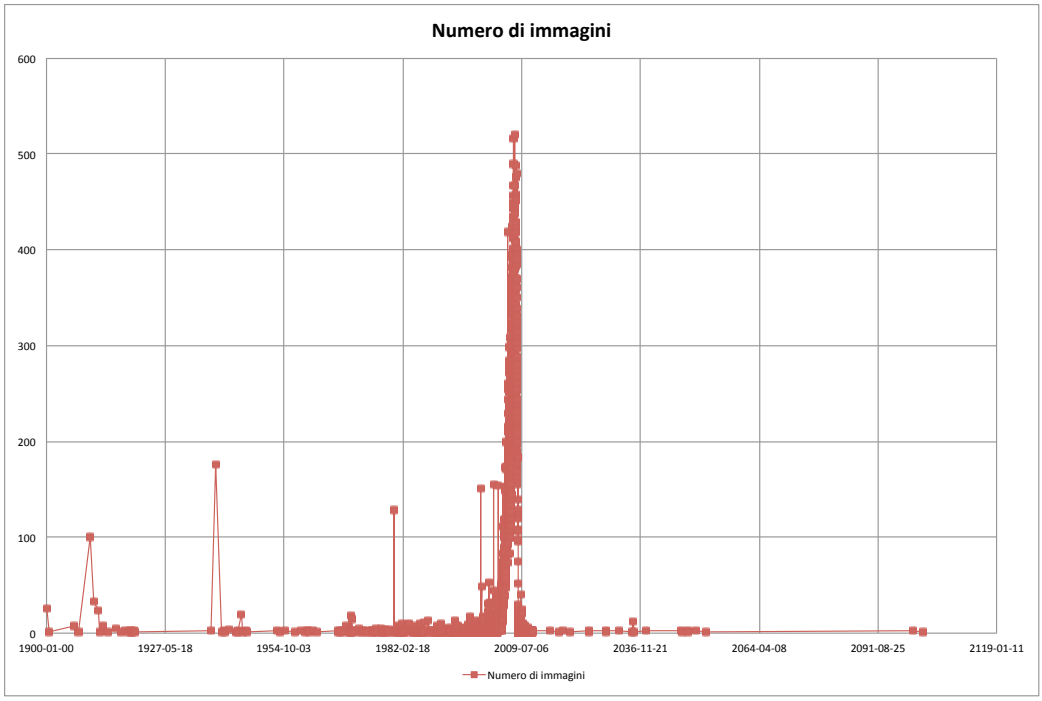

Figura 5.1. La distribuzione delle immagini recuperate da Flickr di NUSWIDE.

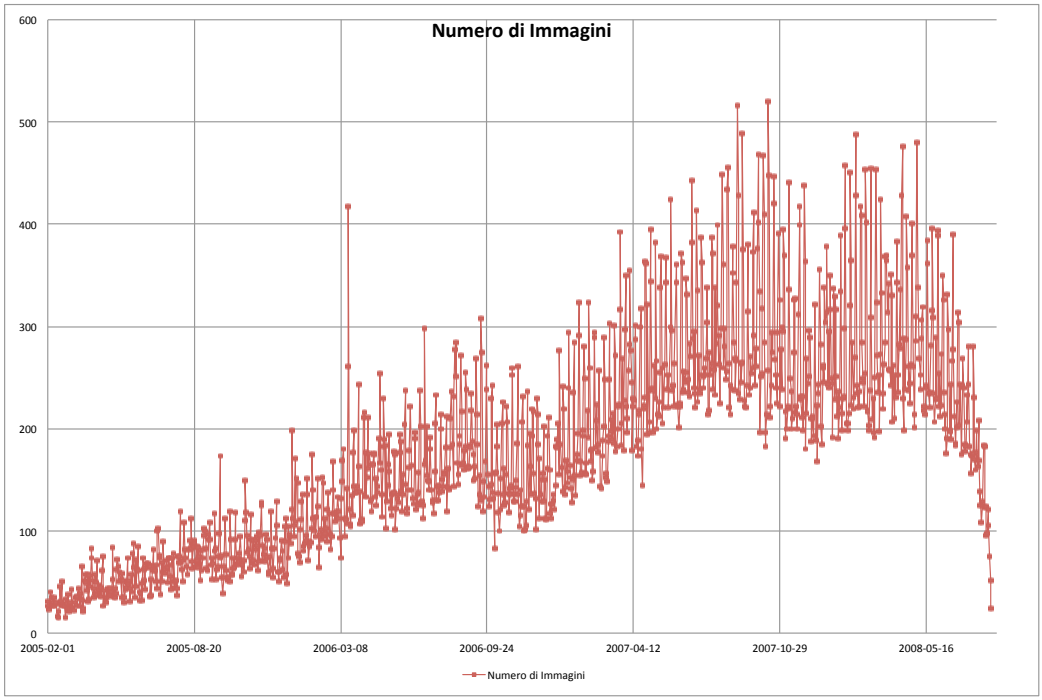

Figura 5.2. La distribuzione delle immagini di NUSWIDE nell'arco di tempo dal 01/02/2005 al 12/08/2008.

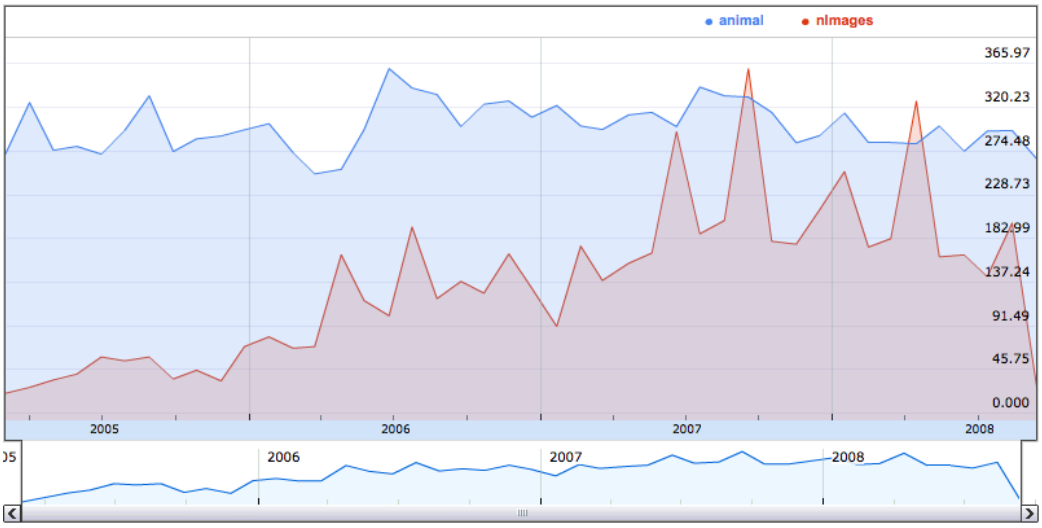

Figura 5.3. La distribuzione dell'etichetta *animal* con granularità mensile in NUSWIDE. L'andamento è generalmente costante.

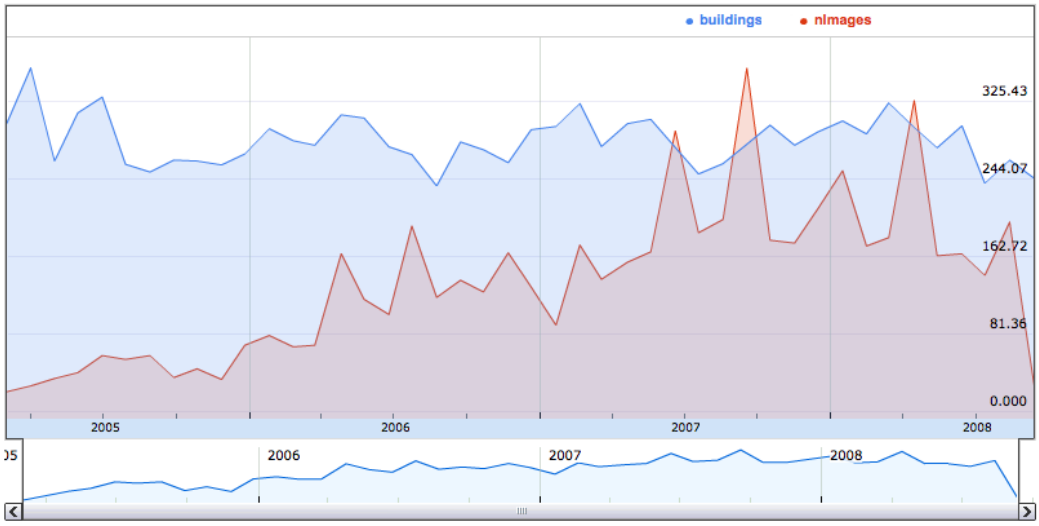

Figura 5.4. La distribuzione dell'etichetta *buildings* con granularità mensile in NUSWIDE. L'andamento è generalmente costante.

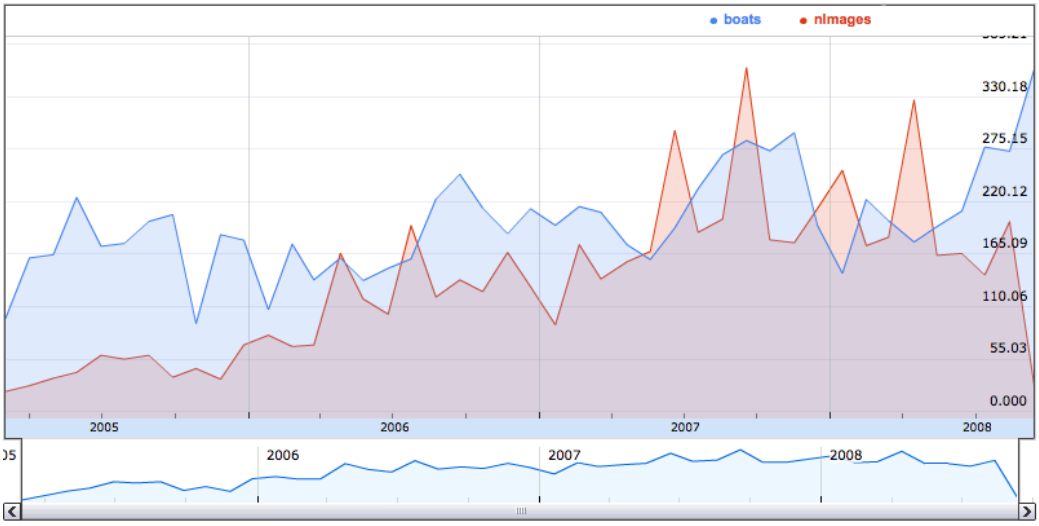

Figura 5.5. La distribuzione dell'etichetta *boats* con granularità mensile in NUSWIDE. L'andamento ha un leggero trend positivo.

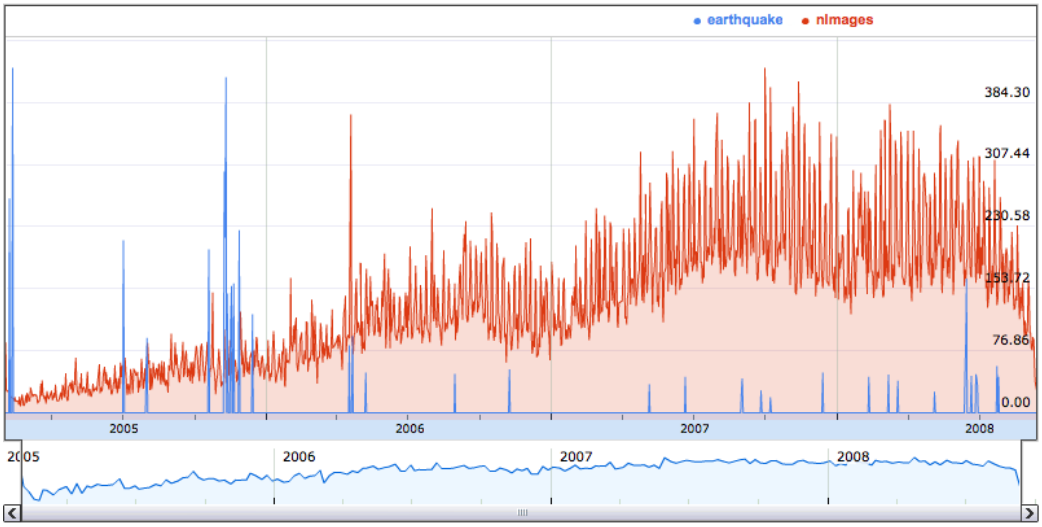

Figura 5.6. La distribuzione dell'etichetta *earthquake* con granularità giornaliera in NUSWIDE. Si notano diversi picchi sparsi in giornate sporadiche.

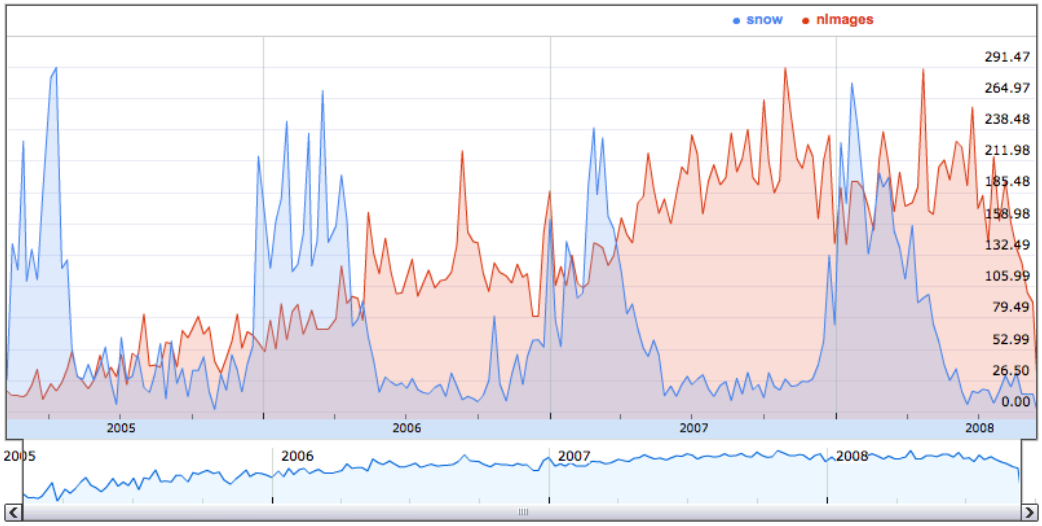

Figura 5.7. La distribuzione dell'etichetta *snow* con granularità settimanale in NUSWIDE. L'andamento è stagionale con un periodo annuale.

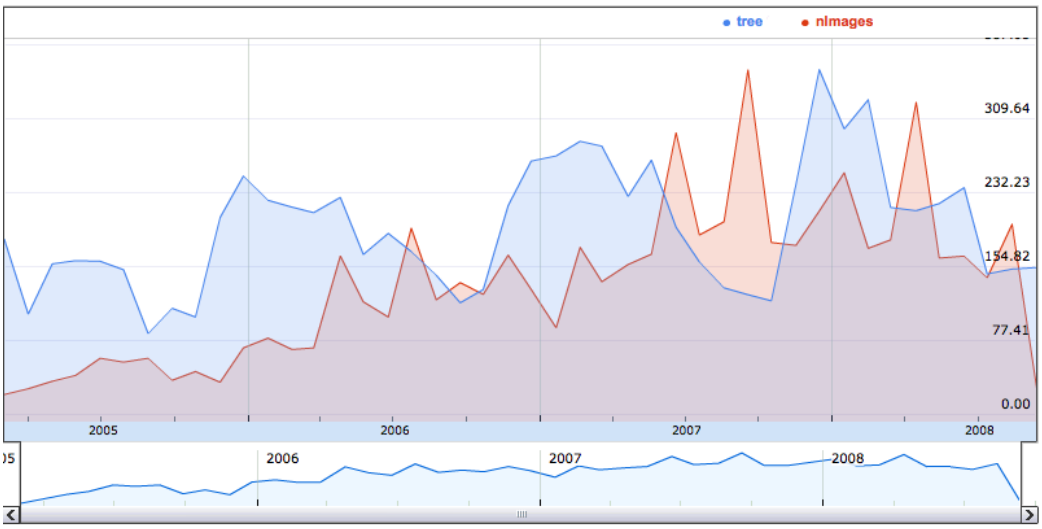

Figura 5.8. La distribuzione dell'etichetta *tree* con granularità mensile in NUSWIDE. L'andamento è stagionale con un periodo annuale.

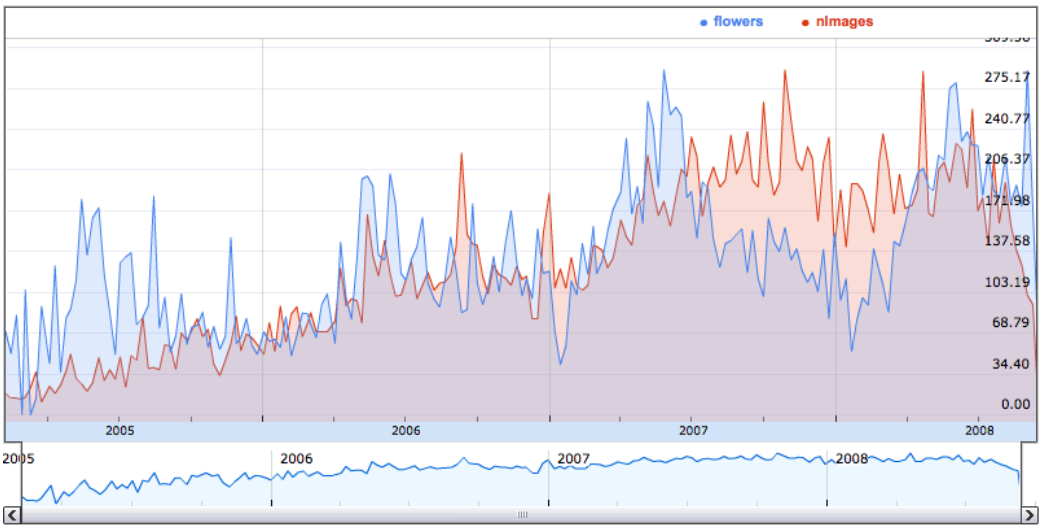

Figura 5.9. La distribuzione dell'etichetta *flowers* con granularità settimanale in NUSWIDE. L'andamento è stagionale con un periodo annuale con trend positivo.

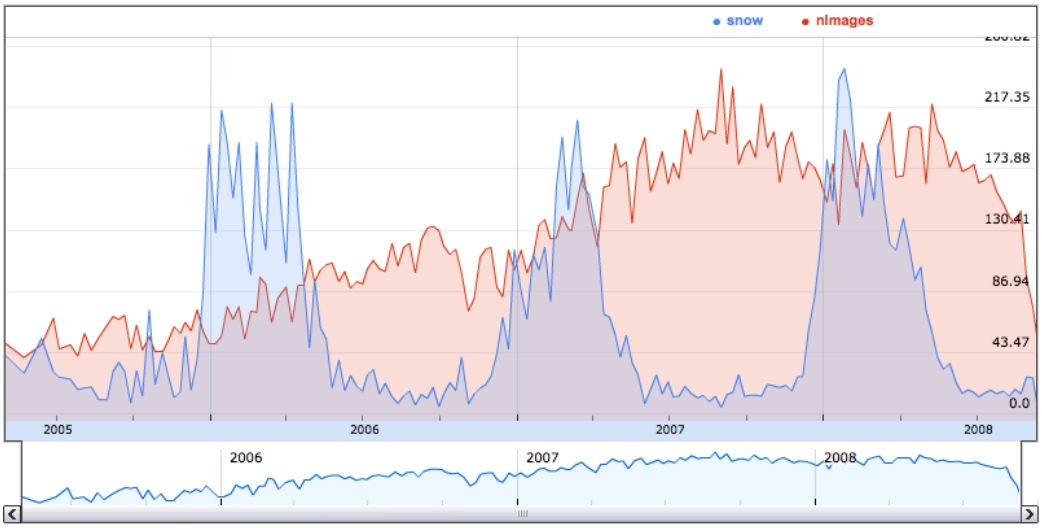

Figura 5.10. La distribuzione del tag *snow* con granularità settimanale in NUSWIDE.

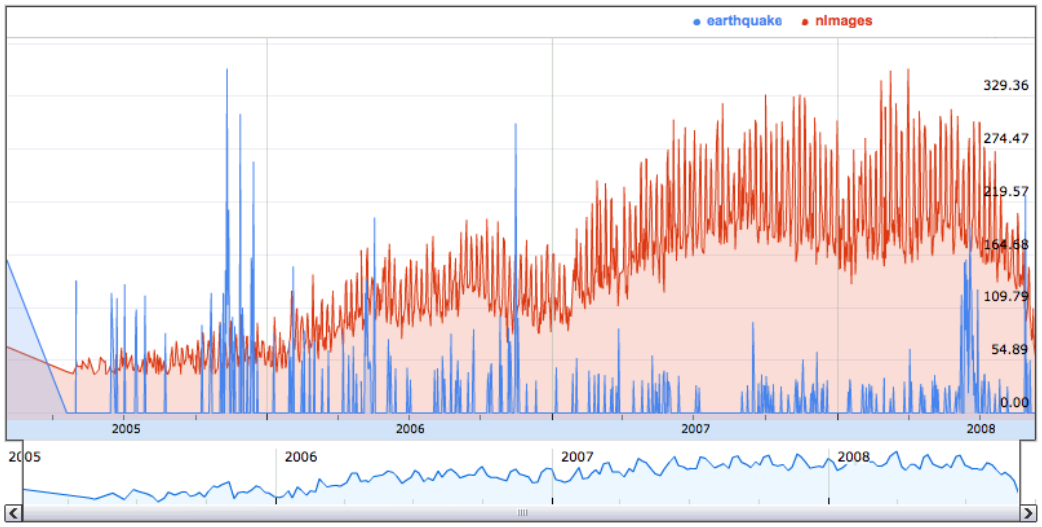

Figura 5.11. La distribuzione del tag *earthquake* con granularità giornaliera in NUSWIDE.

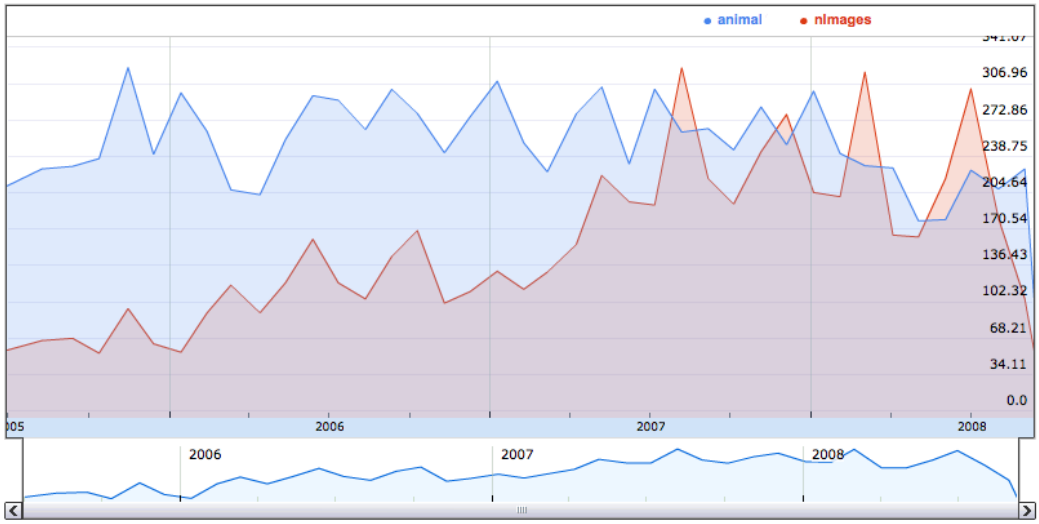

Figura 5.12. La distribuzione del tag *animal* con granularità mensile in NUSWIDE.

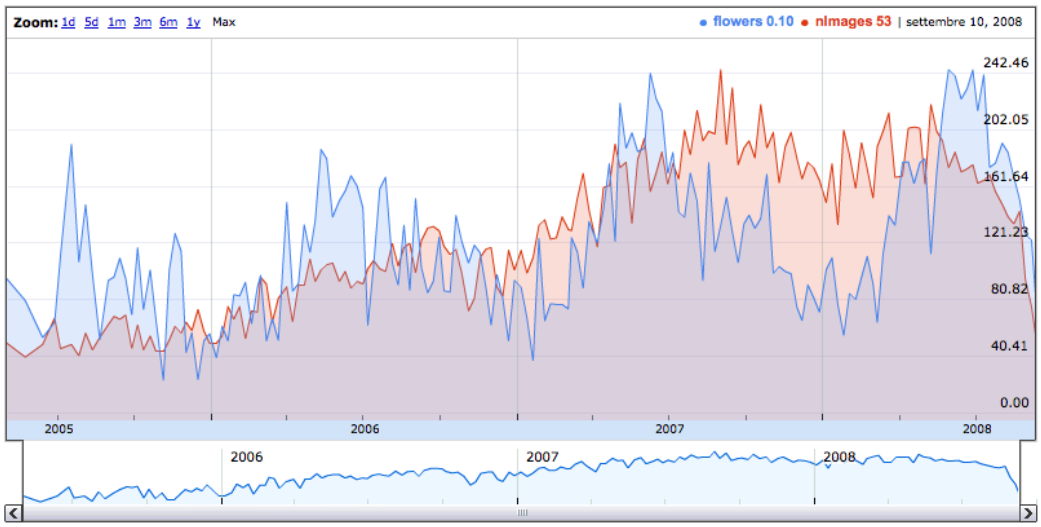

Figura 5.13. La distribuzione del tag *flowers* con granularità settimanale in NUSWIDE.

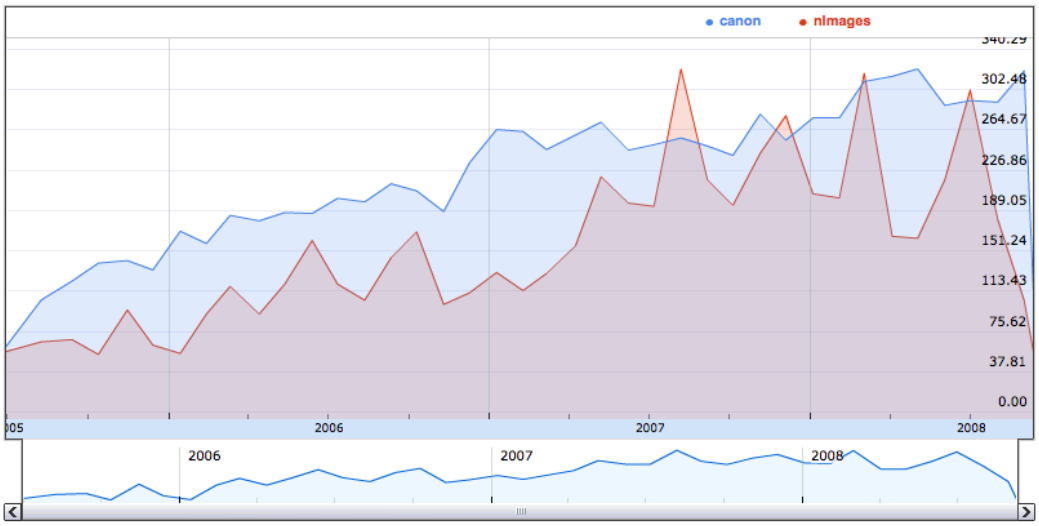

Figura 5.14. La distribuzione del tag *canon* con granularità mensile in NUSWIDE.

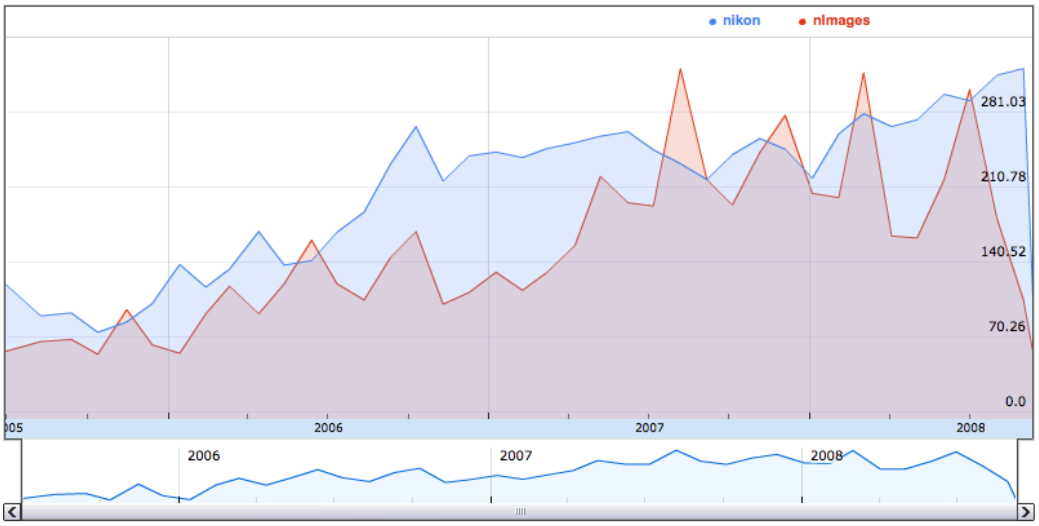

Figura 5.15. La distribuzione del tag *nikon* con granularità mensile in NUSWIDE.

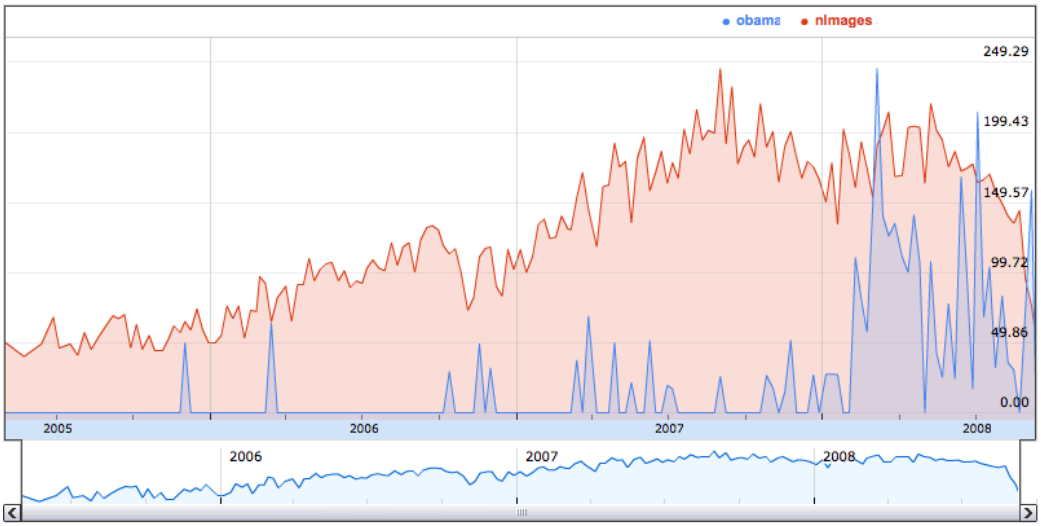

Figura 5.16. La distribuzione del tag *Obama* con granularità settimanale in NUSWIDE.

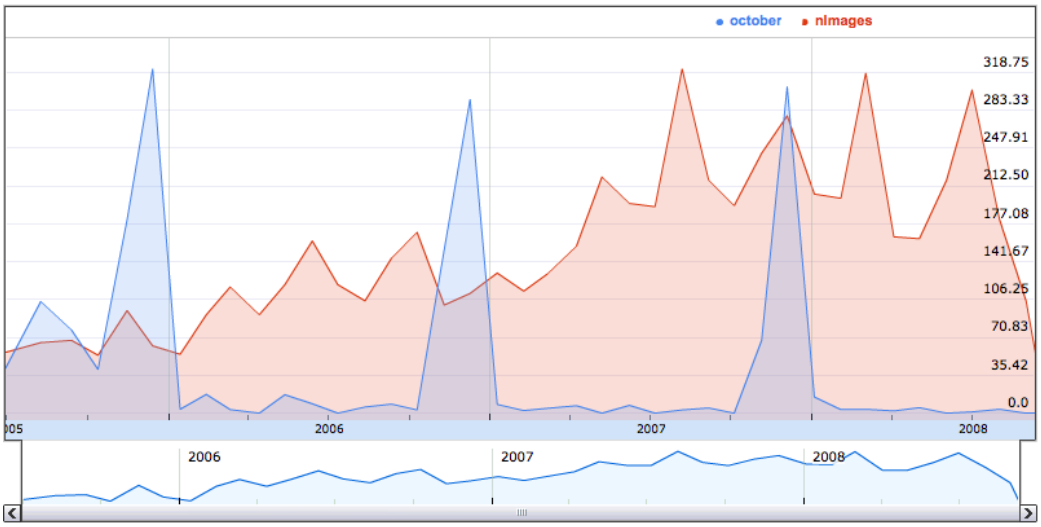

Figura 5.17. La distribuzione del tag *october* con granularità mensile in NUSWIDE.

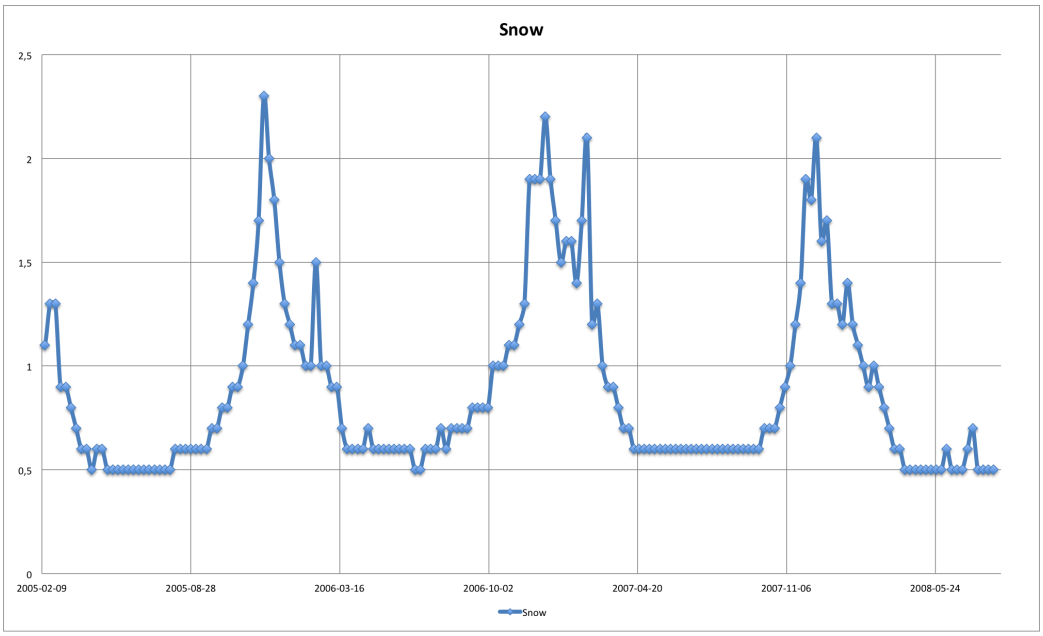

Figura 5.18. Dati di Google Trends relativi al tag *snow* nello stesso arco temporale di NUSWIDE.

5.3.2 MIRFLICKR

MIRFLICKR è un dataset composto da 25.000 immagini. Essendo un campione ridotto rispetto a NUSWIDE, ci si potrebbe aspettare che non siano sufficienti per osservare i fenomeni temporali. In effetti valutando le informazioni EXIF di MIRFLICKR soltanto circa 20.000 immagini hanno la data e l'ora di scatto. Le immagini sono state scelte utilizzando il punteggio di *interestingness* di Flickr, ed la data in cui erano interessanti sono salvate nei metadati del dataset. Decidiamo quindi di analizzare se le immagini interessanti seguono anche loro un qualche andamento temporale ricollegabile alla semantica: invece di utilizzare la data di scatto, usiamo il giorno in cui erano interessanti. L'intervallo temporale delle date va dal 21/03/2007 al 30/06/2008.

Seguiamo lo stesso procedimento usato per NUSWIDE: per ogni tag tracciamo tre grafici dell'andamento raggruppando gli intervalli giorno per giorno, settimana per settimana e mese per mese. Rispetto a NUSWIDE l'arco temporale è ridotto, ma vi è contenuto completamente. Ovviamente riducendo l'arco temporale a soltanto circa 1 anno, ci aspettiamo di individuare meno informazioni sul trend. Analizziamo rapidamente alcuni tag già osservati in NUSWIDE:

- il tag *snow* esibisce il picco nei mesi invernali come in NUSWIDE.
- il tag *animal* è generalmente costante, tuttavia non è molto usato.
- i tag *canon* e *nikon* sono generalmente costanti. In NUSWIDE era in trend positivo.

Il comportamento stagionale è presente significativamente anche in MIRFLICKR, lo stesso per i trend tuttavia sono più difficoltosi da osservare (o non ci sono affatto). I picchi relativi ad eventi sembrano presenti lo stesso, ma la scarsità dei dati non ci permette di poterlo verificare accuratamente.

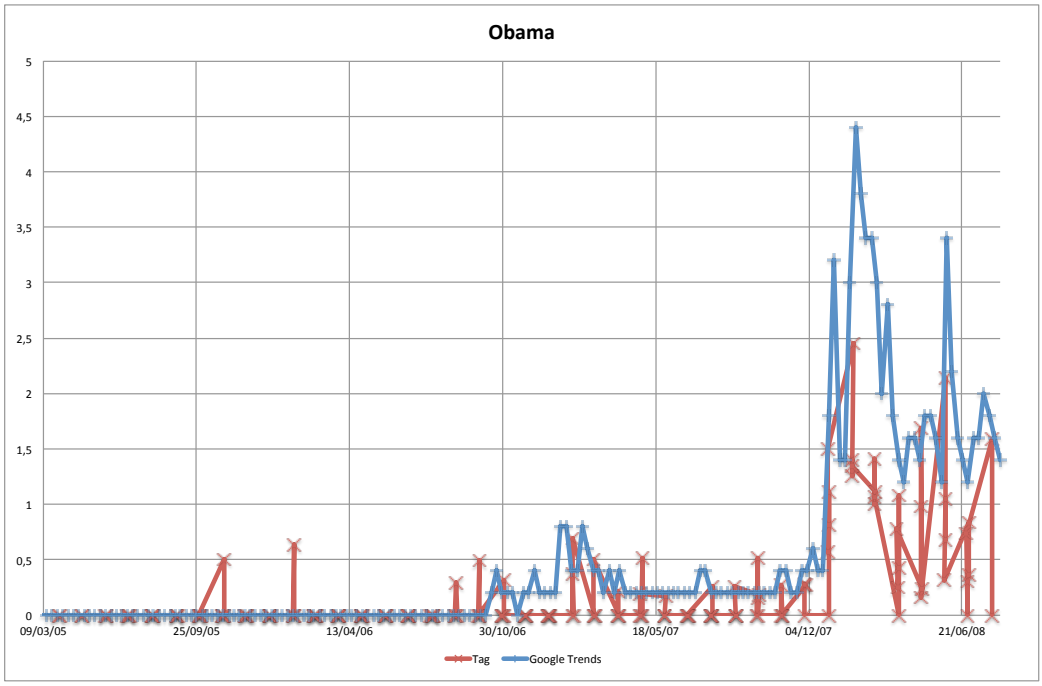

Figura 5.19. Confronto dei dati di Google Trends e dei tag di NUSWIDE relativi all'etichetta *Obama* nello stesso arco temporale.

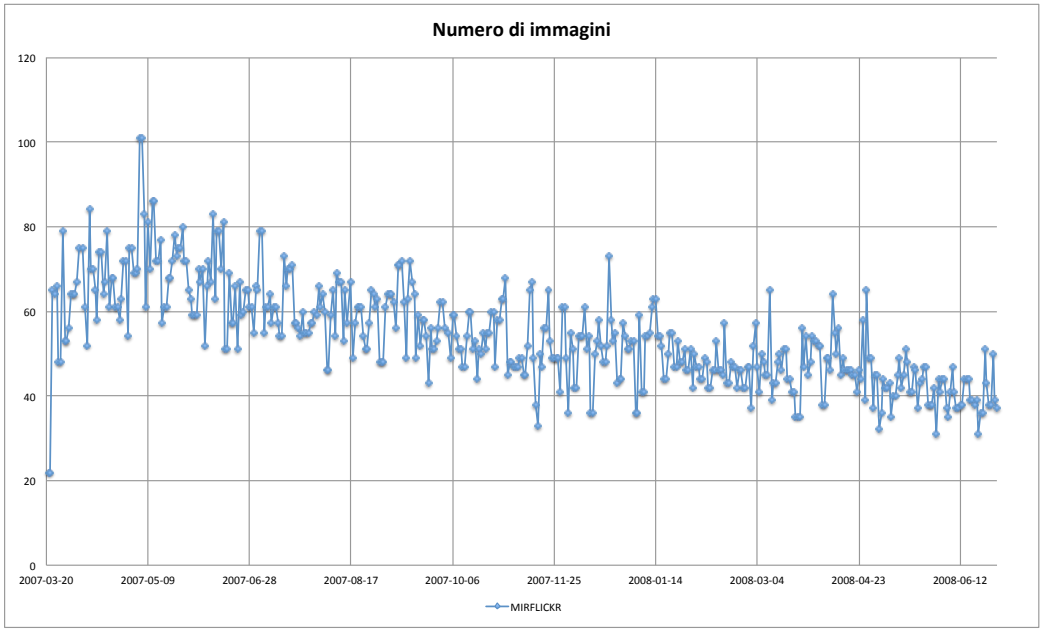

Figura 5.20. La distribuzione delle immagini di MIRFLICKR usando i tempi di *interestingness*.

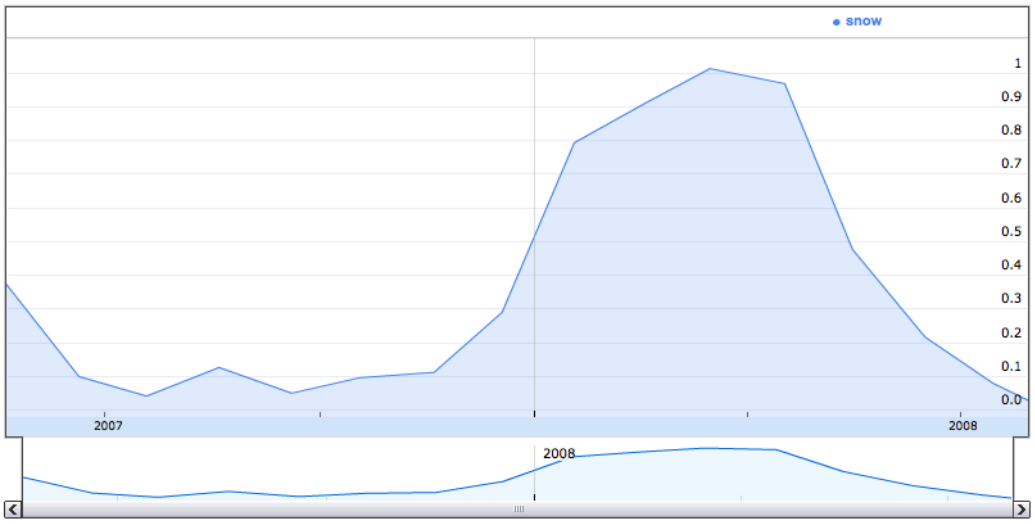

Figura 5.21. La distribuzione del tag *snow* con granularità mensile in MIRFLICKR.

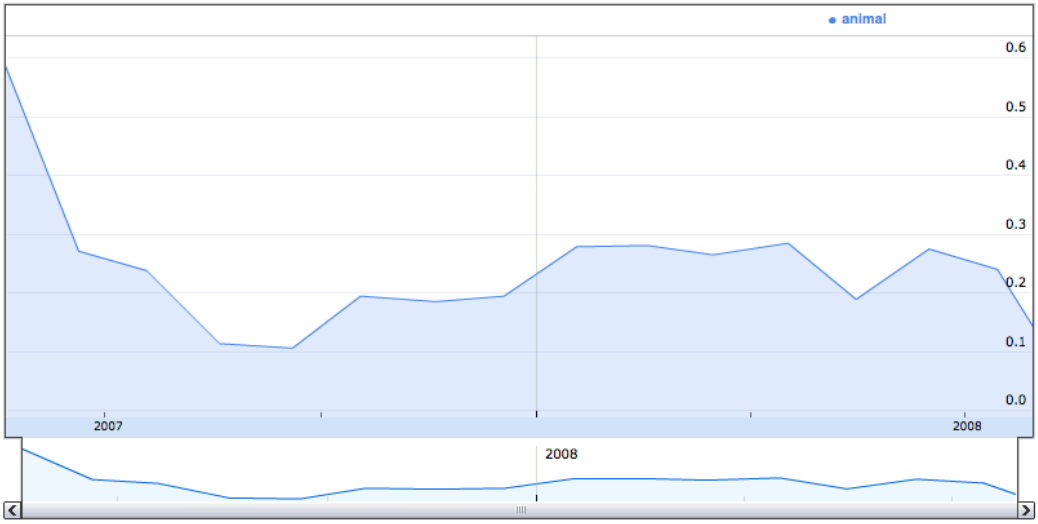

Figura 5.22. La distribuzione del tag *animal* con granularità mensile in MIRFLICKR.

5.4 TagRelevance con dati temporali

Per osservare se le informazioni temporali possono essere utili ai fini del calcolo della rilevanza e del raffinamento dei tag, utilizziamo la tecnica TagRelevance in quanto semplice di implementazione. Data un'immagine I_T di cui vogliamo eseguire il raffinamento, l'idea è di modificare la scelta del vicinato di un'immagine per farlo influenzare anche dal tempo di scatto dell'immagine.

Denominiamo **TagRelevanceTimeTag** la nuova tecnica: utilizziamo un peso per ogni votazione dei tag delle immagini vicine in base alla probabilità che sia presente nel giorno dello scatto. La definizione diventa:

$$\text{tagRTT}(t, I, k) := n_t[N_f(I, k)] * P(t|T = \text{time}(I)) - Prior(t, k) \qquad (5.56)$$

dove t è il tag di cui si vuole misurare la rilevanza, I l'immagine di riferimento, k il numero di vicini da utilizzare, T la variabile temporale. Ogni immagine del vicinato (selezionato sempre in modo che ogni utente abbia massimo un'immagine) può votare i tag che possiede sulla base della probabilità che si trovino nello stesso giorno di scatto dell'immagine. Ad esempio per un tag stagionale tipo *snow*, la probabilità che sia presente in un giorno d'inverno è più alta che in uno d'estate: quindi ad un'immagine scattata d'inverno potrà essere assegnato il tag con pochi voti nel vicinato, mentre d'estate avrà bisogno di molti più voti per essere considerato. Lo stesso tag di tipo sporadico presenti in caso di eventi, possono beneficiare della conoscenza della presenza o meno in base alla probabilità misurata. Si potrebbe anche utilizzare Google Trends o qualsiasi altra fonte per misurare la probabilità di presenza di un tag soggetto ad eventi.

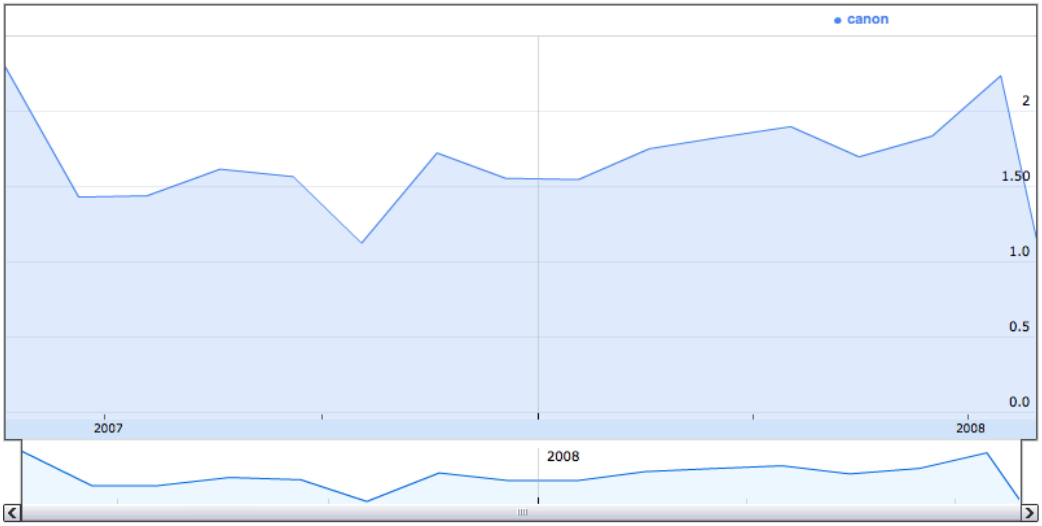

Figura 5.23. La distribuzione del tag *canon* con granularità mensile in MIRFLICKR.

6

Esperimenti

Without experiment I am nothing.
– Michael Faraday, *The Life and Letters of Faraday (1870) Vol. II*

Eseguiamo alcuni esperimenti per valutare le tre tecniche descritte nella sezione 5.2 sui principali dataset descritti nella sezione 4.1. Partiamo dalla valutazione dei dataset ai fini degli esperimenti che vogliamo condurre, dopodiché analizziamo il framework di lavoro ed infine presentiamo i risultati numerici.

6.1 Dataset

Scelte le tre tecniche da valutare, dobbiamo scegliere un dataset adatto per eseguire gli esperimenti. Il dataset ESP Game non rappresenta effettivamente il genere di immagini che troviamo sul web e quindi lo escludiamo. Viene utilizzato soltanto per la verifica dell'implementazione confrontando i risultati in letteratura.

Consideriamo quindi i dataset MIRFLICKR-25K e NUSWIDE. Entrambi hanno alcuni problemi per le nostre necessità:

- **MIRFLICKR** non ha una serie di features standardizzate: siamo costretti a scegliere le features e questo complica la comparazione con le tecniche in letteratura. Non sono presenti le informazioni sui gruppi degli utenti.

- **NUSWIDE** possiede una serie di features standardizzate, ma non possiede le immagini per un controllo visivo. Inoltre non tutte le informazioni necessarie sono presenti: mancano gli utenti delle immagini, i gruppi di cui fanno parte e le date di scatto.

In entrambi i dataset sono presenti gli id di Flickr delle immagini: questo ci permette di eseguire una procedura di crawling per recuperare i dati mancanti. Purtroppo alcune immagini sono state rimosse e alcuni utenti non sono più presenti da quando è stato eseguita la realizzazione dei due dataset. Recuperiamo più informazioni possibili, rimuovendo le immagini che non hanno più le informazioni. Per MIRFLICKR, essendo più piccolo e mancando solo l'informazione dei gruppi, teniamo tutte le immagini e non poniamo gruppi per gli utenti mancanti. Durante l'aumento delle informazioni salviamo anche una copia delle immagini.

Denominiamo **NUSWIDE-MICC** il nuovo dataset ottenuto da NUSWIDE riducendo le immagini ed aumentando le informazioni. Il nuovo dataset ha 243.173 immagini; 395.435 tags di cui 5.410 sono presenti più di 100 volte; 24.625 utenti di cui 3.035 hanno almeno 15 immagini e 579 hanno almeno 50 immagini. Vista la complessità computazionale di eseguire le prove su un dataset così grande, decidiamo di selezionarne un subset: sono conservate le immagini degli utenti che hanno almeno 50 immagini. In questo modo il numero degli utenti diminuisce considerevolmente a 579 pur mantenendo 55.831 immagini.

Per quanto riguarda le features in NUSWIDE, decidiamo di utilizzare quelle standardizzate con la distanza in norma 2:

$$\mu(I_i, I_j) = ||\mathbf{x}_i - \mathbf{x}_j||^2 \tag{6.1}$$

dove le features delle due immagini $\mathbf{x_i}$ e $\mathbf{x_j}$ sono state preventivamente normalizzate usando la norma 2. Si tratta di un vettore di 428 dimensioni con momenti di colore a blocchi di 225 dimensioni, wavelet per le texture di 128 dimensioni e istogramma della distribuzione degli edge di 75 dimensioni.

Invece per MIRFLICKR scegliamo di utilizzare una combinazione di features globali e locali. Per ognuna di esse è utilizzata la distanza che è stata dimostrata più adatta. Si tratta in totale di 11 features diverse:

- Istogrammi di 16 bin per canale per le rappresentazioni RGB, HSV, LAB. Calcoliamo gli istogrammi utilizzando una forma di spazialità dividendo l'immagine in 3 parti in verticale e, separatamente 3 parti in orizzontale. Concateniamo le features delle due divisioni ottenendo 2 features per rappresentazione di $16 \times 3 = 48$ dimensioni. In totale abbiamo così $2 \times 3 = 6$ features basate sugli istogrammi. Per le features basate sulle rappresentazioni RGB e HSV usiamo la distanza L_1, mentre per la rappresentazione LAB usiamo la KL-divergence simmetrica usando la media.

- GIST da 1024 dimensioni. Usiamo la distanza L_2.

- SIFT e HueSIFT sia sparso con Harris Laplacian e denso a griglia. Ogni feature è successivamente trattata nel modo classico della bag of words per realizzare un istogramma di 16 bin. Si valuta la distanza tra gli istogrammi delle immagini utilizzando la distanza χ^2.

Decise quindi le metriche per entrambi i dataset, procediamo al calcolo delle similarità tra ogni coppia di immagini ed al calcolo delle matrici di similarità utilizzate nel metodo RankingDecomposition. L'operazione realizza una matrice per ogni dataset di dimensioni pari al numero di immagini in entrambe le dimensioni. Si tratta di due matrici di dimensioni enormi che hanno richiesto uno sforzo computazionale considerevole.

Tutte le tecniche sperimentate sono state scritte principalmente in linguaggi Python e MATLAB. Le prove sono state eseguite su un modesto cluster di 5 macchine dual core con 8 Gb di RAM l'una.

6.2 Framework di lavoro

Per testare le differenze di performance delle tecniche, fissiamo un framework di test e sostituiamo all'interno solamente le tecniche di raffinamento scelto. La sequenza di operazioni considerata allo stato dell'arte prevede tre operazioni:

1. Una fase di **filtraggio dei tag**. I tag vengono selezionati e rimossi secondo una qualche metodologia: ad esempio matching su un'ontologia, tag con solo lettere alfabetiche, rimozione delle stopword. Considerando che sono presenti molti più strumenti ed informazioni in lingua inglese, potrebbe essere conveniente la traduzione delle parole in lingua inglese. I tag non vengono arricchiti: esperimenti precedenti indicano che l'arricchimento in questa fase aumenta il rumore presente.

2. Una fase di **raffinamento** dove viene impiegata la tecnica scelta.

3. Una fase di **arricchimento** dove sono aggiunti termini utilizzando sinonimi o termini di altre lingue. Ovviamente i termini aggiunti devono essere sicuri, in modo da non aumentare il rumore.

Vista la natura della valutazione, oltre alla fase del raffinamento, consideriamo di utilizzare soltanto la fase di filtraggio ed evitare l'arricchimento. Per il filtraggio valutiamo l'uso di diverse tecniche:

1. **Tag originali inalterati**: i tag non sono filtrati

2. **Matching su WordNet**: i tag sono compattati morfologicamente e sono filtrati in base alla presenza o assenza in WordNet.

3. **Rimozione delle stopword**: sono rimossi i tag che corrispondono a parole comuni tipo *and*, *or*, *my*, e così via.

4. **Traduzione e matching su WordNet**: i tag sono tradotti in lingua inglese, compattati morfologicamente e infine filtrati in base alla presenza o assenza in WordNet.

5. **Divisione e matching su WordNet**: i tag sono confrontati in WordNet. Se non sono presenti viene tentato di dividere la parole in più di una. Questo serve a recuperare tag tipo *thegreatwallofchina* talvolta presenti.

6. **Divisione dei tag con caratteri speciali**: sono divisi i tag contenenti simboli speciali tipo *my_cat*, *old−elephant* e così via.

Eseguiamo un esperimento su MIRFLICKR per valutare le tecniche: valutiamo una delle tecniche o una combinazione di esse. Utilizziamo la F_1 per ogni concetto e la macro-average F_1 per il sistema complessivo. Considerando che la ground truth di MIRFLICKR ha tutti i termini in WordNet, decidiamo che ogni prova avrà come azione finale di filtraggio il matching su WordNet: le altre tecniche vengono applicate in sequenza ed infine il matching. Per le combinazioni tra le prove, è importante l'ordine in cui sono eseguite. Le tecniche di filtraggio testate sono:

1. Matching su WordNet
2. Traduzione delle parole in inglese (utilizzando Bing Translator[1] e MyMemory[2]), matching su WordNet
3. Divisione parole con o senza caratteri speciali, matching su WordNet
4. Traduzione delle parole in inglese, divisione parole con o senza caratteri speciali, matching su WordNet
5. Traduzione delle parole in inglese, divisione parole senza caratteri speciali, matching su WordNet

Per quanto riguarda il calcolo dei positivi nella formula dell'F_1 usiamo un confronto diretto del tag con il termine della ground truth, includendo i termini più specifici nel termine generale. In tabella 6.1 sono indicati i termini aggiuntivi. Uno dei problemi per il confronto dei risultati con altri lavori, come ad esempio [ZYM10], è che molto spesso sono impiegate regole simili ma non sono menzionate. I risultati sintetici delle prove sono riportati in tabella 6.2. Osservando i risultati possiamo notare che i cambiamenti in termine di F_1 sono quasi trascurabili, probabilmente perché si tratta soltanto di 18 classi e questo genere di tag non sono molto influenzati dalle operazioni scelte. Tuttavia si nota che tutte le operazioni hanno un'impatto sulla precision: tutte queste tecniche introducono sicuramente rumore. Tutte le tecniche aumentano di 1-2 punti la recall, aggiungendo probabilmente alcuni tag significativi

[1]Bing Translator. `http://www.bing.com/translator/`

[2]MyMemory Translated.net. `http://mymemory.translated.net`

Tag presente	Tag aggiuntivo predetto
bird, dog	*animal*
birds	*bird, animal*
clouds	*cloud, sky*
sea, ocean, river, lake	*water*
portrait, boy, girl, woman, baby, man	*people*
plants, tree, trees, flower, flowers	*plant*

Tabella 6.1. Le predizioni sulla base dei tag originali in MIRFLICKR.

Tecnica	Precision	Recall	F_1
SoloWordNet	**0,4975**	0,3124	0,3281
Traduzione	0,4932	0,3201	**0,3303**
DivCompleta	0,4487	0,3338	0,3225
DivCompleta + Traduzione	0,4501	**0,3391**	0,3266
DivParole + Traduzione	0,4606	0,3335	0,3272

Tabella 6.2. I risultati delle tecniche di filtraggio su MIRFLICKR.

ai fini del filtraggio. Tuttavia tutte queste tecniche causano un significativo calo della precision e quindi abbassando di fatto la F_1. Considerando che è più importante la precision rispetto alla recall quando si effettuano filtraggi, possiamo sostenere che queste tecniche sono tutte peggiorative.

Ai fini del framework di lavoro, decidiamo di non utilizzare alcuna tecnica di filtraggio. In alternativa filtriamo tutti i tag che non fanno parte della ground truth (o di un tag della tabella di matching 6.1).

Utilizziamo i tag degli utenti come *baseline* di confronto, per misurare se effettivamente vi è un miglioramento delle prestazioni utilizzando i metodi scelti. Questo pone un problema aggiuntivo: i tag degli utenti sono variabili in base all'immagine. Le tecniche scelte non hanno di per se meccanismi per scegliere accuratamente il numero di tag da predire in ogni singola immagine. Nel caso in cui scegliamo di predire meno tag rispetto a quanti ne sono presenti su una singola immagine, prendiamo i primi n tag secondo l'ordine naturale recuperato da Flickr.

Decidiamo infine come valutare le etichette positive per NUSWIDE: oltre all'ovvio confronto diretto tra etichetta predetta ed etichetta della ground truth includiamo i plurali delle etichette. Ogni termine plurale è reso equivalente al termine singolare.

Misuriamo la precision, la recall e la F_1 *micro-average* per valutare le performance globali del sistema.

6.3 TagSimilar

Proviamo la tecnica TagSimilar su entrambi i dataset. L'individuazione dei vicini visuali è fatta utilizzando le matrici di similarità calcolate precedentemente. Si filtrano tutti i tag tranne quelli presenti nella groud truth. L'unico parametro di questa tecnica è il numero di vicini da utilizzare.

Nella prima prova, eseguiamo la tecnica su MIRFLICKR osservando la F_1 in relazione al numero di vicini scelti. I valori di F_1 variano in base al numero di tag predetti. I risultati sono visibili in tabella 6.3 e in figura 6.1.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0.1048**	0,1263	0,1125	0,0854	0,0521
TagSimilar (K = 1)	0,0102	0,0092	0,0114	0,0108	0,0066
TagSimilar (K = 2)	0,0421	0,0432	0,0489	0,0545	0,0538
TagSimilar (K = 10)	0,0862	0,1077	0,1256	0,1168	0,0735
TagSimilar (K = 50)	0,0851	0,1392	0,1776	0,2318	0,2365
TagSimilar (K = 200)	0,0990	0,1695	0,2150	0,2510	0,2458
TagSimilar (K = 500)	0,0950	**0,1795**	**0,2236**	**0,2609**	**0,2491**

Tabella 6.3. I risultati di TagSimilar su MIRFLICKR.

Osservando i risultati si vede che un numero di vicini basso non permette alla tecnica di migliorare i tag originali. Sono necessari almeno 50 vicini per poter produrre un risultato superiore ai tag originali. Inoltre, se si utilizza solo 1 tag, è conveniente prenderne uno inserito dagli utenti rispetto ad utilizzare questa tecnica.

Proviamo ad eseguire la tecnica anche su NUSWIDE con le stesse modalità. I risultati sono riportati in tabella 6.4 e in figura 6.2. A differenza

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0,2543**	**0,2916**	**0,2734**	0,2153	0,1291
TagSimilar (K = 1)	0,1068	0,1221	0,1154	0,0940	0,0574
TagSimilar (K = 2)	0,1073	0,1241	0,1262	0,1328	0,1166
TagSimilar (K = 10)	0,1488	0,1916	0,2097	0,2124	0,1695
TagSimilar (K = 50)	0,1603	0,1996	0,2217	0,2285	0,1958
TagSimilar (K = 200)	0,1644	0,1999	0,2231	0,2314	**0,1984**
TagSimilar (K = 500)	0,1653	0,1983	0,2228	**0,2320**	0,1974
TagSimilar (K = 1000)	0,1659	0,1980	0,2218	0,2304	0,1959

Tabella 6.4. I risultati di TagSimilar su NUSWIDE.

dei risultati su MIRFLICKR, la tecnica fallisce di apportare miglioramenti significativi ai tag originali. Soltanto utilizzando 200 o 500 vicini abbiano un miglioramento e solo se si predicono almeno 10 tag. Questo è un esempio di quanto affermato in [DBLFF10]: tecniche che funzionano con pochi esempi e poche categorie potrebbero non funzionare quando si provano su un numero maggiore di esempi e tante categorie. Notiamo anche che i risultati dei tag originali sono più alti rispetto a MIRFLICKR. Riportiamo in tabella 6.5 e 6.6 rispettivamente i risultati della precision e i risultati della recall delle migliori prove a confronto. Le etichette della ground truth di MIRFLICKR sono più generiche e probabilmente più abusate. Invece la precision dei tag originali in NUSWIDE è molto più alta: nonostante abbia più categorie, sono utilizzate più correttamente dagli utenti. Come era logico aspettarsi, all'aumentare

Tecnica	P 1 tag	P 2 tag	P 3 tag	P 5 tag	P 10 tag
Tag Originali MIRFLICKR	**0,1983**	0,1510	0,1084	0,0665	0,0333
TagSimilar MIRFLICKR	0,1796	**0.2147**	**0.2156**	**0.2031**	**0.1592**
Tag Originali NUSWIDE	**0,3929**	**0.2982**	**0,2319**	0,1527	0,0780
TagSimilar NUSWIDE	0,2554	0,2028	0,1890	**0,1645**	**0,1193**

Tabella 6.5. La precision di TagSimilar sui due dataset a confronto.

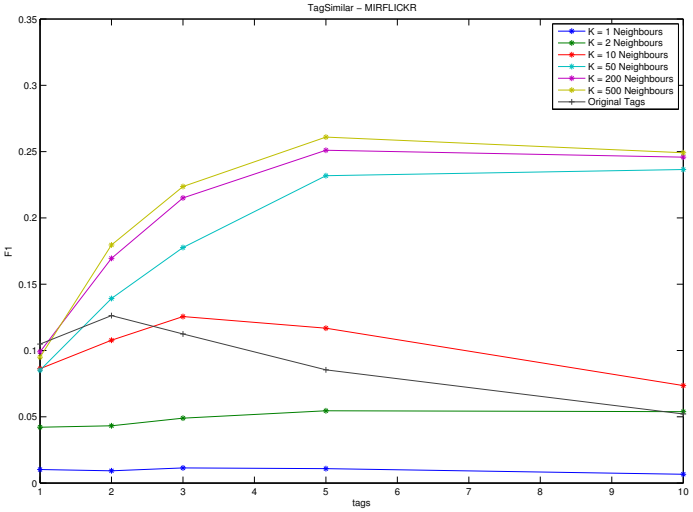

Figura 6.1. I risultati di TagSimilar su MIRFLICKR al variare del numero di tag predetti e al variare del numero di vicini considerati.

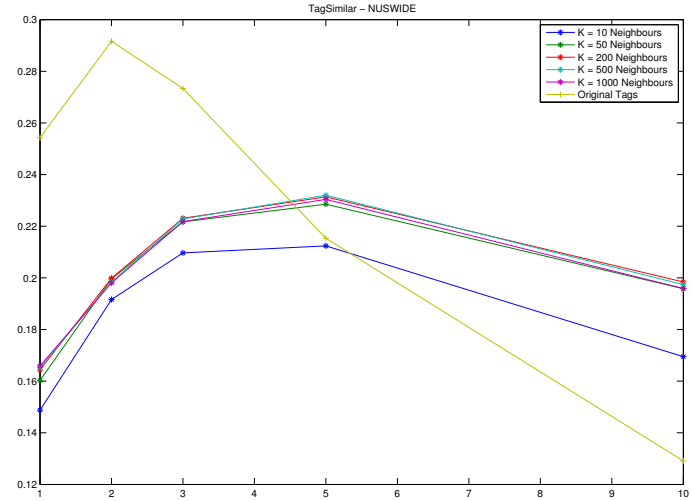

Figura 6.2. I risultati di TagSimilar su NUSWIDE al variare del numero di tag predetti e al variare del numero di vicini considerati.

Tecnica	R 1 tag	R 2 tag	R 3 tag	R 5 tag	R 10 tag
Tag Originali MIRFLICKR	**0,1983**	0,1510	0,1084	0,0665	0,0333
TagSimilar MIRFLICKR	0,1796	**0.2147**	**0.2156**	**0.2031**	**0.1592**
Tag Originali NUSWIDE	**0,1880**	**0,2853**	**0,3329**	0,3652	0,3734
TagSimilar NUSWIDE	0,1222	0,1941	0,2713	**0,3936**	**0,5709**

Tabella 6.6. La recall di TagSimilar sui due dataset a confronto.

del numero di tag predetti (sia per gli utenti sia per la tecnica esaminata) la precision tende a diminuire mentre la recall tende ad aumentare. L'azione di trasferimento delle etichette è particolarmente evidente grazie all'aumento notevole della recall.

6.4 TagRelevance

TagRelevance ha una struttura di funzionamento molto simile a TagSimilar. Anche in questo caso l'unico parametro della tecnica è il numero di vicini da utilizzare. Si filtrano tutti i tag tranne quelli presenti nella groud truth. L'individuazione dei vicini visuali è fatta utilizzando le matrici di similarità calcolate precedentemente. Dopo aver calcolato la distribuzione *Prior* utilizzando la formula specificata nella sezione 5.2.2, eseguiamo la prova con le stesse configurazioni usate su TagSimilar. Osservando i risultati vediamo

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0.1048**	0,1263	0,1125	0,0854	0,0521
TagRelevance (K = 50)	0,0686	0,1264	0,1626	0,2006	0,2397
TagRelevance (K = 200)	0,0957	0,1758	0,2225	0,2616	0,2423
TagRelevance (K = 500)	0,0911	0,1869	0,2283	0,2757	0,2602
TagRelevance (K = 1000)	0,0893	**0,1896**	**0,2436**	**0,2839**	**0,2703**

Tabella 6.7. I risultati di TagRelevance su MIRFLICKR.

che TagRelevance ha successo in entrambi i dataset. Notiamo in particolare un andamento dapprima crescente all'aumentare dei tag ed infine decrescente. Si tratta dello stesso andamento che era presente in TagSimilar. Il mo-

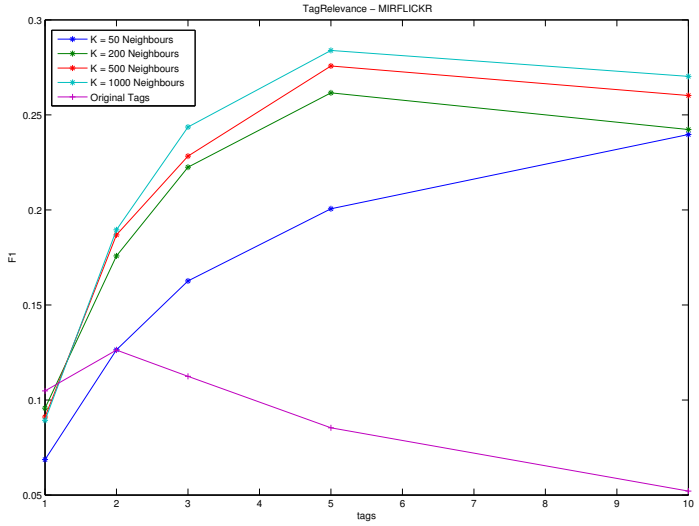

Figura 6.3. I risultati di TagRelevance su MIRFLICKR al variare del numero di tag predetti e al variare del numero di vicini considerati.

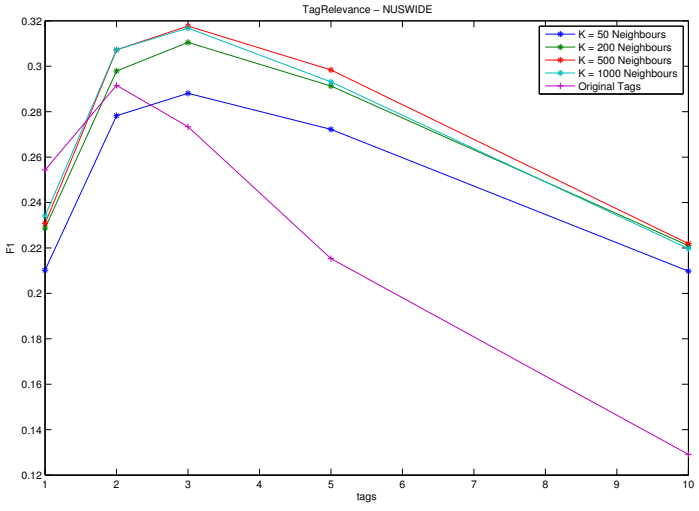

Figura 6.4. I risultati di TagRelevance su NUSWIDE al variare del numero di tag predetti e al variare del numero di vicini considerati.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0,2543**	0,2916	0,2734	0,2153	0,1291
TagRelevance (K = 50)	0,2102	0,2782	0,2881	0,2722	0,2098
TagRelevance (K = 200)	0,2287	0,2979	0,3105	0,2913	0,2210
TagRelevance (K = 500)	0,2308	0,3072	**0,3177**	**0,2984**	**0,2219**
TagRelevance (K = 1000)	0,2341	**0,3073**	0,3168	0,2931	0,2197

Tabella 6.8. I risultati di TagRelevance su NUSWIDE.

tivo è da ricercare nel numero di tag richiesti per esprimere correttamente un'immagine, in questo caso il riferimento è alla groundtruth. Risulta infatti che NUSWIDE ha una media di etichette per immagine inferiore rispetto a MIRFLICKR.

Sperimentiamo anche la tecnica che abbiamo denominato TagRelevancePlus nella sezione 5.2.2, ovvero prendiamo i tag degli utenti e utilizziamo TagRelevance per derivare i tag mancanti. Anche qui usiamo la stessa logica adottata in precedenza: valutiamo un numero di tag variabile da 1 a 10, quando i tag inseriti dagli utenti sono insufficienti rispetto a quanti richiesti usiamo TagRelevance per aggiungere i mancanti. I risultati sono riportati in tabella 6.9 e in figura 6.9 per MIRFLICKR e in tabella 6.10 e in figura 6.6 per NUSWIDE. Osservando i risultati si notano prestazioni notevolmente

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0.1048	0,1263	0,1125	0,0854	0,0521
TagRelevancePlus (K = 50)	0,1875	0,2257	0,2401	0,2506	0,2568
TagRelevancePlus (K = 200)	**0,2077**	0,2672	0,2909	0,3020	0,2586
TagRelevancePlus (K = 500)	0,2033	**0,2780**	**0,3106**	**0,3215**	**0,2830**
TagRelevancePlus (K = 1000)	0,2046	0,2764	0,2980	0,3155	0,2748

Tabella 6.9. I risultati di TagRelevancePlus su MIRFLICKR.

superiori rispetto al caso precedente. I tag degli utenti immessi nell'immagine sotto test (una volta filtrati) sono generalmente più corretti di quelli che potrebbero essere inseriti utilizzando la tecnica. Le prestazioni migliori

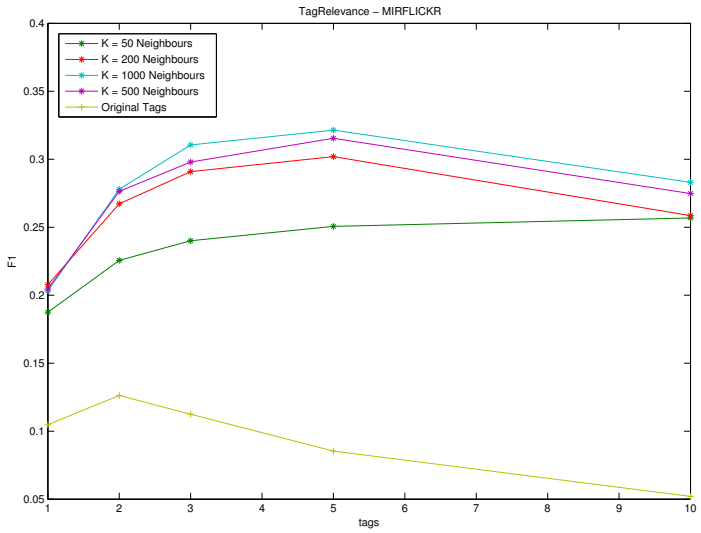

Figura 6.5. I risultati di TagRelevancePlus su MIRFLICKR al variare del numero di tag predetti e al variare del numero di vicini considerati.

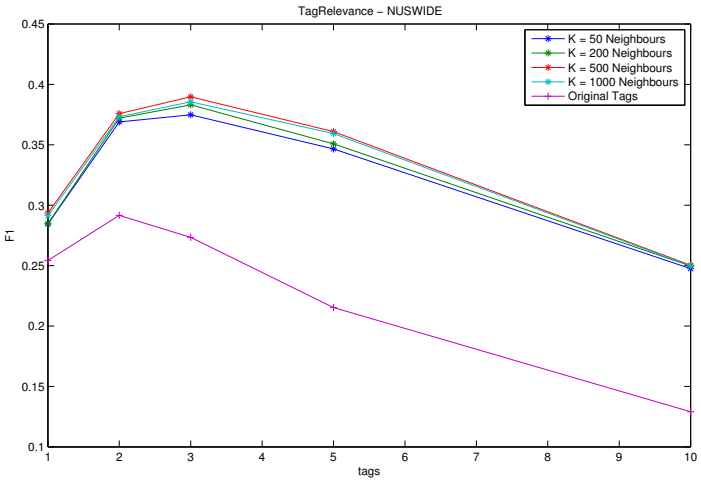

Figura 6.6. I risultati di TagRelevancePlus su NUSWIDE al variare del numero di tag predetti e al variare del numero di vicini considerati.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0,2543	0,2916	0,2734	0,2153	0,1291
TagRelevance (K = 50)	0,2845	0,3689	0,3748	0,3466	0,2475
TagRelevance (K = 200)	0,2847	0,3721	0,3831	0,3508	0,2497
TagRelevance (K = 500)	**0,2936**	**0,3758**	**0,3897**	**0,3609**	**0,2503**
TagRelevance (K = 1000)	0,2913	0,3732	0,3856	0,3593	0,2498

Tabella 6.10. I risultati di TagRelevancePlus su NUSWIDE.

suggeriscono che è meglio conservare i tag esistenti e, in caso di necessità, completare le mancanze.

6.5 RankingDecomposition

Per quanto riguarda la tecnica RankingDecomposition, è più difficile riprodurre risultati simili a quanto riportato in letteratura. I parametri da gestire sono molti di più, il codice sorgente non è disponibile e non sono specificati gli eventuali filtri eseguiti.

I primi parametri da individuare sono le dimensioni delle componenti latenti. Per ogni modo (utenti, immagini e tag) è necessario scegliere un numero intero r_u, r_i, r_t del numero di componenti da usare. Una tecnica consigliata in letteratura consiste nell'eseguire la matricizzazione del tensore originale secondo un modo alla volta e di calcolarne la decomposizione SVD. Il numero di componenti necessarie per ricostruire la matrice è il limite superiore da utilizzare. Successivamente è possibile selezionare un numero di componenti in base all'energia conservata utilizzando gli autovalori ed eseguire una cross validazione. Purtroppo la soluzione di usare la decomposizione SVD risulta difficoltosa a causa dell'elevata dimensione del numero di immagini ed utenti. Decidiamo quindi di fissare semplicemente i parametri a valori stabiliti empiricamente sulla base del tempo necessario ad eseguire le prove, e di osservarne gli effetti.

Proviamo variando ogni componente singolarmente ed infine variandole tutte insieme. Il numero di elementi è indicato con (r_u, r_i, r_t). Annullia-

mo l'effetto della regolarizzazione di similarità e lasciamo la regolarizzazione in norma 2 ponendo $\gamma = 0.0001$. Utilizziamo la discesa del gradiente come descritto in 5.2.3. La funzione di costo raggiunge un minimo dopo circa 300 iterazioni. Si nota che su MIRFLICKR l'utilizzo delle informazioni

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0,2510**	**0,2880**	0,2702	0,2132	0,1278
RankDec (16, 16, 16)	0,2038	0,2499	0,2552	0,2358	0,1802
RankDec (32, 32, 32)	0,2233	0,2697	0,2740	0,2503	**0,1860**
RankDec (32, 32, 64)	0,2355	0,2812	**0,2820**	**0,2524**	0,1831
RankDec (32, 64, 32)	0,2162	0,2635	0,2674	0,2452	0,1835
RankDec (64, 32, 32)	0,2317	0,2771	0,2772	0,2495	0,1841

Tabella 6.11. I risultati di RankingDecomposition al variare del numero di componenti su NUSWIDE.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0,1048	0,1263	0,1125	0,0854	0,0521
RankDec (8, 8, 8)	0,2289	0,2872	0,2992	0,3008	0,2693
RankDec (8, 8, 16)	0,2381	0,2876	0,3019	0,2982	0,2676
RankDec (8, 16, 8)	0,2275	0,2900	0,3054	0,3027	0,2670
RankDec (16, 8, 8)	0,2332	**0,2978**	**0,3148**	**0,3127**	**0,2779**
RankDec (16, 16, 16)	**0,2472**	0,2936	0,3058	0,3008	0,2659
RankDec (32, 32, 32)	0,2456	0,2948	0,3073	0,3040	0,2687

Tabella 6.12. I risultati di RankingDecomposition al variare del numero di componenti su MIRFLICKR.

relazionali è già sufficiente per avere un miglioramento dei tag esistenti. Probabilmente è dovuto allo sbilanciamento delle classi e alla misurazione micro average F_1 che tende ad essere più influenzata dalle classi più frequenti. Ciò deriva dal fatto che è sufficiente rimuovere le classi poco frequenti per ottenere un miglioramento completando il tensore. Evidenzia anche che il sistema è in under fit. Viceversa su NUSWIDE il comportamento è simile a quello dei tag originali anche se peggiorativo. Tutte le combinazioni di fattori latenti

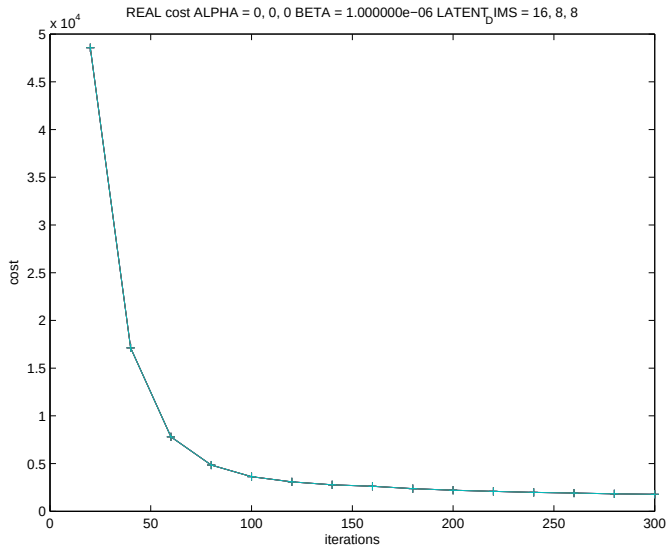

Figura 6.7. Il costo all'aumentare delle iterazioni.

scelti su NUSWIDE non hanno la possibilità di modellare l'energia completa del tensore originale.

Visti i dati e le disponibilità computazionale, decidiamo di utilizzare i parametri **(32, 32, 32)** che si comportano bene su NUSWIDE e non overfittano. Poiché la configurazione scelta è in underfit su MIRFLICKR, è sufficiente aumentare il parametro di regolarizzazione in norma 2 per bilanciarlo.

Procediamo quindi ad un processo di *model selection* variando i parametri di regolarizzazione γ, $\gamma_{\alpha,1}$, $\gamma_{\alpha,2}$, $\gamma_{\alpha,3}$. I risultati più significativi sono riportati nelle tabelle 6.14 per MIRFLICKR e in 6.13 per NUSWIDE. Dal-

Parametri RankDec	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0.2510**	**0.2880**	0.2702	0.2132	0.1278
$\gamma = 10^{-4}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0.2334	0.2824	**0.2844**	**0.2590**	**0.1920**
$\gamma = 10^{-7}, \gamma_\alpha = (10^{-5}, 10^{-6}, 10^{-5})$	0.1780	0.2276	0.2391	0.2291	0.1801
$\gamma = 10^{-4}, \gamma_\alpha = (10^{-5}, 10^{-6}, 10^{-5})$	0.1887	0.2383	0.2488	0.2355	0.1834
$\gamma = 10^{-4}, \gamma_\alpha = (10^{-6}, 10^{-7}, 10^{-6})$	0.2226	0.2737	0.2788	0.2550	0.1898
$\gamma = 10^{-6}, \gamma_\alpha = (10^{-6}, 10^{-7}, 10^{-6})$	0.2021	0.2490	0.2549	0.2356	0.1804
$\gamma = 10^{-6}, \gamma_\alpha = (0, 0, 0)$	0.2233	0.2697	0.2740	0.2503	0.1860

Tabella 6.13. Alcuni risultati al variare dei parametri di regolarizzazione in NUSWIDE.

Parametri RankDec	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0,1048	0,1263	0,1125	0,0854	0,0521
$\gamma = 0, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,2335	0,2613	0,2841	0,2633	0,2331
$\gamma = 10^{-2}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1985	**0,3084**	**0,3174**	0,2840	0,2545
$\gamma = 10^{-3}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,2387	0,2698	0,2848	0,2653	0,2331
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-3}, 10^{-4}, 10^{-3})$	**0,2564**	0,2952	0,2970	0,2859	0,2569
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0,2460	0,2888	0,2970	0,2902	0,2631
$\gamma = 0, \gamma_\alpha = (0, 0, 0)$	0,2456	0,2944	0,3075	**0,3037**	**0,2686**

Tabella 6.14. Alcuni risultati al variare dei parametri di regolarizzazione in MIRFLICKR.

l'analisi delle prove si vede che la tecnica viene maggiormente influenzata dai parametri di regolarizzazione rispetto ai parametri di dimensione dei fattori

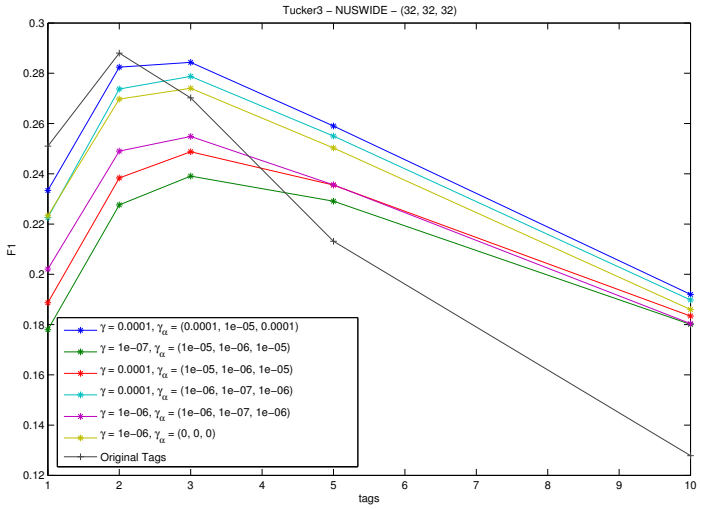

Figura 6.8. Alcuni risultati al variare dei parametri di regolarizzazione in NUSWIDE.

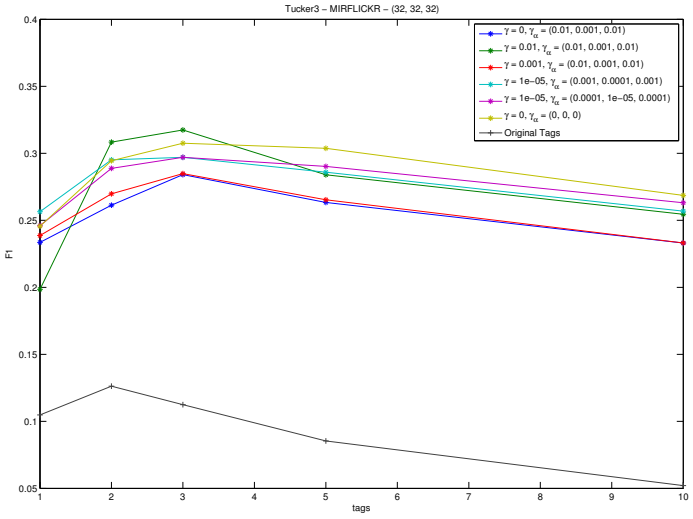

Figura 6.9. Alcuni risultati al variare dei parametri di regolarizzazione in MIRFLICKR.

latenti. Altre prove eseguite con dimensioni diversi hanno portato a risultati con piccole variazioni non significative. La tecnica, applicata a NUSWIDE, conferma l'esperienza presente in letteratura: le variazioni dei fattori latenti portano a delle performance migliorative. Tuttavia i valori ad assegnare ai parametri di regolarizzazione non sono di facile individuazione. Alcune delle prove con regolarizzazione hanno riportato performance peggiorative rispetto alle prove senza regolarizzazione. Inoltre a seconda del dataset potrebbero essere parecchio diversi, aumentando la difficoltà nell'individuare i parametri da utilizzare.

Parametri RankDec	P 1 tag	P 2 tag	P 3 tag	P 5 tag	P 10 tag
Tag Originali	0,1983	0,1510	0,1084	0,0665	0,0333
$\gamma = 0, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,4417	0,3125	0,2738	0,2050	0,1490
$\gamma = 10^{-2}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,3756	**0,3688**	**0,3059**	0,2210	0,1627
$\gamma = 10^{-3}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,4515	0,3226	0,2746	0,2065	0,1490
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-3}, 10^{-4}, 10^{-3})$	**0,4851**	0,3530	0,2863	0,2226	0,1642
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0,4654	0,3453	0,2863	0,2259	0,1682
$\gamma = 0, \gamma_\alpha = (0, 0, 0)$	0,4646	0,3521	0,2964	**0,2364**	**0,1717**

Tabella 6.15. Alcuni valori di precision al variare dei parametri di regolarizzazione in MIRFLICKR.

Parametri RankDec	P 1 tag	P 2 tag	P 3 tag	P 5 tag	P 10 tag
Tag Originali	**0,3889**	**0,2951**	0,2296	0,1513	0,0773
$\gamma = 0, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,3616	0,2894	**0,2417**	**0,1839**	**0,1162**
$\gamma = 10^{-2}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,2758	0,2333	0,2032	0,1626	0,1089
$\gamma = 10^{-3}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,2924	0,2442	0,2114	0,1672	0,1109
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-3}, 10^{-4}, 10^{-3})$	0,3449	0,2805	0,2369	0,1810	0,1148
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0,3131	0,2552	0,2166	0,1672	0,1091
$\gamma = 0, \gamma_\alpha = (0, 0, 0)$	0,3460	0,2764	0,2329	0,1777	0,1125

Tabella 6.16. Alcuni valori di precision al variare dei parametri di regolarizzazione in NUSWIDE.

MIRFLICKR invece ha un comportamento diverso, in particolare le prove con la regolarizzazione non hanno portato generalmente a risultati migliori

rispetto al loro non utilizzo. In parte è dovuto al fatto che le classi sono sbilanciate e che rispetto a NUSWIDE ci sono molti meno tag assegnati dagli utenti e disponibili per realizzare un buon modello. Solo in un caso (riportato in tabella e nei grafici) si è avuto un miglioramento significativo con la predizione di 3 tag.

Parametri RankDec	R 1 tag	R 2 tag	R 3 tag	R 5 tag	R 10 tag
Tag Originali	0,0712	0,1085	0,1168	0,1194	0,1196
$\gamma = 0, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1587	0,2245	0,2951	0,3682	0,5352
$\gamma = 10^{-2}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1349	**0,2650**	**0,3298**	0,3970	0,5845
$\gamma = 10^{-3}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1622	0,2318	0,2959	0,3710	0,5353
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-3}, 10^{-4}, 10^{-3})$	**0,1743**	0,2536	0,3086	0,3998	0,5899
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0,1672	0,2482	0,3086	0,4058	0,6043
$\gamma = 0, \gamma_\alpha = (0, 0, 0)$	0,1669	0,2530	0,3195	**0,4247**	**0,6169**

Tabella 6.17. Alcuni valori di recall al variare dei parametri di regolarizzazione in MIRFLICKR.

Parametri RankDec	R 1 tag	R 2 tag	R 3 tag	R 5 tag	R 10 tag
Tag Originali	**0,1853**	**0,2812**	0,3282	0,3605	0,3685
$\gamma = 0, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1723	0,2758	**0,3454**	**0,4380**	**0,5534**
$\gamma = 10^{-2}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1314	0,2223	0,2904	0,3874	0,5190
$\gamma = 10^{-3}, \gamma_\alpha = (10^{-2}, 10^{-3}, 10^{-2})$	0,1393	0,2327	0,3022	0,3983	0,5285
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-3}, 10^{-4}, 10^{-3})$	0,1643	0,2673	0,3386	0,4312	0,5471
$\gamma = 10^{-5}, \gamma_\alpha = (10^{-4}, 10^{-5}, 10^{-4})$	0,1492	0,2431	0,3096	0,3984	0,5199
$\gamma = 0, \gamma_\alpha = (0, 0, 0)$	0,1648	0,2634	0,3328	0,4232	0,5360

Tabella 6.18. Alcuni valori di recall al variare dei parametri di regolarizzazione in NUSWIDE.

6.6 TagRelevanceRTT

Utilizziamo le stesse modalità scelte per TagRelevance standard. Per i dati temporali utilizziamo le probabilità recuperate dai tag del dataset NUSWIDE

in quanto più accurati avendo a disposizione un numero maggiore di immagini. Si potrebbe modellare la probabilità utilizzando tecniche classiche, quali l'analisi delle serie temporali di Box Jenkins, ed ottenere una modellizzazione dell'andamento temporale per fare predizioni sui dati mancanti. Visto che l'arco temporale delle immagini di NUSWIDE contiene interamente l'arco temporale di MIRFLICKR optiamo per utilizzare direttamente i valori frequentisti ottenuti per lo studio temporale in sezione 5.3.1. Utilizziamo i dati con granularità settimanale. I risultati sono riportati rispettivamente nelle tabelle 6.19 e 6.20 per MIRFLICKR e NUSWIDE. I dati temporali

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0.1048	0,1263	0,1125	0,0854	0,0521
TagRelevanceRTT (K = 50)	0,1028	0,1525	0,1716	0,2069	**0,3080**
TagRelevanceRTT (K = 200)	0,1173	0,1679	0,2087	0,2553	0,2427
TagRelevanceRTT (K = 500)	0,1293	0,1702	**0,2156**	0,2666	0,2588
TagRelevanceRTT (K = 1000)	**0,1431**	**0,1712**	0,2103	**0,2777**	0,2719

Tabella 6.19. I risultati di TagRelevanceRTT su MIRFLICKR.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	**0,2543**	**0,2916**	0,2734	0,2153	0,1291
TagRelevanceRTT (K = 50)	0,2184	0,2876	**0,3089**	**0,2724**	**0,2023**
TagRelevanceRTT (K = 200)	0,2024	0,2798	0,2999	0,2717	0,2020
TagRelevanceRTT (K = 500)	0,1982	0,2591	0,2837	0,2592	0,1976

Tabella 6.20. I risultati di TagRelevanceRTT su NUSWIDE.

utilizzati in questo modo hanno l'effetto di peggiorare le prestazioni su entrambi i dataset. La tecnica tende a predire più spesso i soliti tag per tutte le immagini. Un modo per risolvere il problema potrebbe essere quello di modulare l'importanza del fattore temporale diminuendone gli effetti nella selezione delle immagini. Inoltre sarebbe interessante valutare l'andamento delle serie temporali negli esempi classificati non correttamente per verificare la presenza delle strutture temporali viste in precedenza.

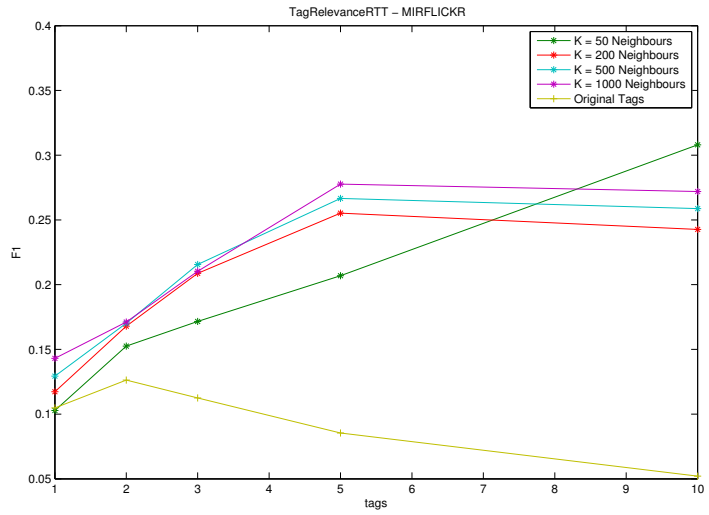

Figura 6.10. I risultati di TagRelevanceRTT su MIRFLICKR al variare del numero di tag predetti e al variare del numero di vicini considerati.

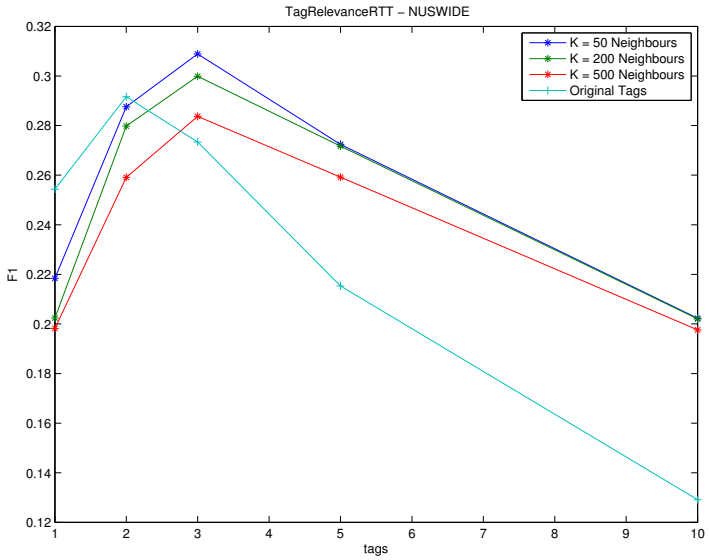

Figura 6.11. I risultati di TagRelevanceRTT su NUSWIDE al variare del numero di tag predetti e al variare del numero di vicini considerati.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali MIRFLICKR	0,1048	0,1263	0,1125	0,0854	0,0521
TagRelevance MIRFLICKR	**0,2033**	**0,2780**	**0,3106**	**0,3215**	**0,2830**
TagRelevanceRTT MIRFLICKR	0,1431	0,1712	0,2103	0,2777	0,2719
Tag Originali NUSWIDE	**0,2510**	0,2880	0,2702	0,2132	0,1278
TagRelevance NUSWIDE	0,2308	**0,3072**	**0,3177**	**0,2984**	**0,2219**
TagRelevanceRTT NUSWIDE	0,2184	0,2876	0,3089	0,2724	0,2023

Tabella 6.21. Confronto di prestazioni tra TagRelevance e TagRelevanceRTT.

6.7 Comparazione finale delle tecniche

Prendiamo i risultati migliori dei tre metodi e confrontiamo i valori ottenuti su entrambi i dataset. I risultati su MIRFLICKR sono riportati in tabella 6.22 e in figura 6.12; n tabella 6.23 e in figura 6.13 i valori ottenuti su NUSWIDE. Nel caso di MIRFLICKR osserviamo che la tecnica Ranking-

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0,1048	0,1263	0,1125	0,0854	0,0521
TagSimilar	0,0950	0,1795	0,2236	0,2609	0,2491
TagRelevance	0,0893	0,1896	0,2436	0,2839	0,2703
TagRelevancePlus	0,2033	0,2780	**0,3106**	**0,3215**	**0,2830**
RankDec	**0,2456**	**0,2944**	0,3075	0,3037	0,2686

Tabella 6.22. Comparazione finale delle tecniche su MIRFLICKR.

Decomposition ottiene la miglior performance tranne nel caso di predizione di 10 tag. TagRelevance riesce ad avere performance migliori soltanto nella versione TagRelevancePlus e solo nel caso di 3 o più tag predetti. Nel caso di NUSWIDE vediamo che il miglior risultato è quello della tecnica TagRelevance. Tutti e tre i metodi non riescono a superare le performance dei tag originali se ci limitiamo ad usare soltanto un tag. Ciò è verificato anche dal vantaggio di TagRelevancePlus che non scarta i tag immessi dagli utenti. Solo nel caso in cui i tag richiesti siano maggiori di 3 abbiamo un vantaggio per tutte e tre le tecniche, tuttavia il motivo principale è la mancanza di tag

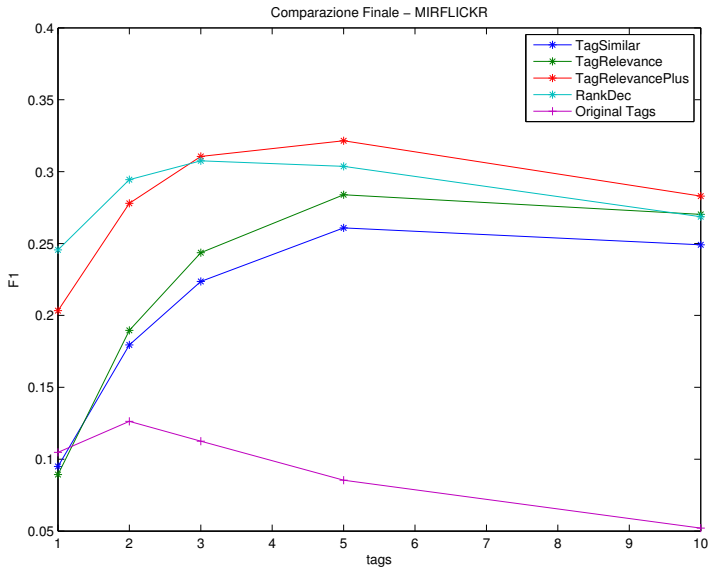

Figura 6.12. Comparazione finale delle tecniche su MIRFLICKR.

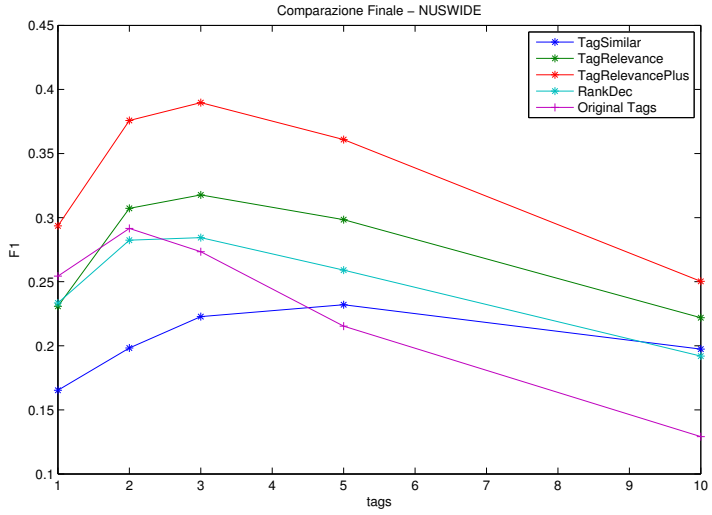

Figura 6.13. Comparazione finale delle tecniche su NUSWIDE.

Tecnica	F_1 1 tag	F_1 2 tag	F_1 3 tag	F_1 5 tag	F_1 10 tag
Tag Originali	0,2510	0,2880	0,2702	0,2132	0,1278
TagSimilar	0,1653	0,1983	0,2228	0,2320	0,1974
TagRelevance	0,2308	0,3072	0,3177	0,2984	0,2219
TagRelevancePlus	**0,2936**	**0,3758**	**0,3897**	**0,3609**	**0,2503**
RankDec	0.2334	0.2824	0.2844	0.2590	0.1920

Tabella 6.23. Comparazione finale delle tecniche su NUSWIDE.

immessi dagli utenti per soddisfare la richiesta su ogni immagine. La tecnica TagRelevance è l'unica che riesce ad ottenere un punteggio superiore ai tag originali in quasi tutti i casi.

7

Conclusioni

In questo lavoro è stato affrontato il tema del tag refinement applicato a due dataset standard denominati MIRFLICKR e NUSWIDE. Il problema principale individuato durante lo studio è la difficoltà di comparazione tra diversi lavori in quanto è possibile individuare numerosi tipi di filtraggi e di valutazioni che non permettono di ripetere facilmente gli esperimenti. Si è cercato di stabilire un framework di lavoro, il più dettagliato possibile, in modo da costituire un possibile standard per il testing di nuove tecniche. Il framework costruito è stato testato su tre diverse tecniche in letteratura che hanno evidenziato comportamenti diversi in base alla dimensione del dataset e alla sparsità delle classi. In particolare, nei due dataset, si è evidenziata l'importanza dei tag utente che (una volta filtrati) possono essere considerati generalmente corretti. Alla base delle tecniche analizzate vi è una discreta quantità di strumenti che sono stati approfonditi e descritti nel dettaglio come ad esempio le tecniche di descrizione delle immagini con i descrittori globali e locali o le decomposizioni di matrici e la loro estensione ai tensori. È stato analizzato l'andamento temporale dei tag utente nei dataset disponibili, evidenziando fenomeni tipici delle serie temporali come la stagionalità, il trend e i comportamenti randomici. L'analisi è proseguita evidenziando le correlazioni tra gli andamenti temporali delle etichette della ground truth con i tag utente ed infine la correlazione tra i tag, le notizie giornalistiche e le ricerche sul motore di ricerca Google. Tutto questo ha portato a formulare

una tecnica basata su TagRelevance di Li et al. [LSW08] che però non ha portato i benefici sperati. I segnali temporali individuati sono osservabili facilmente: ulteriori studi sull'andamento temporale dei tag non classificati correttamente potrebbero individuare altre correlazioni utili a migliorare la tecnica.

Bibliografia

[Ahn06] Luis von Ahn. Games with a purpose. *Computer*, 39(6):92–94, June 2006.

[AY09] Evrim Acar and Bulent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21:6–20, 2009.

[BBDB+10] Lamberto Ballan, Marco Bertini, Alberto Del Bimbo, Marco Meoni, and Giuseppe Serra. Tag suggestion and localization in user-generated videos based on social knowledge. In *Proceedings of second ACM SIGMM workshop on Social media*, WSM '10, pages 3–8, New York, NY, USA, 2010. ACM.

[BBZ08] Léon Bottou, Olivier Bousquet, and Google Zürich. O.: The tradeoffs of large scale learning. In *In: Advances in Neural Information Processing Systems 20*, pages 161–168, 2008.

[BLRF11] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2011.

[BP11] Teresa Bracamonte and Barbara Poblete. Automatic image tagging through information propagation in a query log based graph structure. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1201–1204, New York, NY, USA, 2011. ACM.

[BTVG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006. 10.1007/1174402332.

[CKT+10] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popovic, and Foldit players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010.

[CTH+09] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[DBLFF10] Jia Deng, Alexander Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-15555-06.

[DLJ10] Chris H. Q. Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):45–55, January 2010.

[DLPP06] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 126–135, New York, NY, USA, 2006. ACM.

[FCH+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[Ff05] Li Fei-fei. A bayesian hierarchical model for learning natural scene categories. In *In CVPR*, pages 524–531, 2005.

[FK79] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

[FLX10] Songhe Feng, Congyan Lang, and De Xu. Beyond tag relevance: integrating visual attention model and multi-instance learning for tag saliency ranking. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 288–295, New York, NY, USA, 2010. ACM.

[GMVS09] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *In ICCV*, 2009.

[Har70] Richard Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16, 1970.

[HL08] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.

[Hoy04] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, December 2004.

[HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[JB08] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1877–1890, November 2008.

[JLM03] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 119–126, New York, NY, USA, 2003. ACM.

[JQ07] Sen Jia and Yuntao Qian. A complexity constrained nonnegative matrix factorization for hyperspectral unmixing. In *Proceedings of the 7th international conference on Independent component analysis and signal separation*, ICA'07, pages 268–276, Berlin, Heidelberg, 2007. Springer-Verlag.

[KP07] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, June 2007.

[KXT10] Gunhee Kim, Eric P. Xing, and Antonio Torralba. Modeling and analysis of dynamic behaviors of web image collections. In *Proceedings of the 11th European conference on Computer vision: Part V*, ECCV'10, pages 85–98, Berlin, Heidelberg, 2010. Springer-Verlag.

[KZWS07] Xiangzhen Kong, Chunhou Zheng, Yuqiang Wu, and Li Shang. Molecular cancer class discovery using non-negative matrix factorization with sparseness constraint. In *Proceedings of the intelligent computing 3rd international conference on Advanced*

intelligent computing theories and applications, ICIC'07, pages 792–802, Berlin, Heidelberg, 2007. Springer-Verlag.

[LHWZ10] Dong Liu, Xian-Sheng Hua, Meng Wang, and Hong-Jiang Zhang. Image retagging. In *Proceedings of the international conference on Multimedia*, MM '10, pages 491–500, New York, NY, USA, 2010. ACM.

[LHY+09] Dong Liu, Xian-Sheng Hua, Linjun Yang, Meng Wang, and Hong-Jiang Zhang. Tag ranking. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 351–360, New York, NY, USA, 2009. ACM.

[LHZ11] Dong Liu, Xian-Sheng Hua, and Hong-Jiang Zhang. Content-based tag processing for internet social images. *Multimedia Tools and Applications*, 51:723–738, 2011. 10.1007/s11042-010-0647-3.

[Lin98] Tony Lindeberg. Feature detection with automatic scale selection. *Int. J. Comput. Vision*, 30(2):79–116, November 1998.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[LS00] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

[LS09] Xirong Li and Cees G.M. Snoek. Visual categorization with negative examples for free. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 661–664, New York, NY, USA, 2009. ACM.

[LSW08] Xirong Li, Cees G.M. Snoek, and Marcel Worring. Learning tag relevance by neighbor voting for social image retrieval. In *Proceedings of the 1st ACM international conference on Multimedia*

information retrieval, MIR '08, pages 180–187, New York, NY, USA, 2008. ACM.

[LSW10] Xirong Li, Cees G. M. Snoek, and Marcel Worring. Unsupervised multi-feature tag relevance learning for social image retrieval. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 10–17, New York, NY, USA, 2010. ACM.

[MJHL10] B. Thomee Mark J. Huiskes and Michael S. Lew. New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. In *MIR '10: Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, New York, NY, USA, 2010. ACM.

[MPK10] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for image annotation. *Int. J. Comput. Vision*, 90(1):88–105, October 2010.

[MTS+05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, November 2005.

[MY09] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM J. Img. Sci.*, 2(2):438–469, April 2009.

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML'11*, pages 809–816, 2011.

[OOG05] Arzucan Özgür, Levent Özgür, and Tunga Güngör. Text categorization with class-based and corpus-based keyword selection. In *Proceedings of the 20th international conference on Computer and Information Sciences*, ISCIS'05, pages 606–615, Berlin, Heidelberg, 2005. Springer-Verlag.

[OT06] Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006.

[PBMB10] Barbara Poblete, Benjamin Bustos, Marcelo Mendoza, and Juan Manuel Barrios. Visual-semantic graphs: using queries to reduce the semantic gap in web image retrieval. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1553–1556, New York, NY, USA, 2010. ACM.

[RMNSt09] Steffen Rendle, Ro Balby Marinho, Ros Nanopoulos, and Lars Schmidt-thieme. L.s.: Learning optimal ranking with tensor factorization for tag recommendation. In *In: KDD '09: Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[RPS99] Maximilian Riesenhuber, Tomaso Poggio, and Early Studies. Hierarchical models of object recognition in cortex, 1999.

[SS09] Arjan T. Setz and Cees G. M. Snoek. Can social tagged images aid concept-based video search? In *Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, ICME'09, pages 1460–1463, Piscataway, NJ, USA, 2009. IEEE Press.

[SSSS07] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM.

[SvZ08] Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 327–336, New York, NY, USA, 2008. ACM.

[SXL12] Jitao Sang, Changsheng Xu, and Jing Liu. User-aware image tag refinement via ternary semantic analysis. *IEEE Transactions on Multimedia*, 14(3-2):883–895, 2012.

[TJL+11] David Tsai, Yushi Jing, Yi Liu, Henry A. Rowley, Sergey Ioffe, and James M. Rehg. Large-scale image annotation using visual synset. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 611–618. IEEE, 2011.

[TMY78] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. *IEEE Transaction on Systems, Man, and Cybernetics*, 8(6):460–472, June 1978.

[Tuc66] Ledyard Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966. 10.1007/BF02289464.

[vdSGS10] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[vdWS06] Joost van de Weijer and Cordelia Schmid. Coloring local feature extraction. In *Proceedings of the 9th European conference on Computer Vision - Volume Part II*, ECCV'06, pages 334–348, Berlin, Heidelberg, 2006. Springer-Verlag.

[vGVSG10] Jan C. van Gemert, Cor J. Veenman, Arnold W. M. Smeulders, and Jan-Mark Geusebroek. Visual word ambiguity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1271–1283, July 2010.

[WFZY10] Zheng Wang, Jiashi Feng, Changshui Zhang, and Shuicheng Yan. Learning to rank tags. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, pages 42–49, New York, NY, USA, 2010. ACM.

[YfCKH07] Akira Yanagawa, Shih fu Chang, Lyndon Kennedy, and Winston Hsu. Columbia university's baseline detectors for 374. Technical report, Columbia University ADVENT, 2007.

[ZYM10] Guangyu Zhu, Shuicheng Yan, and Yi Ma. Image tag refinement towards low-rank, content-tag prior and error sparsity. In *Proceedings of the international conference on Multimedia*, MM '10, pages 461–470, New York, NY, USA, 2010. ACM.

Ringraziamenti

Desidero ringraziare i professori Alberto Del Bimbo e Paolo Frasconi per avermi dato la possibilità di fare questa tesi con loro. È stata un'esperienza divertente, a tratti sicuramente stancante se non demotivante, ma c'è qualcosa che rende questo tipo di lavoro una delle più belle cose che ho mai fatto. Il professor Del Bimbo è stato in grado di riportarmi in carreggiata in un momento in cui ero perso, restituendomi fiducia e capovolgendo la situazione a mio vantaggio. Il professor Frasconi mi ha trasmesso un po' della sua passione per la ricerca e mi ha seguito lungo tutto lo studio delle tecniche di decomposizione matriciale. Un grazie speciale a Marco Bertini e Lamberto Ballan che hanno avuto la pazienza e la voglia di seguirmi lungo tutto il cammino di questo lavoro, anche quando apparivo dal nulla con le domande più strane. Tutti i "ragazzi" del laboratorio *Lorenzo*, *Andrea*, *Beppone*, *Beppino*, *Giuseppe*, *Andy*, *Iacopo*, *Svebor*, *Gianpaolo* (spero di non scordarmi nessuno, non me ne vogliate sono 36 ore che sono sveglio :)) che mi hanno sostenuto con i loro consigli e la presenza sempre amichevole.

Grazie alla mia famiglia e a tutto quello che ho mi è stato donato. Grazie alla mia ragazza Valentina. Grazie a tutti gli amici che ci sono stati e che ci sono, che hanno condiviso qualcosa con me e che hanno fatto parte della mia vita anche solo per poco.

Una persona ha in sé la somma di tutti i singoli apporti e quindi grazie a tutti per tutto quello che mi avete donato.