Università degli Studi di Firenze
Facoltà di Ingegneria

Corso di Laurea in
Ingegneria Informatica

Annotazione automatica di video mediante similarità visuale e tag suggestion

Tesi di Laurea di
Michael Francalanci

28 Novembre 2011

Relatore:

Prof. Alberto Del Bimbo

Correlatore:

Ing. Marco Bertini
Ing. Lamberto Ballan

Anno Accademico 2010/2011

Ai miei genitori

Indice

Capitolo 1

Introduzione

1.1 Scenario

Negli ultimi anni abbiamo assistito ad un grande aumento di contenuti multimediali sul web, soprattutto di foto e video, grazie alla popolarità di siti come YouTube, Google Video, Flickr e Facebook e di una struttura sempre più veloce che ci permette ormai di condividerli in "real-time". Questi file dovranno essere classificati e in seguito ricercati sulla base di parole chiave che ne descrivono il loro contenuto visuale. E' sempre più importante catalogare questi contenuti multimediali in maniera corretta anche in vista di un *Semantic Web* dove verrano trattati come oggetti, non solo i documenti, ma anche le relazioni tra essi. In questo modo è possibile modellare la conoscenza di Internet, dove ogni contenuto o oggetto dà come contributo alla rete non solo il dato stesso ma anche i concetti sematici che esso rappresenta. In precedenza erano i webmaster che decidevano quali meta-data associare ad ogni documento, poi con l'avvento del Web 2.0 ciò non è più controllabile e sono gli stessi utenti che possono interagire e decidere come classificare i contenuti. Da qui ogni sito che contiene file multimediali (come per esempio

Flickr e Picasa per le immagini, o Youtube e Vimeo per i video) dà la possibilità all'utente di taggare i propri file. Questi tag vengono poi utilizzati per facilitare la gestione e il recupero dei contenuti nei siti.

Dobbiamo distinguere però la procedura di tagging nelle foto o immagini e quella nei video. Il tagging delle immagini può essere fatto riferendosi all'intera immagine o solo a parti di essa in modo molto semplice e veloce. Taggare un video e le sue scene è invece più complesso e richiede tempo che spesso gli utenti non vogliono perdere. Infatti gli utenti tendono a taggare l'intero video e non le scene, dando un'idea generale del contenuto. Tutto questo avviene manualmente quindi l'utente tende ad associare pochi tag e in maniera molto sconnessa tra loro pur di risparmiare tempo.

Per questi motivi si deve trovare un modo per annotare i video in maniera corretta e allo stesso tempo non far pesare questa procedura sull'utente finale.

1.2 Obiettivi

L'obiettivo ultimo del lavoro è annotare automaticamente i video di *Youtube* su base semantica a livello shot utilizzando una validazione visuale. Cioè aumentare il bagaglio semantico e localizzarlo correttamente nel corso del video. Per associare un concetto semantico ad una scena del video utilizziamo la similarità visuale tra il concetto e i frame della scena. Grazie a queste annotazioni viene *espanso* lo spazio semantico dei metadati del video, così che in futuro i video possano essere classificati e ricercati nel web in maniera più specifica e sulla base delle loro relazioni semantiche.

1.3 Organizzazione tesi

Nel capitolo 2, dedicato allo Stato dell'Arte, fornirò una breve panoramica dei lavori che hanno costituito la mia base di partenza. Nel capitolo 3 definisco in dettaglio i *descrittori locali visuali* usati (SIFT e SURF), la struttura dei *Visual Vocabulary* usati per comparare la similarità visuale delle immagini, passando attraverso i modelli a cui si riferisce. Nel capitolo 4 entrerò nello specifico dell'algoritmo che ho realizzato fornendo anche le specifiche implementative. Il Capitolo 5 contiene i risultati sperimentali sui dataset di riferimento ed il Capitolo 6 enuncia le conclusioni di questo lavoro.

Capitolo 2

Stato dell'arte

In questo capitolo analizzo gli articoli che in letteratura riguardano i lavori della mia tesi. Essi sono alla base del mio progetto e mi hanno dato ispirazione durante lo sviluppo.

Ogni articolo viene introdotto in modo generale ed è presente nella bibliografia

2.1 Tag suggestion and localization in user-generated videos based on social knowledge

2.1.1 Introduzione

Il lavoro presentato nell'articolo [1] suggerisce un metodo per annotare in modo semiautomatico un video suggerendo nuovi tag e localizzandoli temporalmente al suo interno. basandosi su social sites come YouTube e Flickr. Questo metodo si divide in due parti principali: la similarità visuale tra immagini e il suggerimeto dei tag. Ho utilizzato il modello rappresentato in

questo articolo come base per lo sviluppo della mia tesi. Ora cerchiamo di capire come funziona più in dettaglio.

2.1.2 Tag suggestion for video annotation

L'articolo proposto vuole raggiungere due obiettivi: aumentare il numero di tag associati al video e associare agli shot solo i tag ad esso rilevanti. Il video che vogliamo analizzare viene suddiviso in scene attraverso un metodo semplice e veloce che si basa sul gradiente di luminosità. Per ogni scena vengono estratti tre *keyframe* che creano il set $K = k_1, k_2, ..., k_o$. Il numero di tag di partenza viene "espanso" utilizzando un set di sinonimi ottenuti da *Wordnet*. Il nuovo set di tag viene utilizzato per scaricare le immagini da *Flickr* che costituiranno il DataSet di immagini. Ogni immagine del DataSet è annotata attraverso uno o più tag. L'unione di tutti questi tag rappresenta il dizionario delle parole con cui possono essere annotati i *keyframe* del video. Ovviamente non tutti i tag sono rilevanti, si deve quindi calcolare un punteggio di rilevanza di ogni singolo tag per identificare quelli più importanti. Il calcolo della rilevanza si basa sull'algoritmo presente in [2], che calcola il numero di occorrenze dei tag presenti nei *k-nearest neighbours* del frame meno la frequenza assoluta del tag. I *k-nearest neighbours* sono le k immagini visualmente più simili al frame. Per il confronto tra frame e immagine di Flickr viene usata una *visual features* a 72 dimensioni: 48 dimensioni per un correlogramma di colore calcolato nello spazio di colore HSV, 6 dimensioni date dal momento di colore calcolato nello spazio di colore RGB e un vettore a 18 dimensioni per tre feature di Tamura (granularità, il contrasto e la direzionalità). Ottenuta la lista dei tag rilevanti per lo shot dai *k-nearest neighbours* viene applicato l'algoritmo *Vote+* ottenendo un punteggio per ogni tag nel seguente modo:

Sia C l'insieme dei candidati e K quello dei keyframe estratti dal video di partenza; allora per ogni $c \in C$ e $k \in K$ si avrà:

$$score(c,k) = score(c,T_k) \cdot \frac{\lambda}{\lambda + (rank_c - 1)} \tag{2.1}$$

I cinque tag con punteggio più alto vengono associati allo shot. Le prestazioni sono incoraggianti anche se variano molto da categora a categoria. E' stato utilizzato un dataset formato da video di YouTube, 4 per ognuna delle 14 categorie, così da coprire i vari contesti possibili. I risultati dipendono molto anche dall'importanza sociale dell'argomento del video.

Sono indicate due direzioni per lo sviluppo futuro: utilizzare descrittori locali per localizzare meglio oggetti o scene all' interno delle immagini ed espandere semanticamente lo spazio dei tag prima della costruzione del DataSet.

2.2 Learning Social Tag Relevance by Neighbor Voting

Il seguente articolo [2] propone un metodo di *tag suggestion* applicato alle immagini basato sui voti dei "vicini" visualmente simili. La rilevanza dei tag di un' immagine è ottenuta dalla loro frequenza nel set di immagini simili. Maggiore è la frequenza del tag più esso è rilevante per l'immagine. Può succedere che alcuni tag che hanno una frequenza alta non siano sempre dei tag corretti, ma possono essere dei tag di significato generale che sono molto presenti nel DataSet. Per ovviare a questo problema viene presa in considerazione, oltre alla loro distribuzione nel set di immagini simili, anche la distribuzione nell'intero set di immagini.

2.2.1 Relevance criteria

Si denota con D l'insieme di immagini annotate e con W il vocabolario dei tag utilizzato in D. Data un'immagine I $\in D$ ed un tag w $\in W$, di definisce r(w, I) : W, D $\rightarrow \mathbb{R}$ una misura della rilevanza del tag.

Questa misura soddisfa i seguenti criteri:

- date due immagini I_1 , $I_2 \in D$ ed un tag w $\in W$, se w è rilevante rispetto a I_1 ma irrilevante rispetto a I_2 , allora

$$r(w, I_1) > r(w, I_2)$$

- dati due tag w_1 , $w_2 \in W$ ed un immagine I $\in D$, se I è rilevante rispetto a w_1 ma irrilevante rispetto a w_2 , allora

$$r(w_1, I) > r(w_2, I)$$

2.2.2 Tag Relevance

La misura di rilevanza utilizzata in questo articolo esprime come l'importanza del tag w dipenda dal numero delle sue occorrenze nei *k-nearest neighbors* dell'immagine I e dalla frequenza a priori del tag stesso. Questa misura prende il nome di *tagRelevance* e si ottiene nel seguente modo:

$$tagRelevance(w, I, k) := n_w[N_f(I, k)] - Prior(w, k)$$

Cerchiamo di capire il significato di questa formula. N_f restituisce i *k-nearest neighbors* dell'immagine I utilizzando una distanza *f*. Invece n_w conta le occorrenze del tag w nell'insieme delle immagini annotate che gli viene passato. Nel suo complesso la formula $n_w[N_f(I, k)]$ indica il numero di "voti" dei vicini per il tag w.

La funzione *Prior* la possiamo approsimare con

$$Prior(w,k) = k\frac{|L_w|}{|D|}$$

dove k è il numero di vicini visuali, $|L_w|$ il numero di immagini etichettate con il tag w e |D| il numero di immagini del DataSet. In questo modo hanno ottenuto una funzione *tagRelevance* che soddisfa i 2 criteri prima citati e che prende in considerazione la distribuzione dei tag a livello locale (attraverso i *k-nearest neighbors*) e a livello globale grazie alla funzione *Prior*.

2.3 Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning

Nell'articolo [3] viene proposto un algoritmo per la creazione di una foresta di vocabolari adattativi per indicizzare immagini e per l'apprendimento di categorie. Questi vocabolari si adattano all'aggiunta di nuove immagini al database, adattandosi a loro volta alla conoscenza immagazzinata. Questo algoritmo utilizza la rappresentazione piramidale e il suo confronto, le specifiche sono presenti nei seguenti articoli [4] e [5]

Inserimento feature

Il metodo di accrescimento dell'albero è composto dai seguenti passi:

- Inserimento di una nuova feature nell'albero
- I nodi sovraffollati vengono rimossi per poi essere ristrutturati
- Clusterizzazione dei punti rimossi nel passo precedente e reinserimento nel vocabolario

Eliminazione nodi obsoleti

Essendo un vocabolario dinamico, che quindi cresce nel tempo, esistono dei nodi che non vengono utilizzati per un lungo periodo. Esiste un limite superiore per il numero di foglie che un albero può possedere; se le dimensioni dell'albero superano questo limite, allora vengono eliminate le foglie utilizzate meno recentemente.

Questo significa che i nodi sono obsoleti rispetto all'obiettivo del nostro problema e soprattutto rispetto al dominio che abbiamo preso in considerazione. Questi due passi lavorano a livelli diversi. Il passo precedente adatta la struttura ad un nuovo inserimento, mentre l'eliminazione dei nodi obsoleti adatta l'albero al dominio, quindi adatta la sua conoscenza.

Mantenimento della piramide

Ogni nodo dell'albero ha una lista di puntatori ai bin delle piramidi a cui tale nodo corrisponde. Gli eventi che modificano il vocabolario e comportano delle variazioni nel sistema di rappresentazione delle piramidi sono:

- Inserimento di una nuova feature nel vocabolario
- Rimozione e ristrutturazione di un nodo del vocabolario

Combinazione di alberi

Vengono combinati i singoli vocabolari ad albero, cosi da portare due vantaggi importanti: in prima istanza, una foresta riduce gli effetti di quantizzazione in prossimità dei contorni dei nodi dei singoli alberi.; secondo, si possono migliorare le prestazioni durante l'esplorazione di una nuova area dello spazio delle feature, per esempio introducendo una nuova classe di immagini.

2.4 Altri contributi

2.4.1 Web Semantico

"Con il termine web semantico, termine coniato dal suo ideatore, Tim Berners-Lee, si intende la trasformazione del World Wide Web in un ambiente dove i documenti pubblicati (pagine HTML, file, immagini, e così via) siano associati ad informazioni e dati (metadati) che ne specifichino il contesto semantico in un formato adatto all'interrogazione, all'interpretazione e, più in generale, all'elaborazione automatica. Con l'interpretazione del contenuto dei documenti che il Web semantico propugna, saranno possibili ricerche molto più evolute delle attuali, basate sulla presenza nel documento di parole chiave, e altre operazioni specialistiche come la costruzione di reti di relazioni e connessioni tra documenti secondo logiche più elaborate del semplice collegamento ipertestuale." (Wikipedia [1])

Internet viene interpretato come una grande "biblioteca" dove i dati hanno bisogno di essere classificati e organizzati prendendo in considerazione i legami tra di documenti. Ci sono tre livelli: il dato vero e proprio, i metadati che esprimono i concetti contenuti in esso (inseriti in uno schema o ontologia) e le relazioni tra i concetti. In questo modo è possibile modellare la conoscenza di Internet: ogni dato dà come contributo alla rete non solo il dato stesso ma anche i concetti sematici che esso rappresenta.

Sarà possibile in futuro effettuare ricerche specifiche ed intelligenti esprimendo delle asserzioni composte da soggetto, predicato e oggetto.

[1] http://www.Wikipedia.org

2.4.2 Automatic Tagging Steps

Il tagging è diventato il metodo più usato per etichettare il materiale presente sul Web, in particolare grazie alla sua semplicità. Infatti chi effettua l'upload può associare uno o più tag in poco tempo, permettendo alle comunità di organizzare le risorse e presentarle attraverso browser agli utenti.

Choudhury ([6]) identifica la struttuta di un *framework* per l'*automatic tagging*:

- espansione dei tag definiti dall'utente per la creazione di un set esteso di parole chiave
- ranking del set esteso
- filtraggio per elimiare il rumore
- mapping delle parole chiave ottenute dalle fasi precedenti con repositori ontologici (disambiguazione)

Questi articolo mi è serivito per avere una panoramica generale sull'organizzazione del mio algoritmo e per capire su quali punti focalizzare la mia attenzione.

2.4.3 Le difficoltà

Quello che voglio creare è uno strumento che annoti automaticamente un qualsiasi video sul Web che abbia degli *user tag*. Il fine ultimo della mia annotazione è quello di voler rendere più efficace nel futuro la classificazione e la ricercare dei video sulla base del loro contenuto semantico. Per effettuare una buona annotazione si deve **aumentare il numero di tag** associati al video, proporli nello **spazio temporale giusto** e suggerire tag **corretti**, **espandendo** lo spazio semantico dei metadati che sono a disposizione del video.

La difficoltà principale è proporre uno strumento il più generale possibile, che riesca ad essere efficace con qualsiasi categoria o dominio di video. Per quanto riguarda il suggerimento dei tag da associare, i problemi principali che ho riscontrato sono:

- Come detto prima, possiamo identificare un video in un dominio che ne connota il suo contesto generale, ma poi ognuno ha una sua realtà con i suoi particolari. E' veramente difficile associare al video dei tag che sono rilevanti solo per quella realtà e non per qualsiasi video del dominio.

- Il suggerimento dei tag, come proposto da [1], si basa sul *Social Knowledge* che il Web offre. Se a livello sociale alcune branche della conoscenza non sono state trattate o non sono state approfondite, allora ciò si ripercuote sul nostro sistema di suggerimento visto che non può trovare parole chiavi da suggerire.

Capitolo 3

Presentazione degli strumenti visuali utilizzati

In questo capitolo descriverò i principali argomenti che ho dovuto affrontare, analizzare e in seguitogli strumenti che ho utilizzato nel mio lavoro per il riconoscimento visuale. Come primo argomento tratterò i descrittori locali, dall'individuazione dei *keypoint* alla loro rappresentazione per effettuare il matching visuale tra immagini. Approfondirò i due tipi di descrittori locali visuali che utilizzo, SIFT e SURF spiegando le loro caratteritiche principali.

In secondo luogo presenterò la struttura dati *Adaptive Vocabulary Tree* spiegando perchè ho deciso di utilizzarla e il procediamento di costruzione, passando attraverso il *k-means* clustering e il modello *Bag of Visual Word* che sono i suoi due "pilastri" principali.

3.1 Descrittori locali

Le features locali sono parti dell'immagine con proprietà speciali. Queste features vengono rappresentate attraverso descrittori locali: un insieme di

dati numerici espressi attraverso vettori multidimensionali. Questo approccio consente di riconoscere oggetti all' interno dell'immagine anche se non separati dal background. In letteratura questo metodo viene utilizzato per la classificazione e il risconoscimento di immagini e il matching tra immagini con viste diverse. Le caratteristiche principali di queste features sono l'invarianza a rotazioni, a cambi di scale o di luce e a trasformazioni geometriche affini. Esse determinano la proprietà più importante di questi *keypoint* la ripetibilità (la facilità con cui questi punti vengono ritrovati in viste diverse di immagini). I passi principali per generare l'insieme di features a partire da un'immagine sono i seguenti

- Scale-space extrema detection
- Keypoint localization
- Orientation assignment
- Keypoint descriptor

Descrivo in dettaglio i due tipi di descrittori locali utilizzati nella tesi: Scale Invariant Feature Transform e Speeded Up Robust Feature.

3.1.1 Scale Invariant Feature Transform

Individuazione dei Keypoint

Scale Space

Per individuare locazioni che siano invarianti a cambi di scala questo descrittore utilizza come *kernel*, vista la sua affidabilità, una funzione Gaussiana. Lo Scale Space viene rappresentato da una funzione L(x,y,σ) che si ottiene dalla convoluzione della funzione Gaussiana con l'immagine:

$$L(x , y , \sigma) = G(x , y , \sigma) * I(x , y)$$

dove G rappresenta la seguente funzione $G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

Nell'articolo si utilizza una differenza di funzioni Gaussiane (Difference of Gaussian, DoG), al posto della funzione Gaussiana che rende la localizzazione dei punti più stabile.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Per individuare i massimi e i minimi locali per ogni punto vengono presi in considerazioni gli otto vicini dell'immagine che stiamo analizzando e i nove vicini delle immagini ottenute ad una scala inferiore e superiore, come in figura 3.1. Tale punto viene selezionato come estremo locale solo se è il minimo o il massimo tra tutti i suoi vicini.

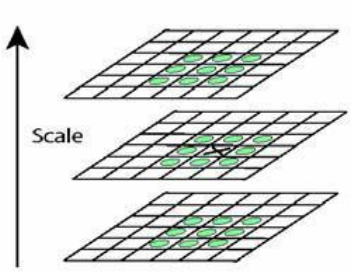

Figura 3.1: Localizzazione degli estremi locali: il punto X viene confrontato con i 26 vicini

Localizzati i punti di interesse viene fatto un filtraggio. Vengono eliminati tutti quei punti che hanno basso contrasto e quelli mal localizzati sull'edge.

Eliminazione dei punti non stabili

Viene sviluppata in [7] una serie di Taylor della funzione 3D quadratica (arrestandoci al secondo ordine), D(x,y,σ):

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x$$

Con D e le derivate valutate nell'estremo locale **x** è possibile valutare se punti a basso contrasto sono da eliminare. A questo punto viene scelta una soglia per eliminare tutti i punti per cui |D(**x**)| è inferiore a tale valore di soglia.

Eliminazione dei response errati degli edge

La funzione DoG restituisce una curvatura importante in presenza di edge e una piccola risposta nella direzione perpendicolare, anche se in tal modo la posizione non è ben determinata. Per risolvere questo problema consi
-deriamo la Hessiana

$$H(D(x,y)) = \begin{bmatrix} \frac{\partial D(x,y)}{\partial x^2} & \frac{\partial D(x,y)}{\partial x \partial y} \\ \frac{\partial D(x,y)}{\partial x \partial y} & \frac{\partial D(x,y)}{\partial y^2} \end{bmatrix}$$

Sia σ l'autovalore più grande e β quello più piccolo, arriviamo a

$$Tr(H) = D_x x + D_y y = \sigma + \beta Det(H) = D_x x D_y y - (D_x y)^2 = \sigma\beta$$

Nel caso in cui il determinante sia negativo, le curvature avranno segni differenti e il punto, non essendo un estremo, verrà scartato. Altrimenti sia σ = rβ, per controllare che la curvatura sia sotto un valore di soglia r

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

dove r è minimo se gli autovalori sono uguali.

Orientazione Keypoint

Ottenuti i keypoint stabili ad ognuno di essi viene assegnata una o più direzioni. Attraverso il gradiente della regione attorno al nostro keypoint, viene

costruito un istogramma formato da 36 bin che rappresentano i 360 gradi possibili. Ogni campione che viene aggiunto all'istogramma viene pesato con il modulo del gradiente e una finestra Gaussiana, con σ 1,5 volte la scala, che conferisce maggiore importanza ai punti vicini al keypoint. La direzione con il picco massimo viene associata al keypoint insieme a tutte le direzioni che hanno come valore almeno l'80% del valore massimo.

Descrittore locale dei punti di interesse

Il descrittore deve essere robusto a trasformazioni affini, cambiamenti di luce e rumore, rimanendo distintivo e veloce per il matching. Qui di seguito spiego la creazione del descrittore facendo riferimento alla figura 3.2. Inizialmente viene calcolato il gradiente nella regione circolare attoro al keypoint (vedi line verde) e pesato attraverso una campana Gaussiana. Si crea un istogramma delle orientazioni del gradiente per le 4x4 sotto-regioni. La figura mostra 8 direzioni per ogni istogramma , dove la lunghezza di ogni vettore corrisponde alla somma dei moduli delle orientazioni corrispondenti a quella direzione nella sotto-regione.

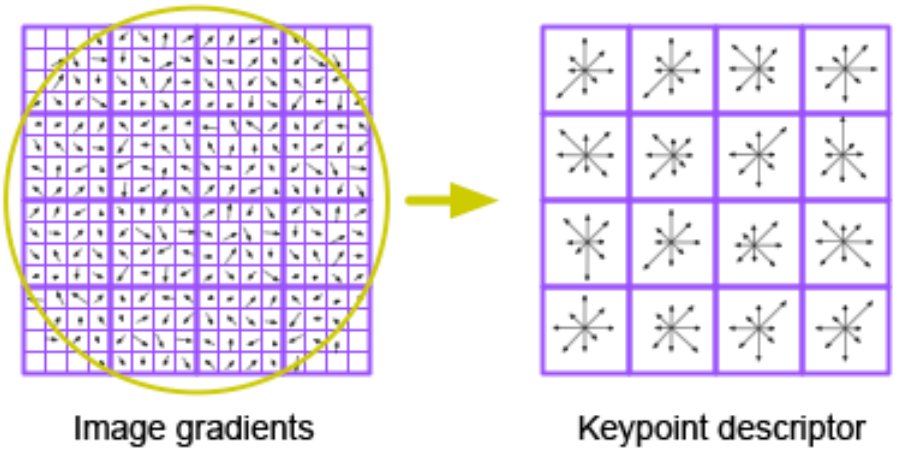

Figura 3.2: Dal gradiente al descrittore locale a 128 dimensioni.

3.1.2 Speeded Up Robust Feature

Individuazione dei Keypoint

Immagine Integrale L'immagine integrale o Integral Image viene utilizzata nell'individuazione dei *keypoint* per aumentarne le performance. Grazie ad essa è possibile calcolare la somma delle intensità di un'area rettangolare di qualsiasi dimensione in 4 operazioni. Data un' immagine I e un suo punto *(x,y)*, l'immagine integrale *II(x,y)* si ottiene sommando le intensità dei pixel contenuti nel rettangolo tra *(x,y)* e l'origine dell'immagine posta in alto a sinistra.

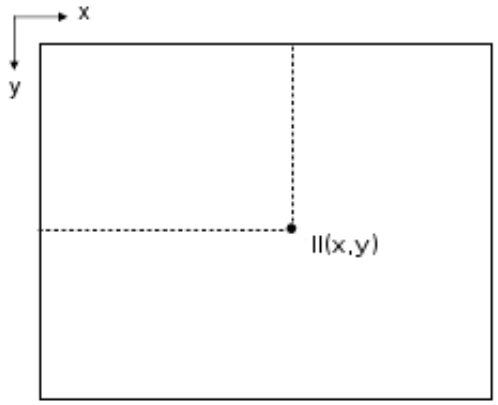

Figura 3.3: Computazione area dell'immagine integrale partendo dall'origine.

La formula per ottenere l'*Immagine Integrale* è la seguente:

$$II(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x,y) \tag{3.1}$$

A questo punto possiamo calcolare, come detto prima, qualsiasi area in quattro operazioni: presi i vertici del rettangolo L_1, L_2, L_3 e L_4 di area A, la somma delle intensità si ottiene nel seguente modo

$$II = L_1 + L_4 - (L_3 + L_2) \tag{3.2}$$

dove L_1, L_2, L_3 e L_4 sono i valori integrali di tali vertici partendo dall'origine.

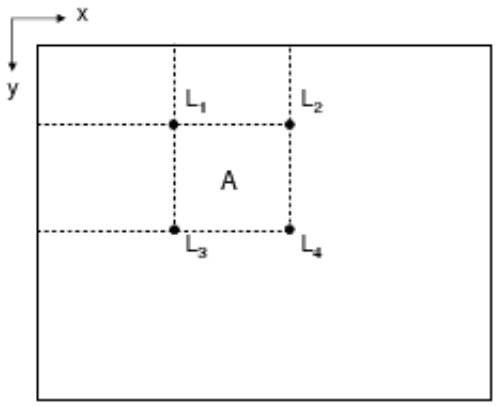

Figura 3.4: Computazione area dell'immagine integrale di un rettangolo

Fast-Hessian Per individuare i *keypoint* il SURF detector utilizza il determinante della matrice Hessiana. Per capire meglio il funzionamento del detector introduco delle nozioni di teoria su tale matrice.

Data una funzione continua *f(x,y)* in due variabili, se tutte le derivate parziali seconde di f esistono allora la matrice Hessiana è una matrice simmetrica composta dalle derivate parziali della funzione *f*.

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x^2} & \frac{\partial f(x,y)}{\partial x \partial y} \\ \frac{\partial f(x,y)}{\partial x \partial y} & \frac{\partial f(x,y)}{\partial y^2} \end{bmatrix}$$

Il determinante della matrice o hessiano è il prodotto degli autovalori della matrice attraverso il quale classifichiamo i punti. Se il determinante è negativo il punto non è estremale, mentre se è positivo (cioè se gli autovalori sono entrambi positivi o entrambi negativi) il punto preso in considerazione è estremale (massimo o minimo locale). La funzione che prendiamo in con-

siderazione è l'intensità dell'immagine I(x,y) e al posto delle derivate parziali viene utilizzato un filtro Gaussiano del secondo ordine normalizzato. La matrice Hessiana viene calcolata nello spazio $\mathbf{p}$=(x,y) e alla scala σ

$$H(I(\mathbf{p},\sigma)) = \begin{bmatrix} L_{xx}(\mathbf{p},\sigma) & L_{xy}(\mathbf{p},\sigma) \\ L_{xy}(\mathbf{p},\sigma) & L_{yy}(\mathbf{p},\sigma) \end{bmatrix}$$

dove $L_{xx}(\mathbf{p},\sigma)$ è la convoluzione tra la derivata Gaussiana del secondo ordine $\frac{\partial g(\sigma)}{\partial x^2}$ e l'immagine nel punto $\mathbf{p}$, in egual modo vengono calcolate le altre derivate. Queste derivate, chiamate Laplacian of Gaussian, vengono approssimate con dei filtri discretizzati.

Scale Space La scale-space è una funzione che può essere usata per trovare punti estremali attraverso tutte le possibili scale. In letteratura lo scale-space è una funzione Gaussiana che viene ripetutamente convoluta con l'immagine originale a diverse scale, come ho spiegato nella sezione *3.1.1.*

L' approccio SURF, a differenza dei SIFT, lascia invariata l'immagine originale e cambia la scala dei filtri, così facendo è possibile computare simultaneamente diversi livelli multipli della piramide, offrendo performance migliori.

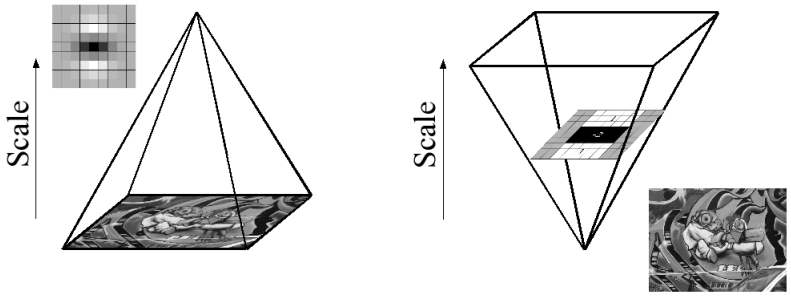

Figura 3.5: Piramide del filtro: nel filtro a sinistra prima viene ridimensionata l'immagine e poi convoluta con la funzione Gaussiana, mentre nel filtro a destra l'immagine originale non cambia scala, sono i filtri ad essere ridimensionati.

La scale-space è divisa in ottave, dove ogni ottava fa riferimento ad una serie di risposte a filtri fino al raddoppiamento della loro scala. L'output del filtro 9x9 è il livello di scala base, al quale riferirsi con un valore di scala s = 1.2, perchè corrispondente alle derivate Gaussiane con $\sigma = 1.2$.

I livelli successivi vengono ottenuti incrementando i filtri base, mantenendo però le giuste proporzioni: 9x9, 15x15, 21x21, 27x27 e via discorrendo. Il rapporto di filtraggio rimane costante anche dopo la scalatura.

Localizzare i keypoint Per individuare la scala dei nostri keypoint bisogna applicare i tre passi seguenti:

- Filtrare con un valore di soglia sotto il quale vengono scartati i punti
- Ogni keypoint viene confrontato con gli otto pixel vicini dell'immagine originale e con i 9 pixel vicini nella scala superiore e inferiore: se l'intensità è il massimo o il minimo il punto viene mantenuto (vedi figura 3.1)
- Interpolare i punti vicini per migliorare l'accuratezza

Descrittore locale dei punti di interesse

Il descrittore SURF ci restituisce come informazione la distribuzione di intensità nell'intorno del punto di interesse trovato. La distribuzione è ottenuta grazie all'utilizzo dei filtri Haar.

Figura 3.6: Haar-wavelet utilizzate per i SURF.

Il primo passo per la creazione del descrittore è trovare la direzione del keypoint data da una regione circolare intorno a tale punto. Vengono calcolati i response delle *wavelet* formati da un response orizzontale e verticale, successivamente pesati con una Gaussiana per dare più importanza ai response vicini al punto di interesse. L'orientazione totale si ottiene sommando tutti i response ottenuti nelle varie direzioni. In seguito viene creata una finestra sulla base di questa orientazione grande 20σ dove σ è la scala con cui è stato calcolato il punto di interesse. Questa finestra viene suddivisa in 4x4 sotto-regione che a loro volta sono suddivise da una griglia regolare 5x5.

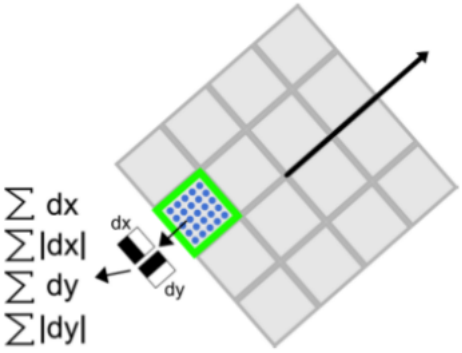

Figura 3.7: Descrittore SURF a 64 dimensioni.

Ad ogni punto della griglia sono applicati i filtri Haar lungo le direzioni x e y. Per ogni blocco viene costruito un vettore nel seguente modo:

$$\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{3.3}$$

risultando un vettore di descrittori a 64 dimensioni date dalle 4 dimensioni per ciascuna delle $4x4$ sotto-regioni.

Oltre al descrittore SURF a 64 dimensioni, esiste anche quello a 128 dimensioni. Nel descrittore a 128 dimansioni le somme di dx e di |dx| vengono calcolate separatamente per dy < 0 e per dy ≥ 0. Lo stesso accade anche per la somma di dy e di |dy|, per cui il numero delle feature raddoppia.

3.2 Vocabulary Tree

Per apprendere in maniera corretta come organizzare e ricercare materiale multimediale in spazi di grandi dimensioni, conviene avere almeno un'idea di come sono stati affrontati questi problemi in precedenza per i documenti testuali. Infatti tanti metodi, soprattutto a livello visuale, prendono ispirazione da lavori svolti con i documenti di testo.

3.2.1 Categorizzazione testuale

Nel tempo il numero di documenti sul Web è aumentato a livello esponenziale ed è diventato sempre più necessario classificarli in maniera accurata per poi rendere possibile agli utenti il loro recupero attraverso tecniche "intelligenti" (Information Retrieval). Per fare questo è necessaria una rappresentazione comune e conveniente per tutti i documenti e una misura di similarità che, dato un documento di partenza, riesca a restituire i documenti ad esso più simili.

Come documento testuale si intende una sequenza di parole che contiene uno o più topic (argomenti), da ora in poi in questa sezione con il termine documento intendo un documento di testo. Dato un'insieme di documenti e come *features* le parole, è possibile costruire un dizionario unendo tutte le parole presenti nei testi. A questo punto il dizionario diventa lo spazio di rappresentazione dei documenti. Un esempio nel caso testuale è il seguente, dati due documenti D_1 e D_2

1. D_1 = John likes to watch movies. Mary likes too.
2. D_2 = John also likes to watch football games.

definiamo il dizionario

dictionary=1:John, 2:likes, 3:to, 4:watch, 5:movies, 6:also, 7:football, 8:games, 9:Mary, 10:too,

Possiamo definire un documento nel seguente modo: un documento è un punto (vettore) nello spazio delle parole del dizionario. Con un vocabolario di dimensione n un documento D viene rappresentato da un vettore

$$D = (w_1, w_2, ..., w_n)$$

dove ogni termine w_i è il peso della parola i nel documento

$$w_i = \begin{cases} 1, & \text{se la parola } i \text{ appare nel docuemnto} \\ 0, & \text{altrimenti} \end{cases}$$

Alla fine il termine w_i rappresenta il numero di volte che quella parola appare nel documento, di conseguenza il vettore rappresenta l'istogramma delle parole nel documento. Spesso oltre al peso della parola si prende in considerazione anche la frequenza della parola all'interno del dizionario. Questo viene

fatto perchè non sempre una frequenza alta è indice di una parola importante: se una parola è troppo comune viene penalizzata dalla sua frequenza alta all'interno del dizionario.

Questo tipo di rappresentazione si chiama **Bag of Words** cioè un'insieme non ordinato di termini. Ora che abbiamo ottenuto una rappresentazione comune per i documenti è possibile confrontarli tra di loro e ottenere un punteggio di similarità. Per fare questo si usa il prodotto scalare tra i due documenti

$$sim(D_1, D_2) = \langle D_j, D_k \rangle = \sum_{i=1}^{n} w_j^i w_k^i$$

In questo modo due documenti hanno un punteggio di similarità massimo quando i loro vettori sono paralleli, ciò non significa che i due testi sono uguali ma che l'occorenza delle parole è uguale (non si tiene conto dell'ordine delle parole). Solitamente il testo prima di essere sottoposto a questo metodo viene *pre-processato* eliminando una parte del rumore (stop-word, lowercase, stemming etc,etc).

Utilizzando una rappresentazione di tipo **Bag of Words** è possibile utilizzare diversi tipi di classificatori come k-nearest neighbour o support vector machines (SVM).

3.2.2 Dal testo alle immagini

Nell'articolo [8] viene proposto un modello di *Bag of Words* di tipo visuale. Sivic e Zisserman definiscono per la prima volta un **Visual Vocabulary** con il quale è possibile utilizzare le tecniche di IR sviluppate per i documenti di testo. I Visual Vocabulary sono nati anche grazie agli ottimi riscontri che hanno avuto i descrittori locali di immagini.

Per adattare il modello BoW a un livello visuale si effettua un analogia tra parole testuali e i cluster dei descrittori locali. In questo modo rimane sempre una rappresentazione non ordinata. Come in precedenza che abbiamo definito un dizionario, cioè uno spazio con il quale rappresentare i documenti di testo anche per i descrittori visuali si deve definire uno spazio. Lo spazio dei descrittori locali è più grande e complesso rispetto a quello delle parole e quindi non basta effettuare una loro unione, si deve discretizzare lo spazio in un numero finito di prototipi. È proprio a questo che serve il *Visual Vocabulary*.

BoW si basa sull'utilizzo di un dizionario in comune per rappresentare i documenti, mentre Bag of Visual Words utilizza un vocabolario comune che propone un numero finito di simboli con cui rappresentare le immagini. L'immagine viene rappresentata dai prototipi associati alle regioni locali che la compongono. In definitiva il vocabolario visuale è rappresentato da due componenti: un algoritmo per la creazione delle parole visuali (creazione del vocabolario) e una regola per la quantizzazione dei descrittori visuali, cioè un regola che, dato un descrittore che non appartiene all'insieme usato per la creazione del vocabolario, mi restituisca la parola visuale ad esso corrispondente.

3.2.3 Creazione vocabolario

L'obiettivo ultimo è discretizzare uno spazio continuo ad alta dimensionalità. Per fare questo di deve utilizzare un algoritmo di clustering, in sostanza un algoritmo che seleziona e raggruppa elementi omogeni in un insieme di dati. L'algoritmo più diffuso e che viene utilizzato anche in questa tesi (nella creazione del *Adaptive Vocabulary Tree* in modo ricorsivo) è il *k-means*.

Dati N elementi con i attributi, modellizzati come vettori in uno spazio vettoriale i-dimensionale, definiamo $X = X_1, X_2, ..., X_N$ come insieme degli oggetti. Si definisce partizione degli elementi il gruppo di insiemi $P = P_1, P_2, ..., P_K$ che soddisfano le seguenti proprietà:

- l'unione di tutti i cluster deve contenere tutti gli elementi
- ogni elemento può appartenere ad un unico cluster
- almeno un oggetto deve appartenere ad un cluster e nessun cluster può contenere tutti gli elementi

Ovviamente $1 < k < N$ in quanto non avrebbe senso suddividere lo spazio in un'unica partizione e tanto meno creare una partizione per ogni elemento. Definiamo una partizione come una matrice $U \in \mathbb{N}^{KxN}$ dove ogni suo elemento $u_{k,n} = 0, 1$ indica l'appartenenza o meno del vettore n al cluster k. Indichiamo con $C = C_1, C_2, ..., C_K$ i centro cluser delle k partizioni e con V la funzione di cui vogliamo calcolare il minimo

$$V(C, U) = \sum_{i=1}^{K} \sum_{X_j \in P_i} \|X_j - C_i\|^2$$

Il minimo viene calcolato utilizzando questo algoritmo in maniera iterativa

1. Genera U_v e C_v casuali
2. Calcola U_n che minimizza $V(U, C_v)$
3. Calcola C_n che minimizza $V(U_v, C)$
4. Se l'algoritmo converge ci si ferma, altrimenti $U_v = U_n, C_v = C_n$ e torna al passo 2

Per convergenza si intende che non vi è nessun cambiamento nella matrice U o che la differenza fra i valori della funzione obiettivo in due iterazioni successive non supera una soglia prefissata.

3.2.4 Struttura ad albero

A questo punto il nostro spazio dei descrittori locali è suddiviso in partizioni ognuna delle quali è rappresentata dal suo centro cluster. Ogni centro cluster rappresenta un parola visuale con il suo insieme di descrittori locali. Dato un descrittore locale se noi vogliamo sapere quale parola visuale associarli dobbiamo confrontare il vettore, come detto prima, con tutti i centro cluster. Ovviamente se il numero di partizioni è molto alto un processo di questo genere inzia ad essere oneroso. Allora viene utilizzata una struttura gerarchica ad albero che ottimizza e diminuisce il numero di confronti. Questa struttura si ottiene applicando ricorsivamente l'algoritmo di *K-means* alle partizioni ottenute. Alla fine le foglie ottenute sono le parole visuali e un vettore per sapere a quale parola visuale è associato deve effettuare un numero di confronti pari a kL dove k è il *branching factor* del *K-means* e L è l'altezza dell'albero.

Nelle figure riportate viene evidenziato il passaggio da uno spazio suddiviso attraverso il *K-means* ricorsivo ad una struttura gerarchica come quella ad albero.

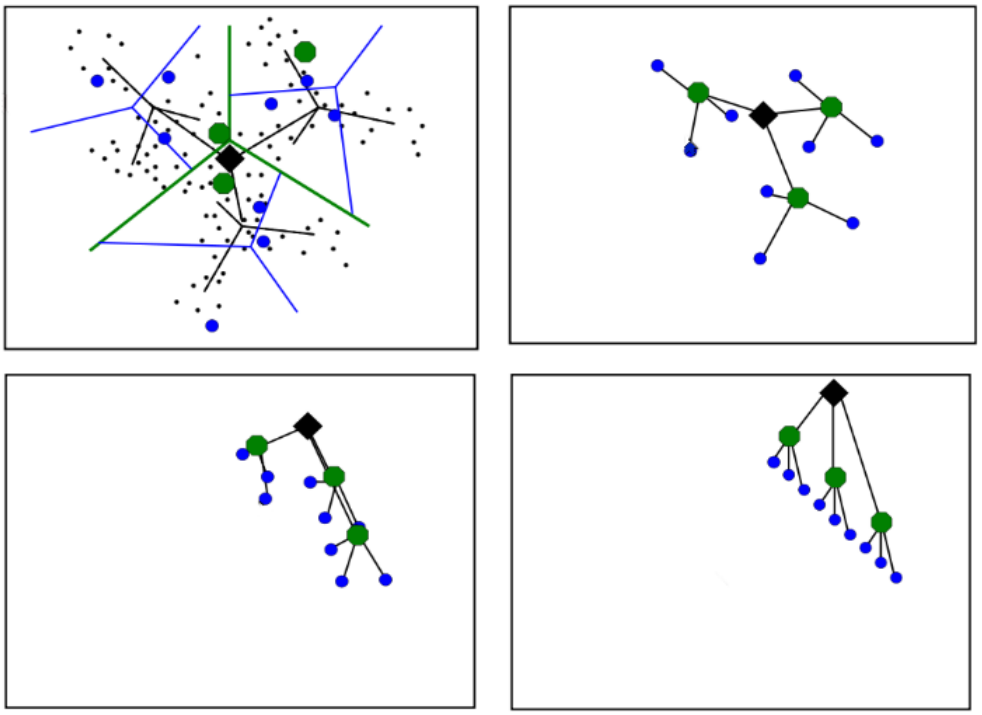

Figura 3.8: Creazione di un albero mediante applicazione di clustering K-means gerarchico. Nella prima immagine notiamo i puntini neri che sono i descrittori locali e l'applicazione del clustering *K-means* con branching factor 3 in modalità ricorsiva. Le immagine che seguono fanno vedere il passaggio ad un vocabolario ad albero.

Fino a questo punto è stata analizzata la creazione delle parole visuali che compongono il nostro *Visual Vocabulary*. Per quanto riguarda l'utilizzo del Vocabolario per effettuare il matching, tra descrittori locali e parole visuali, viene analizzato nella sezione *4.7.2*.

Capitolo 4

L' Algoritmo

In questo capitolo viene inizialmente introdotta la struttura generale ed il contesto operativo del lavoro, fino a specificarne le singole parti e la loro implementazione.

4.1 Struttura generale

L'obiettivo ultimo del lavoro è annotare automaticamente i video di *Youtube* su base semantica a livello shot utilizzando una validazione visuale, cioè aumentare il bagaglio semantico e localizzarlo correttamente nel corso del video. Per associare un concetto semantico ad una scena del video utilizziamo la similarità visuale tra il concetto e i frame della scena.

L' algoritmo ha come ingresso il video preso in considerazione da cui viene recuperata la lista dei tag di partenza. Il video viene suddiviso in scene o shot utilizzando un algoritmo semplice e veloce basato sul gradiente di luminosità. Da ogni scena vengono estratti di default tre frame: uno iniziale, uno nel mezzo e uno alla fine cercando di coprire tutta la scena. Dal nostro set di tag scarichiamo le immagini che andranno a costituire il

DataSet di immagini annotate. Queste immagini sono scaricate da 4 fonti diverse: *ImageNet*, *GoogleImage*, *Flickr* e *Picasa Web*. Queste fonti sono diverse tra loro per due motivi:

- Le immagini di *Flickr* e *Picasa* sono annotate dalla comunità con uno o più tag, mentre le immagini di *GoogleImage* e di *ImageNet* sono sprovviste di tag quindi viene loro associato un unico tag che corrisponde al termine di ricerca.

- Le quattro fonti hanno affidabilità diversa, dove per affidabilità intendiamo la proprietà per cui dato un termine di ricerca vengono restituite immagini correlate con tale termine.

Dal nostro *DataSet* e anche dall' insieme di frame del video vengono estratte delle *features* visuali per effettuare successivamente un matching tra le immagini e i frame. I descrittori visuali utilizzati per rappresentare queste *features* sono di due tipi: SIFT e SURF spiegati in dettaglio nel *Capitolo 3*.

Attraverso i descrittori visuali del nostro *DataSet* di immagini annotate costruiamo un *Adaptive Vocabulary Tree*. Per ogni frame estratto il vocabolario visuale restituisce le prime k immagini simili nel nostro *DataSet* ordinate per punteggio di somiglianza. Ognuna di queste k immagini è correlata da uno o più tag. Questi tag vengono votati o meglio viene calcolata la loro occorenza pesata. L'occorrenza è pesata sulla base della similarità dell'immagine con il frame e dell'affidabilità della fonte dell'immagine. I tag che hanno un'occorrenza maggiore o uguale alla *media* + *varianza* della distribuzione delle occorrenze dei tag relativi alla scena vengono associati ad essa.

Alla fine viene prodotto un file *.srt* che contiene i tag suggeriti per ogni scena, quindi localizzati all'interno del video. Se il tag è stato scritto in

maiuscolo vuol dire che è tra quelli associati al video in partenza altrimenti se è in minuscolo è un tag suggerito dall' applicazione.

Nella figura 4.1 è mostrata una vista d'insieme dell'intera struttura.

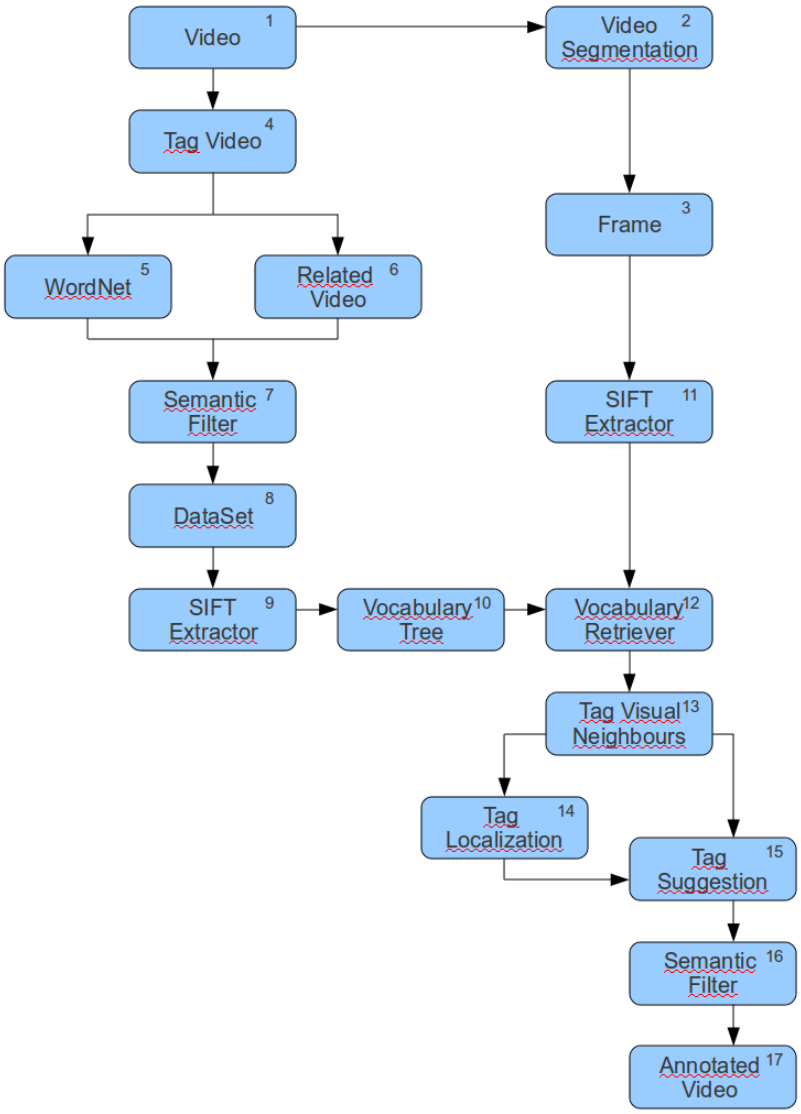

Figura 4.1: Schema generale della struttura del progetto

In seguito vengono affrontati tutti i punti spiegati in precedenza in maniera dettagliata e approfondita. Per capire a fondo il funzionamento si deve aver letto il *Capitolo 2* o essere a conoscenza dei punti che contiene.

4.2 Video: download e data retrieval

4.2.1 Youtube-dl

L'oggetto di partenza della mia applicazione è un video di *YouTube* e le informazioni ad esso correlate. Inizialmente, dato il link del video, esso viene scaricato attraverso uno script in Phyton. Si ricorda che questo script potrebbe aver bisogno di essere aggiornato direttamente dal sito ufficiale[1]. Lo script viene eseguito istanziando una classe run-time

```
Process p = null;
Runtime run = Runtime.getRuntime();
```

che esegue il comando, creando un processo figlio p

```
p = run.exec("./youtube-dl.py -o ./"+param.DATABASE+"/"
  +id_video+"/"+id_video+param.VIDEO_FORMAT+" "
  +url_video);
```

Il comando che viene eseguito è formato nel seguente modo:

- Nome script phyton
- Opzione -o per indicare il path dove salvare il video, compreso nome del video e formato
- Link del video da scaricare

[1] http://rg3.github.com/youtube-dl/

Il path è interamente personalizzabile attraverso il file *parameters.properties*.

Ottenuto il video vengono recupreate le informazioni relative ad esso utilizzando le API di *YouTube*. In particolare quello che ci interessa sono i tag associati, per poterli espandere e successivamente costruire il nostro *DataSet* di immagini annotate.

4.2.2 Dettagli API YouTube

Le *YouTube Data API* consentono si ricercare e aggiornare i contenuti di *YouTube* sotto forma di *Google Data API feeds*. Le operazioni effettuate sono di due tipi: autenticazione e ricerca delle informazioni a noi utili relative al video.

Autenticazione

L'autenticazione utilizzata, ClientLogin authentication, necessita di una *developer key*. Per autenticarsi basta instanziare un oggetto di tipo *YouTubeService*

```
YouTubeService service = new YouTubeService(deveKey);
```

e poi fornire le credenziali, e-mail e password, quando si invoca il metodo setUserCredentials

```
service.setUserCredentials("jo@gmail.com", "password");
```

Ovviamente le credenziali usate sono salvate nell'applicazione locale.

Data Retrieval

Ora vediamo come estrarre i metadati che ci interessano (titolo, categoria, tag ...) a partire da un video specifico. Molti *feeds* di YouTube sono delle

collezioni di *VideoEntry*. La classe *VideoEntry* corrisponde a un video specifico di YouTube e contiene come attributi i metadati del video. Le righe di codice corrispondente selezionano la *VideoEntry* relativa all'ID del nostro video

```
String videoEntryUrl = "http://gdata.youtube.com/feeds
    /api/videos/"+id;
VideoEntry videoEntry = service.getEntry(new
    URL(videoEntryUrl), VideoEntry.class);
```

dal quale possiamo estarre i metadati che ci interessano con il metodo getMediaGroup()

```
YouTubeMediaGroup mediaG = videoEntry.getMediaGroup();
```

Istanziato il nostro oggetto *YouTubeMediaGroup* è semplicissimo ricavare e salvare i metadati. Qui sotto mostro come recuperare i tag

```
MediaKeywords keywords = mediaG.getKeywords();
ps.print("Keywords: ");
ArrayList<String> tags=new ArrayList<String>();
for(String keyword : keywords.getKeywords()) {
ps.print(keyword + ",");
tags.add(keyword.toLowerCase());
}
```

Per comodità viene creato un file di testo nominato con l'ID del video nel quale memorizzo le seguenti informazioni: titolo, utente del upload, link, la categoria e i tag. In questo modo ogni video avrà il suo file di testo e sarà possibile controllare in tempo reale se il video che stiamo analizzando corrisponde a quello voluto.

Le API di YouTube verrano utilizzate anche successivamente per l'espansione dello spazio dei tag, in particolare viene recuperata la lista dei *Related Video* cioè dei video correlati a quello di partenza. Nell' immagine 4.2 è riportato un esempio del formato con cui vengono salvate le informazioni.

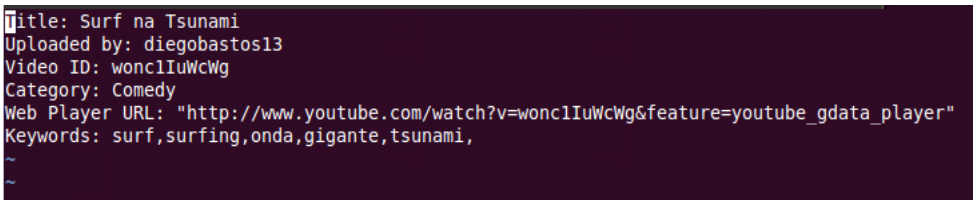

Figura 4.2: Esempio delle informazioni ottenibili da Youtube relativamente ad un video.

4.3 Segmentazione video

4.3.1 Introduzione

Il video che abbiamo scaricato in precedenza viene suddiviso in scene o shot. Per capire esattamente il significato del termine, utilizzo la definizione che ci fornisce *Wikipedia*: "Nel linguaggio cinematografico con il termine scena si intende solitamente un insieme di inquadrature unite tra loro da una continuità di spazio, di tempo e di azione. Nel momento in cui uno di questi tre fattori subisce un cambiamento allora si assiste ad un cambio di scena"[2]. Per ogni scena del nostro video vengono estratti quando possible tre frame: all'inizio, nel mezzo e alla fine. Con questi tre frame, in quanto un scena è abbastanza omogenea, si vuole ricoprire l'intero contenuto visuale della scena mantenendo un livello di computazione basso. Per capire meglio: se estraessi un singolo frame dalla scena potrebbe non coprirmi l'intero contenu-

[2] http://it.wikipedia.org/wiki/Scena(cinema)

to visuale perdendo precisione, invece se estraessi 10 o più frame per scena i passi successivi dell'algoritmo aumenterebbero di complessità senza per forza aumentare la copertura visuale.

Il metodo che utilizzo per segmentare il video è un metodo semplice e computazionalmente poco oneroso, che si basa sul gradiente di luminosità dei frame. In poche parole, per ogni frame viene calcolato il gradiente e creato un vettore lungo quanto l'area del frame. Dato il nostro pixel si calcola la differenza tra il pixel sulla sua sinistra e quello sulla sua destra (lungo l'asse x) nelle tre componenti R, G, B e nello stesso modo lungo l'asse y. A questo punto facciamo la media delle tre differenze lungo x e la media delle tre differenze lungo y, il massimo delle due medie rappresenta il gradiente del pixel. Attraverso questo vettore vengono classificati i cambi di scena; se il gradiente cambia di una soglia prefissata istantaneamente o in modo graduale attraverso un numero di frame (nel nostro caso 15), viene segnalato il cambio di scena.

L' implementazione della segmentazione non è presente nella libreria *Xuggle* ma viene citata nella documentazione ufficiale. La classe che utilizzo io, *SceneChange*, è stata ripresa dalla tesi [9], che a loro volta hanno utilizzato la classe *scenechange2jpg*[3] modificandola per renderla compatibile con JVM 1.6.

4.3.2 Implementazione

Come detto in precedenza, per ogni frame viene costruito in vettore grande come l'area del frame

```
gradients = new short[imgWidth * imgHeight];
```

[3]http://www.exactfutures.com/scenechange2jpg.zip

dove per ogni pixel effettuiamo i seguenti calcoli

```
txR = pixelsR[Cindex + 1] - pixelsR[Cindex - 1];
txG = pixelsG[Cindex + 1] - pixelsG[Cindex - 1];
txB = pixelsB[Cindex + 1] - pixelsB[Cindex - 1];

tyR = pixelsR[Cindex + W] - pixelsR[Cindex - W];
tyG = pixelsG[Cindex + W] - pixelsG[Cindex - W];
tyB = pixelsB[Cindex + W] - pixelsB[Cindex - W];

derivx = (Math.abs(txR) + Math.abs(txG) + Math.abs(txB))/3;
derivy = (Math.abs(tyR) + Math.abs(tyG) + Math.abs(tyB))/3;
```

e la maggiore tra le due medie viene aggiunta al vettore e rappresenta il gradiente del pixel preso in esame

```
hypotenuse = derivx;
if(derivy > derivx)
    hypotenuse = derivy;
gradients[Cindex] = (short)(hypotenuse*20) ;
```

La classe crea un file di nome *cuts.txt* che contiene i valori temporali dei cambi di scena. E' anche possibile salvare i frame dei cambi scena (nella cartella *Changes*) così che l'utente possa controllare direttamente la qualità della segmentazione. Questa opzione è facoltativa e si può settare attraverso il file *parameters.properties*. Altri parametri utili sono: chgtol (di default 25) che rappresenta la tolleranza e ctfrt che rappresenta il numero minimo di frame che devono trascorrere tra una scena e l'altra. Ctfrt è impostato a 15 perchè spesso i video di YouTube hanno un frame rate di 15 frame al secondo, in questo modo una scena non può essere più corta di un secondo.

Trovati gli interevalli dei cambi di scena, sappiamo la durata di ogni scena e possiamo estrarre N frame (di default 3) per ognuna di esse. I frame non possono essere estratti in qualsiasi istante perchè potrebbero non essere presenti, per questa ragione c'è un frame rate che ci indica la cadenza dei frame all' interno della scena. Il primo frame viene estratto subito all'inizio mentre quelli seguenti dopo un periodo calcolato nel seguente modo

$$t_1 = (cut2 - cut1)/N \tag{4.1}$$

$$t = (troncamento[t_1/s]) * s \tag{4.2}$$

con s che indica l'inverso del frame rate cioè ogni quanti secondi è presente un frame. A questo punto ogni intervallo t viene estratto il frame subito precedente ottenendo N frame per scena e non superando il *cut* che indica la scena successiva. I frame vengono salvati in un cartella di nome *Frames* e viene prodotto anche un file di testo che contiene i nomi di tutti i frame.

Ottenuta la segmentazione del video possiamo passare alla sezione successiva.

4.4 Espansione semantica dello spazio dei tag

L'espansione semantica proposta in questa sezione suggerisce nuovi tag da aggiungere a quelli di partenza del video per poi costruire il nostro *DataSet*. Quello che devo proporre è un metodo che garantisca un espansione il più controllata possibile, altrimenti introduco del rumore ancor prima di costruire il *DataSet*. Per espansione semantica si intende aumentare il numero di tag, aggiungendo uno o più tag al nostro spazio iniziale solo se essi sono fortemente connessi a livello semantico con quelli di partenza. Per prima cosa viene

costituito un vettore *suggestion* di possibili tag da aggiungere. A questo vettore vengono aggiunti i *sinonimi* e gli *iperonimi* dei tag di partenza del video proposti da *WordNet*. Oltre ad essi vengono aggiunti i tag dei *Releated Video* di YouTube le cui occorrenze sono superiori, data la distribuzinoie, alla somma della media e della varianza. I tag del vettore *suggestion* vengono filtrati semanticamente attraverso l'utilizzo della misura di *Relatedness* di *Wikipedia Miner*. In fine vengono eliminati i tag che sono presenti nella lista di *Stop Words* e i tag che hanno meno di tre lettere. I tag rimanenti sono i prescelti per ampliare lo spazio dei tag del video per formare il DataSet.

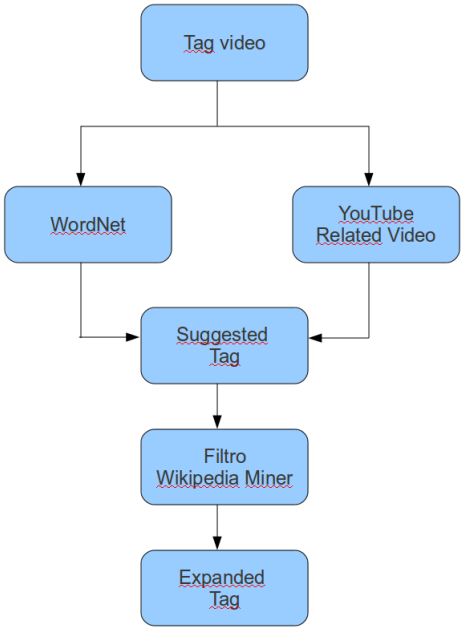

Figura 4.3: Passi principali dell' espansione semantica.

4.4.1 WordNet: sinonimi e iperonimi

Faccio una digressione su *WordNet* e la sua organizzazione. *WordNet* non è un semplice dizionario, ma un'ontologia lessicale. L'ontologia è una branca fondamentale della filosofia che studia l'essere o l'esistenza in quanto tale e le sue categorie fondamentali. In informatica per ontologia si intende il tentativo di creare uno schema concettuale rigoroso di un determinato dominio. Un'ontologia lessicale vuole raccogliere e collegare la conoscenza lessicale delle parole con i concetti semantici e le relazioni che ci sono tra di esse. Per questo motivo è possibile recuperare la categoria a cui un tag appartiene, l' iperonimo, aggiungendo una connotazione generale al nostro spazio dei tag. Oltre ad essi vengono aggiunti i sinonimi di tutti i tag di partenza del video. Qui sotto viene riportato il codice che esegue queste due operazioni. In primo luogo viene caricato il dizionario e aperto,

```
IDictionary dict = new Dictionary(url);
dict.open();
```

ricercato il *synset* corrispondente al nostro tag

```
IIndexWord idxWord = dict.getIndexWord(
videotag.get(i), POS . NOUN );
if(idxWord!=null){
IWordID wordID = idxWord.getWordIDs().get(0) ;
IWord word = dict.getWord(wordID);
ISynset synset = word.getSynset();
```

e poi recuperati sia i sinonimi che gli iperonimi attraverso questi due comandi.

```
IWord synonymous= synset . getWords ()
List <ISynsetID> hypernyms =synset.getRelatedSynsets(
```

```
Pointer . HYPERNYM ) ;
```

In questo modo abbiamo ampliato lo spazio dei tag e dato anche una connotazione più generale. Per aggiungere tag più specifici ci affidiamo in maniera controllata ai *releated video* di YouTube.

4.4.2 YouTube Related video

I *Related Video* sono tutti quei video correlati con il video da analizzare. YouTube non specifica in maniera ufficiale il metodo con cui seleziona questi video, ma effettuando delle prove si capisce che la ricerca si basa sulle informazioni testuali e in particolare sui tag del video corrente. Quindi dato un video V utilizzando le API si può recuperare i primi 25 video correlati $RelatedVideo = \{v_1 , v_2 , ... , v_{25}\}$

Questi ultimi, con i loro tag, formeranno un istogramma delle frequenze, che sarà tagliato dal valore *media + varianza* della distribuzione. I tag che superano il suddetto filtro vengono aggiunti, solamente se diversi dal quelli già presenti, nel vettore *suggest*.

Riassumendo ora abbiamo ottenuto un vettore di possibili tag da aggiungere al nostro spazio dei tag. Questi tag non vengono aggiunti direttamente in quanto sono soggetti a rumore e soprattutto, in questa fase, vogliamo aggiungere solo tag fortemente connessi con il bagaglio semantico del video (altrimenti inquiniamo il *DataSet*) talvolta eliminando anche tag corretti nel contesto ma deboli. Quindi prima devono essere filtrati utilizzando un metodo intelligente.

Per questa parte utilizzo le relazioni semantiche che *Wikipedia Miner* mette a disposizione.

4.4.3 Filtraggio attraverso Wikipedia Miner

Inizio introducendo il progetto *Wikipedia Miner* e in particolare la misura di *Relatedness* che mette a disposizione, facendo riferimento all'articolo scientifico [10].

Wikipedia Miner

Wikipedia oramai è un caposaldo per rappresentare la conoscenza sociale sul Web. Oltre a definire concetti e le loro relazioni semantiche, indica anche l'importanza che la società ha attribuito a questi concetti, è una *folksonomia*. *Wikipedia Miner* è un toolkit open source che permette di navigare all' interno di Wikipedia ed estrapolare informazioni utili attraverso un approccio *object oriented*. Esso indicizza le pagine di Wikipedia in modo da effettuare ricerche efficienti e tempestive.

Quello che viene utilizzato in questa tesi è la misura di *Relatedness* di *Wikipedia Miner*. Tale misura indica la parentela semantica tra due o più concetti, quindi la loro relazione. *Wikipedia Miner* calcola questa misura utilizzando gli hyperlinks degli articoli presenti nel dizionario. La correlazione semantica tra due articoli non si basa su singoli link (che potrebbero essere un errore umano e portare rumore), ma considerandoli in aggregato. Infatti viene considerato il grado di *overlap* dei link in entrata e uscita dei due articoli, eliminando i possibili errori e rappresentando la loro relazione a livello sociale.

L' immagine (fig.4.4) rappresenta una parte del grafo delle relazioni tra i termini *dog* e *cat* e fa capire come lavora il toolkit.

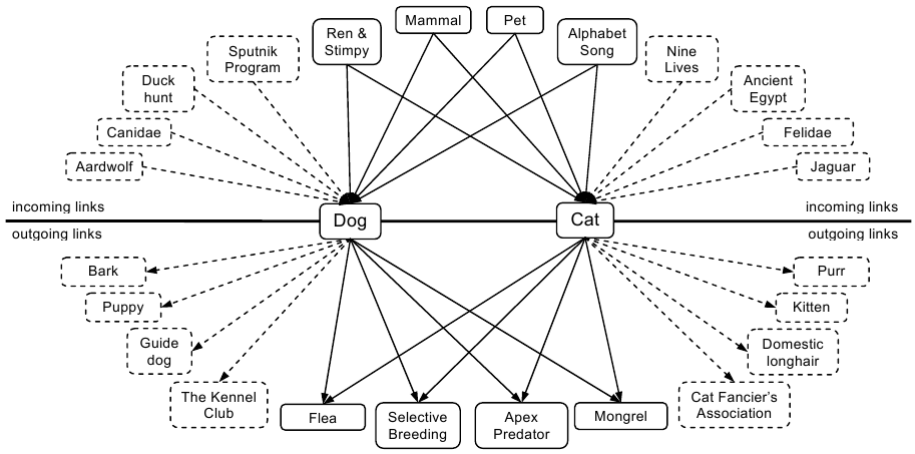

Figura 4.4: Grafo delle relazioni tra gli articoli *dog* e *cat*; possiamo notare come i link vengono considerati in aggregato

Dopo una fase di normalizzazione, si considerano i link che si sovrappongono nella parte superiore (incoming link) e quelli nella parte inferiore (outgoing link) insieme ai link unici e otteniamo un valore di *Relatedness*.

Filtrare i tag utilizzando la Relatedness

Ricollegandoci all'inizio della sezione noi ci troviamo con un vettore *suggestion* di possibili tag da aggiungere al nostro spazio dei tag di partenza. Questi tag sicuramente soggetti a rumore devono essere filtrati. Per ogni tag del vettore suggestion viene calcolato un punteggio ottenuto sommando il valore di *Relatedness* con ogni tag del video. Così facendo ogni possibile futuro tag ha un punteggio che corrisponde alla sua relazione semantica con l'intero bagaglio di tag del video. Attraverso questo metodo si vuole esaltare i tag che sono fortemente connessi con il contesto del nostro video. Calcolata l'intera distribuzione dei punteggi essi vengono filtrati da una campana Gaussiana e solamentei tag rimanenti vengono aggiunti allo spazio dei tag. Come detto in precedenza è possibile che con questo metodo vengano elimi-

nati anche dei tag corretti, però il mio obbiettivo è quello di aggiungere tag che sono sicuramente corretti e fortemente connessi con il video eliminando qualsiasi possibilità di rumore.

Come ultimo filtro, prima di proporre i tag allo step successivo, vengono di novo eliminate le parole che compaiono in una lista di *Stop Words* in ingelse e le parole più corte di tre lettere.

4.5 Creazione DataSet

Arrivati a questo punto ci troviamo nella seguente situazione: abbiamo il video da analizzare suddiviso in scene, recuperato i metadati (tra cui i tag) ed effettuato l' espansione semantica. L'immagine 4.5 mostra, oltre che la situazione attuale, anche il passo successivo la: creazione del *DataSet*.

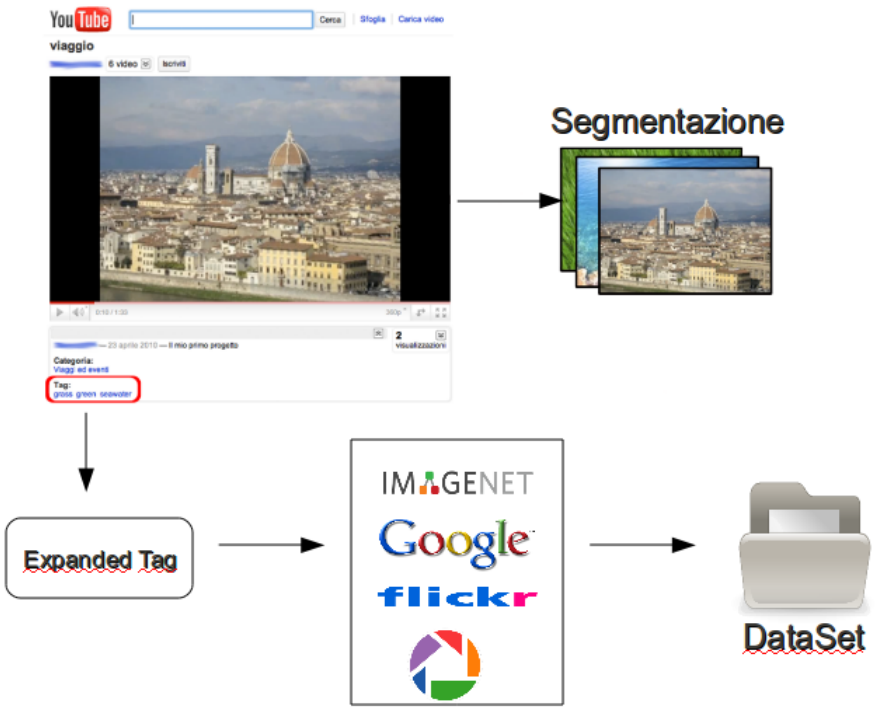

Figura 4.5: Passi dell'algoritmo per la creazione del*DataSet*.

Una volta ottenuta la lista dei tag espansi e depurati dai rumori possiamo creare il nostro *DataSet* di immagini annotate. I tag diventano chiavi di ricerca per scaricare immagini dal Web. Le immagini vengono ricercate in quattro fonti diverse: ImageNet, Google Image, Flickr e Picasa.

4.5.1 ImageNet, Google Image, Flickr e Picasa: tipologie di fonti diverse

Le fonti che utilizzo hanno caratteristiche diverse tra loro. Flickr e Picasa sono delle comunità di persone interessate alla fotografia che condividono album personali e costituiscono anche gruppi tematici. Le foto messe a disposizione dalla comunità sono correlate con uno o più tag che ogni utente

associa alle proprie immagini. Altra premessa invece va fatta per le immagini scaricate da ImageNet e Google Image che non presentano alcun tag, infatti ad esse si associa solamente il termine di ricerca con cui sono state ritrovate. Questa differenza che può sembrare banale è invece fondamentale. Le immagini prese da ImageNet e Google Image servono per effettuare il matching visuale con i frame e collocare i tag del video nelle giuste locazioni temporali, ma non portano nuovi tag da suggerire. Se non ci fossero anche le immagini scaricate da Flickr e Picasa il nostro video verrebbe annotato a livello temporale in maniera corretta solo con i tag del video e i pochi tag aggiunti dall'espansione semantica. Invece, grazie a queste due fonti, è possibile associare nuovi tag alle scene del video.

La seconda differenza è il loro grado di affidabilità. Per affidabilità intendo la proprietà per cui dato un termine di ricerca vengono restituite le immagini correlate con tale termine. ImageNet è un'ontologia visuale basata su WordNet, dato un termine di ricerca, le foto ad esso associate sono sicuramente affidabili. La sua affidabilità copre un numero esiguo di concetti visuali ed è per questo che si deve utilizzare la altre fonti. Per le altre tre fonti è stato effettuato un test per calcolare l'affibadilità. Il test è stato effettuato su 20 tag diversi e per ogni sorgente ho scaricato le prime 20 immagini, per un totale di 1200 immagini. Manualmente ho controllato le immagini e calcolato la percentuale di immagini corrette, riportate in Tabella 4.1. ImageNet come spiegato in precedenza, può essere definita una sorgente controllata e quindi la sua affidabilità è pari a 1.

Questa differenza di affidabilità verrà riutilizzata in seguito insieme alla similarità visuale per calcolare la *Tag Relevance* per il suggerimento dei tag a livello shot.

Query	**Flickr**	**Picasa**	**Google**
Gireff	0.9	0.65	1
Lake	0,85	0,85	1
Barack Obama	0,5	0,65	1
Eiffel Tower	0,85	0,95	1
Bottle	0,85	0,65	0,95
Guitar	0,95	0,85	1
Violin	0,9	0,75	0,95
Opossum	0,8	0,25	1
Courtney Love	0,8	0,9	0,95
Sand	0,9	0,6	1
Lamp	0,8	0,9	1
Berlin	0,8	0,35	1
Berlin reichstag	0,95	1	0,9
Floor	0,85	0,9	0,95
Polarbear	0,95	0,65	1
Wedding	0,8	0,8	0,8
Sunset	0,95	1	1
Military	0,15	0,4	0,85
Zoo	0,9	0,9	0,9
Coca-Cola	0,95	0,7	0,95
Affidabilità media	0.82	0,7	0,96

Tabella 4.1: Affidabilità delle 3 sorgenti per il download

4.5.2 API e Web Service

Le foto scaricate vengono salvate in una cartella di nome *Foto* di default ma comunque impostabile attraverso *parameters.properties*. Nella carella oltre alle immagini è presente un file *Foto_name.txt* che contiene esclusivamente i nomi delle foto formati dalla lettera iniziale della sorgente e da un indice incrementale. Anche un'altro file è presente, *Foto_Termini.txt*, dove ogni riga è formata

```
Foto/P55.jpg met 3**met**new**york;
```

con path della foto, termine di ricerca, numero di tag associati e la lista dei tag separati da **. Attraverso questo file è possibile recuperare per ogni foto i tag da proporre nella *Tag Suggestion*.

I metodi per effettuare il download delle foto sono l'utilizzo delle API o un servizio di Web Service. Per Flickr ho utilizzato le API Flickrj[4] e per Picasa[5] quelle ufficiali di Google. Sul Web sono presenti le documentazioni ufficiali che sono esaustive e correlate di esempi. Per questo motivo non mi dilungo nella spiegazione del loro utilizzo, specifico invece i servizi di Web Service che ho utilizzato per ImageNet e GoogleImage.

Google Image

Google Image indicizza le immagini dei siti web attraverso i meta tag come per esempio alt tag oppure la descrizione dell'immagine. Utilizzando il Web-service dato a disposizione attraverso le ajax google api, effettuo una chiamata al seguente URL "http://ajax.googleapis.com/ajax/services/search/images?

[4]http://www.flickr.com/services/api/

[5]http://code.google.com/apis/picasaweb/code.html

v=1.0&q=search+term&rsz=large&start=0“ che restituisce una pagina web contenente una singola riga dove sono presenti altezza, larghezza, id, url dell'immagine, titolo e altre informazioni per ogni immagine. Attraverso questo metodo non ho bisogno di nessuna chiave di accesso (Apikey) e non devo istanziare nessun oggetto della classe Google. Il codice

```
URL url=new URL("http://ajax.googleapis.com/ajax/
   services/search/images?v=1.0&q="+tag[k]+"&rsz
=large&start="+page);
page=page+8;
urlConn = url.openConnection();
urlConn.setDoInput(true);
urlConn.setUseCaches(false);
File f=new File("./"+dir+"/tmp.txt");
```

apre una connessione in input a tale Url senza l'utilizzo di caches e crea il file temporaneo tmp.txt dove salvare l'output ottenuto. Vediamo come utilizzare questa Url attraverso gli argomenti per effettuare una query:

http://ajax.googleapis.com/ajax/services/search/images?v=1.0&q
=search+term&rsz=large&start=0

dove

- v indica la versione utilizzata
- q specifica il termine da cercare con '+' al posto dello spazio se è un termine composto
- rsz può essere definito small cioè 4 risultati a query oppure large 8 risultati a query
- start è l'indice da cui parte la ricerca

Ho impostato il campo rsz come large e attraverso start e page imposto l'indice come multipli di 8 in modo da non ripetere i risutati già ottenuti. Estrggo l'url delle 8 immagini dal file tmp.txt ricercando la sottostringa "unescapedUrl:" e le salvo nella cartella Foto, mentre nel file Foto_Termini.txt salvo una riga per ogni foto insieme al termine di ricerca.

ImageNet

Con Wordnet traduco i termini di ricerca in synset per poi passarli a ImageNet nella ricerca delle foto. Non tutti i synset che riesco a reperire sono poi validi per la ricerca su ImageNet, infatti non c'è una corrispondenza biunivoca in quanto le immagini su ImageNet sono molto minori rispetto ai termini presenti su Wordnet. Il primo passo è creare un oggetto Dictionary e poi aprirlo come segue:

```
IDictionary dict = new Dictionary(url);
dict.open();
```

Poi va effettuata la conversione da termine di ricerca a synset con il seguente codice:

```
IndexWord idxWord = dict.getIndexWord(tag[i], POS.NOUN);
if(idxWord!=null){
   IWordID wordID = idxWord.getWordIDs().get(0);
   IWord word = dict.getWord(wordID);
   String s[]=wordID.toString().split("-");
   String ID="n"+s[1];
```

per recuperare l'indice del mio termine di ricerca e recupero il suo ID. L'ID viene riformattato per poi effettuare se possibile la ricerca della foto su ImageNet. ImageNet è un database di immagini basata sulla gerarchia di Word-

net (solamente dei nomi). Per questo motivo precedentemente nella formattazione viene aggiunta una n di "nouns" per formare il synset. A differenza delle altre fonti le immagini scaricate da ImageNet sono affidabili nel senso che la foto corrisponde sicuramente con il termine di ricerca. Tutto questo va a discapito dei pochi termini che possono essere ricercati. Per quanto riguarda il synset, esso viene utilizzato nella seguente Url

http://www.image-net.org/api/text/imagenet.synset.geturls?
wnid=mio_synset

che ci restituisce a sua volta le Url delle relative foto. E' necessario salvare tutto in un DataInputStream che conterrà per ogni riga una Url delle foto.

```
URL urlimage=new URL("http://www.image-net.org/api/text
  /imagenet.synset.geturls?wnid="+id);
urlConn = urlimage.openConnection();
urlConn.setDoInput(true);
urlConn.setUseCaches(false);
dis = new DataInputStream(urlConn.getInputStream());
```

Da ogni immagine del *DataSet* che ho scaricato vengono estratti i descrittori locali per poi formare *Adaptive Vocabulary Tree*.

4.6 Features Extraction

Una volta creato il *DataSet* è stata realizzata la procedura per il calcolo dei descrittori. Lo stesso procedimento viene applicato ai frame estratti dal video che servono per il confronto con le immagini annotate. I descrittori locali utilizzati sono i SIFT e SURF a 128 dimensioni. Per estrarre i descrittori

è stata utilizzata la libreria *libpmk-2.5*[6] in C++. Per interfacciarmi con la libreria utilizzo il *Java Native Interface*, un *framework* che permette al codice Java di chiamare librerie native. Da ogni immagine o frame, attraverso il metodo *getFeatures* della classe Extractor.cpp, estraggo i descrittori a 128 dimensioni e li salvo su un file che ha per nome, il nome dell'immaginie o frame e come estensione .psl.

Sotto riporto il metodo *getFeatures* in C++ che si interfaccia con la classe Extractor.java. Nel codice possiamo vedere che prima viene effettuata la *detection* e poi viene passato a *extraction* il vettore di *keypoint* localizzati. Per ultimo viene creato il file .psl che contiene tutti i descrittori dell'immagine.

```
dete.DetectInterestPoints(pathfoto, &vec);
if((int)vec.size()>0)
{
PointSet descSet=extr.ExtractFeatures(pathfoto, &vec);
cout<<" have been extracted "<<descSet.size()
  <<" points.";
 if(descSet.size()!=0){
      psl.add_point_set(descSet);
      flag++;
      psl.WriteToFile(pslfilename);
      }
 else
  { cout<<", can't extract no points."<<endl;
    return 0; }
 }
```

[6] http://people.csail.mit.edu/jjl/libpmk/

```
else
 { cout<<" can't extract."<<endl;
  return 0; }
```

Anche per la creazione dell'*Adaptive Vocabulary Tree* e il *retrieve* sono state create delle classi in C++ utilizzando sempre il JNI; le classi sono rispettivamente Vocabualry.cpp e VocabularyRetriever.cpp. Estratti i descrittori locali da tutte le immagini del *DataSet* viene creato l'*Adaptive Vocabulary Tree*. La struttura del *Vocabulary Tree* è stata analizzata in maniera approfondita nel *Capitolo 3*, quindi in questa sezione approfondisco le caratteristiche adattative di un *Adaptive Vocabulary Tree*.

4.7 Adaptive Vocabulary Tree

4.7.1 Growing

Il vocabolario è costruito utilizzando i descrittori delle immagini del *DataSet*, mentre i descrittori dei frame vengono utilizzati nella parte di Retrieve. A differenza del *Vocabulary Tree*, che per la sua costruzione necessita che tutti i descrittori locali siano estratti dalle immagini prima di applicare il clustering, nell *Adaptive Vocabulary Tree* possiamo inserire un'immagine alla volta. Il processo di inserimento si suddivide in tre passi principali:

1. Un nuovo punto viene inserito nel vocabolario
2. Le foglie sovraffollate vengono rimosse in preparazione della ristrutturazione
3. I punti rimossi nello step precedente vengono clusterizzati e reinseriti nel vocabolario.

Ogni step è controllato da un parametro specifico. Quando la capcità C massima delle foglie, viene sorpassata, si attivato lo step *2*. La ristrutturazione non viene effettuata solo sul nodo sovraffollato ma anche sul vicinato del nodo definendo il numero di vicini che partecipano attraverso il *neighborhood size* S. Per lo step *3* si definisce il numero massimo di punti che vogliamo nelle foglie dopo il clustering, *reduction factor* R, in modo da prevenire ristrutturazioni troppo frequenti nel solito punto dell'albero. Nella mia tesi questi parametri sono son settati nel seguente modo: branch factor 6, capacity 100, neighborhood size 600 e reduction factor 5. Questi parametri possono essere personalizzati attraverso il file *parameters.properties*.

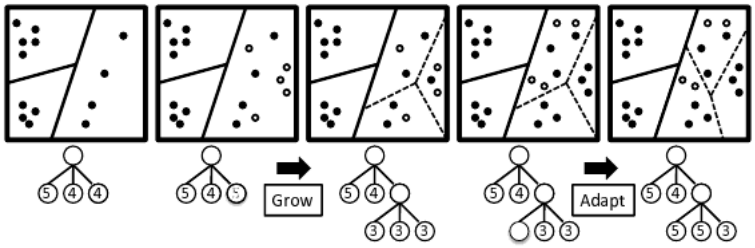

Figura 4.6: Adaptive Vocabulary Tree. Ogni volta che viene inserita una features l'albero cresce e si adatta.

Nello step *1* viene inserito un punto nella radice dell'albero, che si propaga arrivando fino alla parola visuale. Quando viene aggiunto, se la dimensione della parola supera la capacità C, allora si passa all step *2*.

Nello step *2* vengono rilasciati i punti dei nodi del vicinato senza superare S. Si inizia rilasciando i punti della parola visuale che superà C e dei suoi fratelli, poi si passa alla generazione precedente: prima il padre e a sua volta i suoi fratelli, Questo procedimento viene applicato fino a quando non si arriva a un numero di punti pari ad S. In questo modo si passa da un rilascio minimo di un solo nodo, fino a poter rilasciare tutto l'albero.

All'insieme dei nodi rilasciati si applica un clustering *k-means* di tipo ricorsivo (step *3*). Preso x, il nodo in cui si è stoppato il rilascio, e i suoi m figli che non sono stati rilasciati si applica un clustering con $k = B - m$, con B che indica il *branching factor* del vocabolario. In seguito si applica il clustering in maniera ricorsiva. Con questo metodo si mantiene l'albero il più bilanciato possibile e adattato ai punti inseriti.

4.7.2 Retrieve

Grazie al *Visual Vocabulary*, per ogni frame viene restituita la lista delle immagini simili con i relativi punteggi. Prima di tutto va specificato che al vocabolario viene passata una lista di descrittori locali e ogni descrittore viene propagato attraverso i nodi dell'albero. La regola per scegliere il nodo successivo è la seguente: se ci troviamo nel nodo N viene prima calcolare la distanza tra i k figli di N e il descrittore e poi viene scelto il centro cluster (figlio) che minimizza la distanza definita da L2DistanceComputer

```
double L2DistanceComputer::ComputeDistance(const
Point& f1,const Point& f2) const {

  assert(f1.size() == f2.size());
  double sum = 0;
  for (int ii = 0; ii < f1.size(); ++ii) {
    sum += (f1[ii] - f2[ii]) * (f1[ii] - f2[ii]);
  }
  return sum;
}
```

Data h l'altezza dell'albero, per trovare la parola visuale corripondente al descrittore locale, ci vogliano al massimo k_h confronti con k come *branching factor* dell'albero, invece di k_{h+1} confronti che si otterebbero con semplice vocabolario. Per vocabolarii di dimensioni estese il risparmio è consistente.

Quando arriviamo ad una foglia abbiamo ottenuto la parola visuale corrispondente al nostro descrittore. Tramite un *Inverted File* vengono restituite le immagini associate a tale parola visuale e viene creato l'istogramma delle frequenze. Alla creazione dell'istogramma partecipano tutti i descrittori del frame con la relativa parola visuale. La frequenza di ogni immagine viene pesata con la cardinalità dell'immagine all'interno del vocabolario e il numero di descrittori del frame con cui abbiamo fatto la query. A questo punto vengono restituite le prime k immagini simili al nostro frame, cioè le k immagini che hanno la frequenza pesata maggiore. Il numero di immagini simili richieste può essere impostato con il parametro *KNEAREST* del file *parameters.properties* che di default, è impostato a 500. Da questi 3 vettori viene costruito un unico vettore che contiene le k immagini che hanno lo score di similarità più alto.

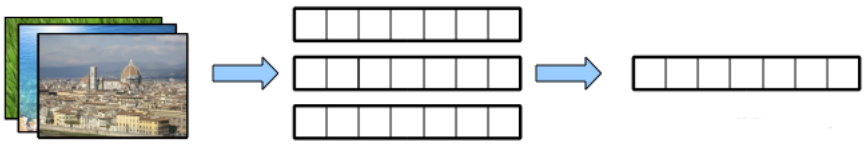

Figura 4.7: Ogni array rappresenta le k immagini simili per uno dei 3 frame. Da questi tre ne otteniamo uno che contiene i massimi dei 3 e rappresenta le prime k immagini simili per lo shot. Siamo passati da una similarità per frame a una per shot.

4.8 Tag Suggestion

La procedura di *Tag Suggestion* che sto per descrivere viene spiegata per un singolo shot, ma è applicata a tutti gli shot del video. I passi principali sono:

- Localizzazione a livello temporale dei tag del video e dei tag espansi nella sezione *4.4* a livello shot
- Suggerimento di nuovi tag a livello shot proposti da Flickr e Picasa
- Filtraggio dei tag suggeriti al passo precedente utilizzando la misura di *Relatedness* di *Wikipedia Miner* con il bagaglio di tag associati al primo passo

4.8.1 Tag localization and first suggestion

Lo shot è correlato da una lista di immagini simili, dove ognuna di queste è annotata da uno o più tag. Come in precedenza viene creato un'istogramma delle frequenze però questa volta viene pesato attraverso la smilarità visuale delle immagini che abbiamo ottenuto con il vocabolario. Ogni immagine porta nell'istogramma i tag che ha correlati pesandoli con il suo punteggio di similarità e anche con il punteggio di affidabilità della fonte da dove è stata scaricata. In questo modo si vuole dare più importanza ai tag delle immagini realmente simili e affidabili. Applicando un primo taglio di tipo statistico (linea rossa-fig 4.8), media e varianza, riesco a selezionare i primi tag da associare al mio shot.

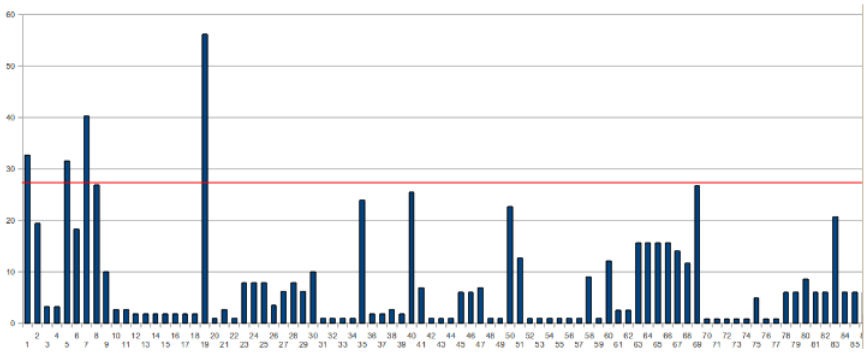

Figura 4.8: Selezione dei tag da associare allo shot. La linea rossa è il primo taglio statistico che viene rapportato. L'istogramma rappresenta solo una parte dei tag votati dalla similarità visuale e affidabilità. In media il numero di tag a questo livello della tag suggestion è di 300-500.

Come spiegato nella sezione *"ImageNet, Google Image, Flickr e Picasa: tipologie di fonti diverse"* tutte le immagini hanno come tag il termine di ricerca, quindi i tag dell'istogramma che riescono a sorpassare questo primo taglio sono i tag del video e dall'espansione semanatica. Può capitare anche che un nuovo tag, proposto quindi da *Flickr e Picasa*, riesca a passare questo taglio ma è veramente difficile. Attraverso questo step abbiamo localizzato temporalmente i tag del video e quelli dell'espansione, formando un primo bagaglio semantico dello shot. In verità avviene già un primo suggerimento dei tag, infatti i tag espansi sono dei tag nuovi, ma a livello statistico vengono trattati come i tag del video anche perchè entrambi sono serviti a costruire il *DataSet*.

4.8.2 Tag suggestion from Flickr and Picasa

Dall'istogramma dei tag vengono eliminati tutti i termini di ricerca: quelli associati allo shot ormai sono stati selezionati e non vogliamo che influenzino ancora la nostra distribuzione, mentre quelli scartati non sono visualmente

simili allo shot. Effettuata questa operazione rimangono solo i tag suggeriti da *Flickr* e *Picasa*. A questo punto ci possiamo trovare in due scenari diversi: una distribuzione con qualche picco sopra la media o una distribuzione piatta. Nel primo caso, mostrato in Fig.4.9, ci sono dei tag che hanno un numero consistente di voti, cioè che sono più volte presenti nei vicini visuali o che hanno ottenuto uno score di similarità alto e che quindi possono essere presi in considerazione per il suggerimento. Nel secondo caso, mostrato in Fig.4.10, nessun tag appartiene a immagini visualmente simili perciò devono essere scartati.

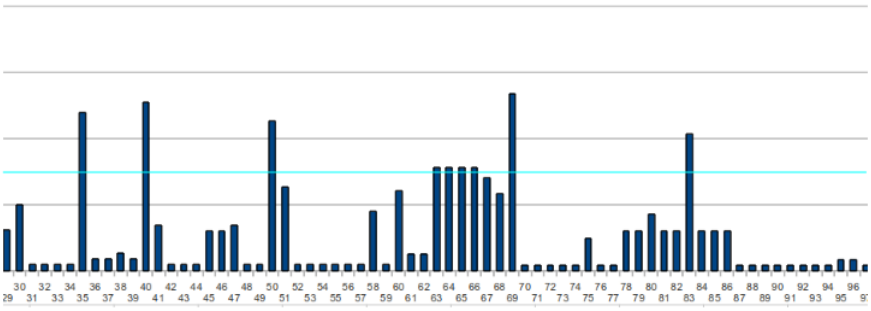

Figura 4.9: Selezione dei tag senza i termini di ricerca. Per ottenere i tag da suggerire viene applicato un taglio (linea azzurra) ottenuto dalla media della distribuzione precedente, vedi fig.4.8, cioè quella con anche i termini di ricerca. Vi è una quantità di tag che sono stati votati sopra la media e che quindi raggiungono la fase di filtraggio.

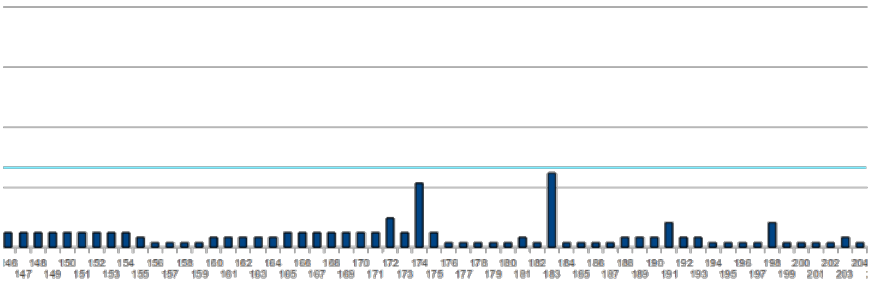

Figura 4.10: Selezione dei tag senza i termini di ricerca. Per ottenere i tag da suggerire viene applicato un taglio (linea azzurra) ottenuto dalla media della distribuzione precedente quella in figura 4.8. Possiamo notare che nessun tag è realmente "vicino" alla nostra immagine quindi vengono scartati tutti.

Inizialmente era stato proposto un taglio statistico calcolato sulla distribuzione senza i termini di ricerca. Questo taglio si adattava alla nuova distribuzione e proponeva sempre dei tag, anche quando ci trovavamo nello scenario dell'immagine 4.10. Per ovviare a questo problema, uno ho deciso di applicare come taglio (linea azzurra) la media della distribuzione dove sono presenti anche i termini di ricerca (vedi 4.8). Utilizzando la media rimangono solo i tag che hanno ricevuto un numero consistente di voti in precedenza e, a questo punto, eliminato il rumore principale, posso applicare il taglio statistico.

I tag rimanenti vengono filtrati nella sezione successiva.

4.8.3 Tag filtering

Il filtraggio dei tag utilizza la misura di *Relatedness* di *Wikipedia Miner*, come visto anche nella sezione *4.4.3*. Il calcolo del punteggio di relazione semantica avviene tra i tag suggeriti nel paragrafo *4.8.2* e i tag associati allo shot nel paragrafo *4.8.1*. Per ogni tag si sommano i punteggi di *Relatedness* con i tag già localizzati nello shot. A differenza del filtraggio che ho proposto nell'espansione semantica, dove per aggiungere un tag allo spazio di partenza

è stata calcolata la *Relatedness* con tutti i tag del video, qui il calcolo avviene a livello locale dello shot. In questo modo prendo in considerazione il bagaglio semantico localizzato nel primo passo e non tutto il bagaglio semantico del video, adattando il suggerimento allo specifico shot. In questo caso ho deciso di utilizza un taglio che non si basasse sulla distibuzione, in modo da non suggerire sempre dei tag. I tag vengono associati allo shot solamente se la loro *Relatedness* media con i tag dello shot supera una soglia *cut*. La soglia *cut* varia in base al numero di tag che etichettano lo shot, più tag etichettanno lo shot e più la soglia si abbassa. Questo accorgimento è stato proposto perchè se il mio shot è etichettato esclusivamente da un tag, e ne voglio proporre uno nuovo, deve esserci una vicinanza stretta e quindi una soglia alta. Invece se lo shot è annotato da tanti tag e ne voglio suggerire uno nuovo, la soglia non può essere troppo restrittiva perchè è difficile che il nuovo tag abbia una forte relazione con tutti, per questo si abbassa la soglia. Il *cut* viene impostato nel seguente modo

```
 if(tag.size()==1)
        cut=0.7;
else
        if(tag.size()<5)
                cut=0.6*tag.size();
        else
                cut=0.55*tag.size();
}
```

I tag che superano il *cut* sono associati allo shot.

Qui di seguito mostro una parte del codice della *Tag Suggestion*. Il vettore *suggesTag* indica i possibili tag da suggerire allo shot, esclusi quelli di partenza, che hanno sorpassato il taglio della media di tutta la distribuzione.

A questi viene applicato il secondo taglio statistico, ottenendo il vettore *nearTag* al quale poi applico il filtro semantico. Alla fine del codice vengono scritti su un file tutti i tag che etichettano lo shot.

```
double suggmedia=0;
for(int i=0;i<suggesTag.size();i++)
        suggmedia=suggmedia+suggesScore.get(i);
suggmedia=suggmedia/suggesTag.size();
double suggscarto=0;
for(int i=0;i<suggesTag.size();i++)
        suggscarto=suggscarto+(suggesScore.get(i)-
        suggmedia)*(suggesScore.get(i)-suggmedia);
suggscarto=scarto/suggesTag.size();
for(int i=0;i<suggesTag.size();i++){
if(suggesScore.get(i)>(suggscarto+suggmedia)){
        nearTag.add(suggesTag.get(i));
        nearScore.add(suggesScore.get(i));
        }
}
SemanticExpansion semantic=new SemanticExpansion();
ArrayList<String> tagsuggested=semantic.getRelatedness(
        shotTag, nearTag, 1);
for(int i=0;i<tagsuggested.size();i++){
if(shotTag.contains(tagsuggested.get(i))!=true)
  shotTag.add(tagsuggested.get(i));
}
for(int i=0; i<shotTag.size();i++){
        out.print(" TAG=\""+shotTag.get(i)+"\"");
```

```
            }
}
```

In fine attraverso il metodo *getShotTag* della classe Shots viene creato un file .srt di sottotitoli. I file .srt sono uno standard che più lettori multimediali possono leggere. La formattazione del file

N
$HH:MM:SS,mmm \rightarrow HH:MM:SS,mmm$
text

dove N è il numero della scena, la durata del sottotitolo è indicata come tempo d'inizio $\rightarrow$ tempo di fine e text rappresenta il testo del sottotitolo.

Questo file viene costruito per presentare i tag degli shot mentre uno guarda il video, così è possibile capire se sono localizzati correttamente. In particolare i tag di partenza sono in maiuscoalo mentre i tag suggeriti in minuscolo.

Capitolo 5

Analisi dei risultati

In questo capitolo descriverò i *DataSet* scelti per effettuare i test e valuterò la qualità dei risultati per avere una vista globale del funzionamento. I *DataSet* utilizzati sono due: YouTube60[1] e YT-22Concepts[2] Il primo *DataSet* viene utilizzato per controllare l'accuratezza delle annotazioni proposte a livello shot e video, mentre il secondo per testare l'applicazione a livello di *concept learning* o *category learning*,

5.1 Test 1: annotazione video

Il *DataSet* YouTube60 è stato costruito scegliendo 4 video per ognuna delle 15 categoria di YouTube fig.5.1, ottenendo così un *DataSet* composto da 60 video che cerca di essere rappresentativo delle varie categorie di video. Da questo insieme di 60 video ho estratto un video per ogni categoria e, per

[1] http://www.micc.unifi.it/ballan/research/tag-webvideos/

[2] http://users.dsic.upv.es/ iaprtc5/data/YT-22Concepts.tgz

ciascuno di essi, ho proposto una tabella riassuntiva dove sono presenti anche i tag suggeriti dalla migliore configurazione. Propongo anche una seconda tabella dove mostro come variano i risultati al variare del *k-nearest*. Il *k-nearest* può assumere tre valori: 1000, 500 e 300. I risultati che specifico nella tabella sono il numero totale di tag nuovi che suggerisco, il numero medio di tag che associo agli shot (tag del video e quelli suggeriti), l'accuratezza a livello shot effettuando il rapporto tra il numero di suggerimenti corretti e il numero globale di suggerimenti sommati su tutti gli shot che compongono il video e l'accuratezza a livello video. Per deteminare l'accuratezza a livello video si calcola semplicemente il rapporto tra i tag corretti e il numero globale di tag suggeriti e si controlla che i tag siano corretti nel contesto visuale del video senza prendere in considerazione la loro localizzazione.

1. “Auto & Vehicles”
2. “Comedy”
3. “Education”
4. “Entertainment”
5. “Film & Animation”
6. “Gaming”
7. “Howto & Style”
8. “Music”
9. “News & Politics”
10. “NoProfits & Activism”
11. “People & Blogs”
12. “Pets & Animals”
13. “Science & Technology”
14. “Sports”
15. “Travel & Events”

Tabella 5.1: 15 categorie dei video di YouTube

Auto & Vehicles

Title:	tractor crash and fun
Durata:	4:45
Video ID:	WlCMfPn021k
Descrittore locale:	SIFT
Numero Shot:	36
Tag Video:	trattori, crash, wrecked, tractors, tractor
Tag Suggeriti	john deere, plane crashes, plane crash
migliore configurazione	accident, truck, incident,
k= 500	train, cars, crashes

Tabella 5.2: riepilogo video categoria Auto & Vehicles

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	9	5.9	0,44	0,67
1000	6	3,14	0,08	0,33
300	11	5,69	0,46	0,63

I tag suggeriti non sono di di buonissima qualità. I termini si dividono in due parti i termini che riguardano i trattori e quelli che riguardano gli incidenti, tra questi “acident”, “incident” e “crashes” sono corretti ma hanno come rumore “plane crashes” e “plane crash”.

Comedy

Titolo:	Talking Cat
Durata:	0:29
Video ID:	964uCtgsDoE
Descrittore locale:	SIFT
Numero Shot:	10
Tag Video:	talking, cat, funny, weird, humour, pet
Tag Suggeriti	humor, pets
migliore configurazione	
k= 500	

Tabella 5.3: riepilogo video categoria Comedy

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	2	4	0,94	1
1000	2	1,8	0,75	1
300	1	3,8	0,98	1

Il tag proposti sono pochi ma corretti, unendo tutti i tag suggeriti al variare di k si ottiene: cats, pets e humor. Nel video è un presente un gatto che ”parla” e nel finale si vedono due gatti, essendo nella categoria Comedy posso dire che i tag proposti rappresentano bene il contenuto del video.

Education

Title:	Graduation Daily 2010: Carnegie - Friday 16th July
Durata:	6:02
Video ID:	azej2fNgwG8
Descrittore locale:	SIFT
Numero Shot:	55
Tag Video:	Leeds, Met, Metropolitan, Graduation, 2010, Students, Graduands, Undergraduate, Postgraduate, Faculty, Carnegie, Sport, Education
Tag Suggeriti	graduate, commencement, campus, university, student,
migliore configurazione	alumni, sports
k=500	

Tabella 5.4: riepilogo video categoria Education

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	7	4,36	0,74	0,87
1000	0	1,37	0,34	0
300	8	5,23	0,69	0,83

I tag iniziali del video sono di buona qualità e numerosi, altrettanto lo sono i tag proposti. Riescono a dare informazioni riguardanti il tipo di evento, la struttura dove è collocato e chi ci partecipa.

Entertainment

Titolo:	Helicopter Crash!!!
Durata:	0:37
Video ID:	e-pZgI3FeJ0
Descrittore locale:	SIFT
Numero Shot:	11
Tag Video:	helicopter, crash, army, accident, airplane
Tag Suggeriti:	
migliore configurazione	
k=300	

Tabella 5.5: riepilogo video categoria Entertainment

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	1	1	0,53	0
1000	2	0,64	0,35	0,5
300	1	2,6	0,62	0

La saggustion in questo video è molto debole infatti propone da uno a due termini, però per k=300 c'è una buona localizzazione dei tag di partenza.

Film & Animation

Titolo:	New Moon Official Trailer #3 (HD)
Durata:	2:28
Video ID:	bs79_5n848Q
Descrittore locale:	SIFT
Numero Shot:	67
Tag Video:	New, Moon, Official, Trailer, #3, HD, Twilight, Saga, MTV
	Robert, Pattinson, Kristen, Stewart, vma, video, music, awards
Tag Suggeriti:	robert pattinson, evening, night, kristen stewart, bella,
migliore configurazione	sky, new moon, sunset, edward cullen,
k=1000	vampire, twilight saga

Tabella 5.6: riepilogo video categoria Film & Animation

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	16	6,24	0,43	0,62
1000	11	2,32	0,39	0,81
300	21	7,34	0,44	0,6

I tag suggeriti sono molto buoni infatti propongono i nomi dei protagonisti per completo e una serie di riferimenti alla saga, tra cui “vampire”.

Gaming

Title:	Need For Speed The Run - Limited Edition Trailer
Durata:	1:37
Video ID:	9t2YtQUun1Q
Descrittore locale:	SIFT
Numero Shot:	52
Tag Video:	Need For Speed,The Run,Limited Edition, Need For Speed The Run,Lamborghini Aventador, Camaro ZL1,Porsche Carrera 911
Tag Suggeriti migliore configurazione k=1000	cars, car, autos, racing, nfs, driving, skyline, drifting, leon, sports, auto, automobile, mx5, automobiles, automotive, rain, carriages, vehicles, vehicle, race, mclaren

Tabella 5.7: riepilogo video categoria Gaming

K-nearest	**Num. Tag Sugg**	**Media Tag per Shot**	**Acc.Shot**	**Acc.Video**
500	27	4.67	0,53	0,67
1000	21	2,56	0,6	0,71
300	30	4,94	0,48	0,67

I tag proposti a livello globale del video sono di buona qualità e interpretano bene il contesto del video. Inoltre il numero di tag suggeriti è sopra la media, quindi è ammesso un pò di rumore.

Howto & Style

Titolo:	ipod nano
Durata:	1:11
Video ID:	rav_WXKpnmg
Descrittore locale:	SIFT
Numero Shot: 16	
Tag Video:	pod, nano
Tag Suggeriti:	apple, iphone, mac, think different
migliore configurazione	
k=300	

Tabella 5.8: riepilogo video categoria Howto & Style

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	5	5,75	0,55	0,4
1000	3	2,44	0,41	0,33
300	4	5,62	0,56	0,5

Il video è la presentazione dell'ipod nano. I tag "apple" e "think different" sono corretti, rappresentano la marca e lo slogan principale mentre i tag "iphone" e "mac" non sono presenti nel video.

Music

Title:	MADONNA Sticky & Sweet Tour - Live in Nice (August 26, 2008)- The Beat Goes On [Professional]
Durata:	2:11
Video ID:	Z94gq-WUIec
Descrittore locale:	SIFT
Numero Shot:	
Tag Video:	Madonna, Sticky, Sweet, Nice, France, The, Beat, Goes, On, HQ, Professional, Recording, live, August, 26, 2008
Tag Suggeriti	mary, maria
migliore configurazione k=300	

Tabella 5.9: riepilogo video categoria Music

K-nearest	**Num. Tag Sugg**	**Media Tag per Shot**	**Acc.Shot**	**Acc.Video**
500	4	2,76	-	0
1000	4	1,76	-	0
300	2	2,68	-	0

Nessuno dei tag proposti nelle tre diverse configurazioni risulta corretto. La migiore configurazione si ottiene con k=300, quella che propone il numero più piccolo di tag sbagliati.

News & Politics

Title:	Korean fire disaster
Durata:	1:49
Video ID:	O6yRPVgRem4
Descrittore locale:	SIFT
Numero Shot:	36
Tag Video:	korea, gas, station, fire, disaster, explosion
Tag Suggeriti:	catastrophe, flame, flames, calamity, accidents, natural
migliore configurazione	
k=500	

Tabella 5.10: riepilogo video categoria News & Politics

K-nearest	**Num. Tag Sugg**	**Media Tag per Shot**	**Acc.Shot**	**Acc.Video**
500	6	5,6	0,66	0,83
1000	11	4,08	0,44	0,56
300	7	6,31	0,64	0,62

I tag suggeriti “catastrophe“ e “calamity“ sono corretti anche se molto simili a ”disaster”. I tag “flame“ e “flames“ che significano ardere, fiamme, vampata amliano maggiormente lo spazio semantico del video. L’unico tag che non è inerente al video è ”natural“ in quanto nel video si vede l’esplosione e poi l’incendio di un stazione di benzina. Molto probabilmente ”natural“ è stato trascinato insieme al tag ”calamity“.

No Profits & Activism

Title:	UNICEF: Child Health Days in Zimbabwe
Durata:	3:06
Video ID:	86n5pvYInHQ
Comedy Descrittore locale:	SIFT
Numero Shot:	36
Tag Video:	UNICEF, Zimbabwe, Child, Health, Days, vaccination, measles, diphtheria, tetanus, polio
Tag Suggeriti:	kid, children, healthcare, healthy
migliore configurazione	
k=500	

Tabella 5.11: riepilogo video categoria No Profits & Activism

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	4	3,97	1	1
1000	2	1,86	0,97	1
300	5	4,64	0,88	0,8

In questo video si vedono le madri in fila con i figli per il vaccino e i babini con i dottori mentre gli viene somministrato. Posso dire che i tag suggeriti sono tutti corretti e anche i tag del video sono corretti, ottenendo un accuratezza sia a livello video che shot alta.

People & Blogs

Title:	Dalai Lama: Inner Peace, Happiness, God and Money
Durata:	1:53
Video ID:	_QvVaZfFDKw
Descrittore locale:	SIFT
Numero Shot:	17
Tag Video:	Dalai, Lama, Renaissance, god, happiness, inner, peace, money
Tag Suggeriti	dalailama, happy
migliore configurazione	
k= 300	

Tabella 5.12: riepilogo video categoria People & Blogs

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	1	4,41	0,72	1
1000	8	3,18	0,65	0,62
300	2	5,88	0,68	1

Nel video è presente il DalaiLama che parla all'interno di una stanza. Con k=300 i tag proposti sono "dalailama" e "happy", che rispecchiano precisamente quello che si vede nel video e anche l'argomento trattato. Con k=1000 i tag proposti sono 8 e tutti appartengano al contesto del video, ma sono di difficile valutazione: "kham", "tibet", "potala", "lhasa", "tibetan", "dalailama", "dharamsala", "his holiness the dalai lama". Viene aggiunto correttamente il tag "dalailama" e il suo nome, "his holiness", ma non sappiamo se la stanza dove è girato il video si trova nel palazzo di Potala e quindi in Tibet.

Pets & Animals

Title:	Jack Russell Puppies
Durata:	1:34
Video ID:	sKEQQJ4VHpY
Descrittore locale:	SIFT
Numero Shot:	34
Tag Video:	jack, russell
Tag Suggeriti:	puppies, dog, puppy, yorkie, dogs, terrier, training,
migliore configurazione	cute, shit
k=500	

Tabella 5.13: riepilogo video categoria Pets & Animals

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	9	5,43	0,74	0,78
1000	7	5	0,7	0,71
300	11	6,03	0,68	0,74

I tag proposti con k=500 sono nove di cui sette rappresentano il contenuto in maniera ottima, mentra “shit“ e “yorkie“ portano rumore a questa suggestion.

Science & Technology

Title:	Agusta A129 Mangusta CBT Mongoose
Durata:	1:15
Video ID:	EM3cdBzKskU
Descrittore locale:	SIFT
Numero Shot:	21
Tag Video:	Agusta, A129, Mangusta, CBT, Mongoose, AgustaWestland, attack, helicopter, Italy, army, aviation, combat, aircraft
Tag Suggeriti:	bell, airplane, westland, air, merlin
migliore configurazione k=500	chopper

Tabella 5.14: riepilogo video categoria Science & Technology

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	6	7,23	0,79	0,5
1000	12	4,38	0,6	0,42
300	4	7,14	0,73	0,5

I tag del video sono numerosi e precisi, per questo motivo l'accuratezza a livello shot è alta. Negli shot sono localizzati soprattutto i tag del video e non quelli suggeriti come è giusto che sia. Infatti i tag del video sono già molto specifici e coprono molto bene l'intero video, rendendo difficile il suggerimento di nuovi tag.

Sports

Title:	Iron Mike Tyson
Durata:	3:28
Video ID:	78ANu_CVGpI
Descrittore locale:	SIFT
Numero Shot:	41
Tag Video:	boxing, 50, cent, mike, tyson
Tag Suggeriti:	knockout, fight, pugilism, boxer, pugilist,
migliore configurazione	mixed martial arts, mma
k=500	

Tabella 5.15: riepilogo video categoria Sports

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	7	4,58	0,75	0,71
1000	5	2,02	0,73	0,6
300	14	5,43	0,74	0,57

Uno dei fattori positivi di questa suggestion è che i termini ”50“ e ”Cent“ che sono rumorosi non influenzano il suggerimento, infatti tutti i tag suggeriti derviano dalla boxing e da Mike Tyson Con k=500 sono proposti sette tag, cinque dei quali rappresentano in modo ottimo il video.

Travel & Events

Title:	Iguazu Falls: the most beautiful waterfalls of the world
Durata:	3:06
Video ID:	6L8845utZI4
Descrittore locale:	SIFT
Numero Shot:	24
Tag Video:	waterfall, Iguazu_Falls, Iguacu, Argentina, Brazil, Travel, Iguazu, Falls
Tag Suggeriti migliore configurazione k=300	brasil, waterfalls, iguacu falls, iguazu falls, nature, pool

Tabella 5.16: riepilogo video categoria Travel & Events

K-nearest	**Num.Tag Nuovi**	**Media Tag/Shot**	**Acc.Shot**	**Acc.Video**
500	2	1,67	0,375	0,5
1000	2	1,5	0,44	0,5
300	6	2,21	0,36	0,83

I tag suggeriti con k=300 rappresentano visualmente il contenuto, con “iguacu falls“ e “iguazu falls“, aggiungono una connotazione generale con ”nature”e ”waterfalls” e il luogo dove è girto il video ”brasil”. L’unico tag che porta rumore è ”pool”. I tag sono ottimi, invece la loro localizzazione non altrettanto.

Effetuando questo test posso concludere i seguenti punti:

1. Se il video è omogeneo e correlato anche con pochi tag ma corretti, i risultati sono più che soddisfacenti.

2. Il sistema riesce ad eliminare il rumore dai tag di partenza, come per esempio nel video di Tyson dove non viene suggerito alcun tag in re-

lazione a 50 Cent o nel video di New Moon dove i tag suggeriti non hanno alcuna relazione con la parte dei tag di partenza relativi a MTV.

3. Il k-nearest migliore è 500, riesce a bilanciare una buona suggestion con la localizzazione a livello shot, con k-nearest 300 spesso viene introdotto troppo rumore e con 1000 la localizzazione è bona ma la suggestion diventa molto debole.

4. A livello di suggestion i termini suggeriti non escono mai dal contesto del video, offrendo un suggerimento ben controllato, ma non sempre riesce ad aggiungere specificità.

5.2 Test 2: categorizzazione video

YT-22Concepts è una collezione di video scaricati da YouTube, ognuno dei quali appartiene ad uno dei 22 concetti elencati nella tabella 5.17 i quali sono stati selezionati manualmente seguendo tre criteri: includere concetti che appartengono a categorie più ampie e diverse (attività, oggetti/animali, luoghi e sport), scegliere concetti che hanno un corrispettivo visuale (escludere concetti astratti come amore, musica, umorismo etc,etc) e controllare che il concetto sia effettivamente presente su *Youtube*.

Da questo insieme ho estrapolato 6 concetti per eseguire i test, uno sport "soccer", un luogo generale "beach", un luogo specifico "eiffel", un oggetto/animale "cats", un'attività "race" e un'azione/sport "swimming". In tutti i video che analizzo il concetto è presente come tag. Per testare se questo metodo riesce a classificare correttamente il video, elimino il concetto dai tag di partenza e controllo che almeno in uno shot del video sia stato riproposto il concetto. Per capire meglio l'obiettivo ultimo di questo test faccio un esempio. Prendo un video di calcio che ha sicuramente tra i suoi tag il termine

basketball	dancing	helicopter	sailing	tank
beach	desert	hiking	secondlife	videoblog
cats	eiffeltower	interview	soccer	
concert	explosion	race	swimming	
crash	golf	roit	talkshow	

Tabella 5.17: 22 concetti contenuti nel DataSet *YT-22Concepts*

"soccer", elimino il termine "soccer" e applico il mio metodo. Alla fine controllo che in almeno una scena del video sia stato riproposto il termine elimiato (cioè che il video sia stato ricatalogato con il suo concetto di partenza).

Nelle tabelle seguenti, per ognuno dei 6 concetti presentati, viene riportato l'ID del video esaminato, il risultato della catalogazione e il numero di shot etichettati con il concetto sul totale degli shot del video.

Concetto "Cats"

Video ID	**Learned**	**NumShot**
lRpNlw-xDLs	yes	25/26
4rrBnpHpG4	yes	26/26

Tabella 5.18: video su cui sono stati applicati il concept learning di "Cats"

Concetto "Soccer"

Video ID	**Learned**	**NumShot**
DQcb-ZkyVP8	no	-
K148M7B77KQ	yes	10/97

Tabella 5.19: video su cui sono stati applicati il concept learning di "Soccer"

Concetto "Beach"		
Video ID	**Learned**	**NumShot**
A4TEGYO_GGM	no	-
T8JrEm7XCuE	yes	9/57

Tabella 5.20: video su cui sono stati applicati il concept learning di "Beach"

Concetto "Eiffel"		
Video ID	**Learned**	**NumShot**
mTeyCYqPIdI	yes	1/4
x9bGOZyaYKg	no	-

Tabella 5.21: video su cui sono stati applicati il concept learning di "Eiffel"

Concetto "Race"		
Video ID	**Learned**	**NumShot**
Go3Fgd1wgic	no	-
UqkJstRAR0Y	no	-

Tabella 5.22: video su cui sono stati applicati il concept learning di "Race"

Concetto "Swimming"		
Video ID	**Learned**	**NumShot**
tExI12HoFvQ	no	-
EwutR1VSUmE	yes	35/35

Tabella 5.23: video su cui sono stati applicati il concept learning di "Swimming"

Il *Concept Learning* ha catalogato in maniera corretta 6 video su 12. Non è possibile parlare di accuratezza in quanto il numero di video è esiguo, però i risultati sono positivi. Il concetto non viene elimiato dai tag dei *Related*

Video, quindi spesso accade che viene reinserito, dopo l'espansione semantica, nei termini che costituiranno il *DataSet*.

Capitolo 6

Conclusione

Questo lavoro di tesi ha affrontato il problema di annotazione automatica di video sulla base dei loro contenuti visuali, localizzando temporalmente i tag nelle scene del video ottenuti sfruttando il *Social Knowledge* presente nelle collezioni di video di Youtube, nelle collezioni di immagini in Flickr e Picasa, nella conoscenza linguistica e visuale presente in ImageNet e nella conoscenza sociale strutturata di Wikipedia. Inoltre propone un metodo libero da qualsiasi forma di apprendimento.

Il lavoro ha analizzato due punti principali:

- ricerca della similarità tra immagini, per trovare, all'interno di un insieme di foto annotate, quelle più simili allo shot di un video
- propagazione dei tag nelle scene sulla base della loro relazione semantica con i termini del video e con la conoscenza sociale del Web

L'espansione semantica dei tag di partenza ottenuta grazie alla misura di *Relatedness* offerta da *Wikipedia Miner*, risponde con buoni risultati, aggiungendo quasi sempre termini corretti anche se in numero limitato. Questa espansione crea un buon punto di partenza perchè amplia lo spazio dei tag

ancora prima di creare il *DataSet*. Per la parte visuale l'approccio utilizza i descrittori locali (SIFT o SURF) e un *Adaptive Vocabulary Tree*, rappresentando le immagini con il metodo *Bag of Visual Word*. Il sistema risulta affidabile per il matching di immagini e con delle buone performance. Il sistema di *Tag Suggestion* realizzato privilegia un suggerimento corretto a discapito del numero di tag suggeriti. Questo tipo di suggerimento tiene conto del bagaglio semantico degli shot, senza mai prendere in considerazione il video nella sua totalità. Vengono suggeriti nuovi tag sulla base della similartà visuale con i frame ma anche sulla base delle relazioni semantiche con i tag di partenza del video localizzati nello shot.

I risultati ottenuti sono molto soddisfacenti dato che la quasi totalità dei tag suggeriti risulta corretto o almeno cade nel contesto del video, anche se la loro valutazione è difficoltosa. Infatti non è possibile individeuare un sistema automatizzato che ne valutasse la correttezza.

Devo però specificare che per alcune classi di *Youtube* è quasi impossibile proporre dei tag che hanno un corrispettivo visuale o che sono realmente collegate al contesto del video. Un esempio su tutti quello dei video musicali, dove gli utenti annotano il video con il nome del cantante e il titolo della canzone ed a quel punto è impossibile risalire ai suoi contenuti visuali. Per altre classi invece non esiste un background sociale dove ricercare i termini da aggiungere al video. Questo è il caso per branche della conoscenza che non sono state approfondite a livello sociale oppure per video che costituiscono una relatà a se. Se il video è omogeneo e presenta almeno tre o quattro tag corretti in partenza i risultati sono davvero soddisfacenti.

In conclusione, posso affermare che l'utilizzo di tag visualmente simili e l'utilizzo del *Social Knowledge* aumenta la qualità dei tag suggeriti ad un video.

Bibliografia

[1] L. Ballan, M. Bertini, A. Del Bimbo, M. Meoni, and G. Serra. Tag suggestion and localization in user-generated videos based on social knowledge. 2010.

[2] X. Li, C.G.M. Snoek, and M. Worring. Learning tag relevance by neighbor voting for social image retrieval. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 180–187. ACM, 2008.

[3] T. Yeh, J. Lee, and T. Darrell. Adaptive vocabulary forests br dynamic indexing and category learning. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[4] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. 2005.

[5] T. Darrell, K. Grauman, T. Darrell, et al. Approximate correspondences in high dimensions. 2006.

[6] S. Choudhury, J. Breslin, and A. Passant. Enrichment and ranking of the youtube tag space and integration with the linked data cloud. *The Semantic Web-ISWC 2009*, pages 747–762, 2009.

[7] M. Brown and D.G. Lowe. Invariant features from interest point groups. In *British Machine Vision Conference, Cardiff, Wales*, pages 656–665. Citeseer, 2002.

[8] J. Sivic and A. Zisserman. Video google: Efficient visual search of videos. *Toward Category-Level Object Recognition*, pages 127–144, 2006.

[9] Sara Fioravanti and Francesca Curradi. Learning social tag relevance and tag propagation in user-generated content (flickr & youtube) francesca curradi francesca curradi. Master's thesis.

[10] D. Milne. Computing semantic relatedness using wikipedia link structure. In *Proceedings of the New Zealand Computer Science Research Student Conference (NZCSRSC), Hamilton, New Zealand*. Citeseer, 2007.