Caranx: Scalable Social Image Index Using Phylogenetic Tree of #Hashtags

Yusheng Xie, Zhuoyuan Chen, Ankit Agrawal, Wei-keng Liao, Alok Choudhary

SC13 Denver, CO | 2013 — HPC Everywhere. — Association for Computing Machinery — IEEE computer society

Introduction

- Photos are most shared items on Facebook: 300M+ are uploaded daily.
- Scalable photo indexing is challenging:
  - Depends heavily on **SIFT** features
  - SIFT features not robust in practice
- **NRDC** has robust feature, but not scalable.
  - Features extracted from pairwise algorithm (see center section for details)
- Caranx uses social tag to make **NRDC** scalable and enable *Scalable* image search
  - Photos on social networks are ***#tagged***.
- Visit our project @ https://github.com/yvesx/caranx

Algorithms

Algorithm 1: Indexing social images with CARANX

Input: I, a set of images with social #tags
Output: M, index data structure for I
Initialize T, set to hold all #tags
Initialize D, matrix to hold #tag pairwise distances
Initialize M, index data structure for I
Initialize PT, phylogenetic tree to structure all #tags
for *image* $i \in I$ **do**
  for *#tag* t *assc. to* i **do**
    Append i to $T[t]$
  end
end
for *pair of #tags* $\langle t, v \rangle \in T \times T$ **do**
  $D(t, v) \leftarrow |T[t] \cap T[v]|$
end
$PT \leftarrow \text{UPGMA}(D)$
for *#tag* $t \in T$ **do**
  $q \leftarrow$ empty queue; Add t to q; $O \leftarrow$ empty set
  $p_1 \leftarrow 1.0; p_2 \leftarrow 1.0$
  while $q \neq \emptyset$ **do**
    $h \leftarrow q$.remove()
    Mark h in PT as visited
    for *image* $i \in T[h]$ **do**
      Add i to O with probability p_1
    end
    Decrease p_1, p_2 by $f(h.\text{Edges})$
    for *#tag* $n \in h$*'s unvisited neighbors* **do**
      Mark n in PT as visited
      Add n to q with probability p_2
    end
  end
  for *images* $\langle i, j \rangle \in O \times O$ **do**
    $W \leftarrow$ visual patches from NRDC(i, j)
    for *patch* $w \in W$ **do**
      $m \leftarrow$ minHash(quantized w)
      Initialize $M[t][m]$ if not exists
      Append i, j to $M[t][m]$
    end
  end
end
return M

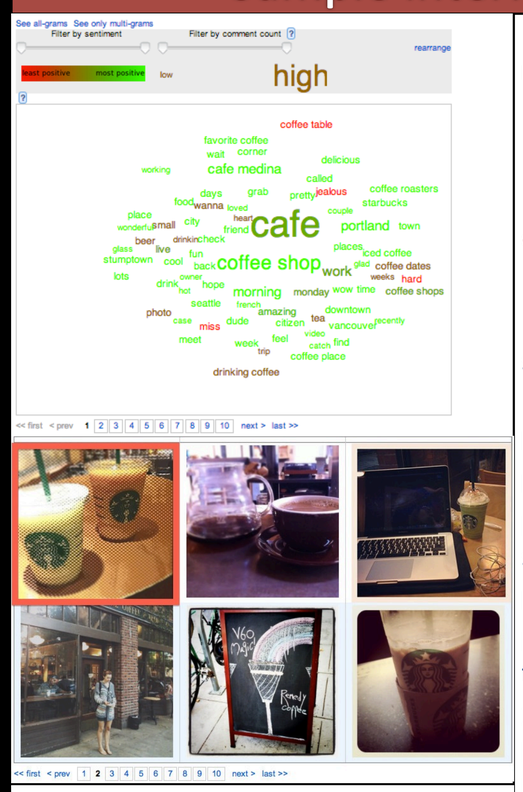
Algorithm 2: Searching social images with CARANX

Input: i, query image with social #tags; T, all #tags; M, image index
Output: R, result set
Initialize R, set to hold return images
for *#tag* t *assc. to* i **do**
  $p \leftarrow 1.0$
  Decrease p if $t \notin T$
  for m *in* $M[t]$ **do**
    Skip with probability $(1 - p)$
    if s *is significant* **then**
      for $j \in M[t][m]$ **do**
        Append j to R
      end
    end
  end
end
return R

INDEXING
Non-Rigid Dense Correspondence for images
Phylogenetic Trees for #tags

QUERYING
Probabilistic BFS
Bag of dense visual keywords

Sample interface



Query image shown in red box:

a *#Starbucks* photo

<<<<<< #tag Explorer

#tags related to the query and #starbucks

Font size ~ relevance

Color spectrum ~ social sentiment

<<<<<< Retrieval Results

Diversity due to probabilistic Tree search

Relevance thanks 1) phylogenetic tree of tags

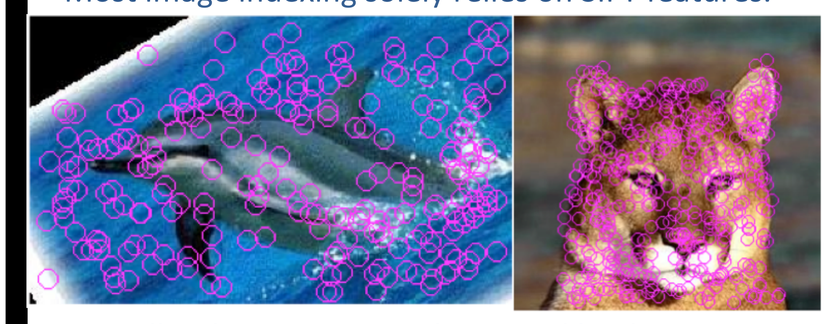
2) Dense visual words like Starbucks Mermaid logo.

<<<<<< #kobe

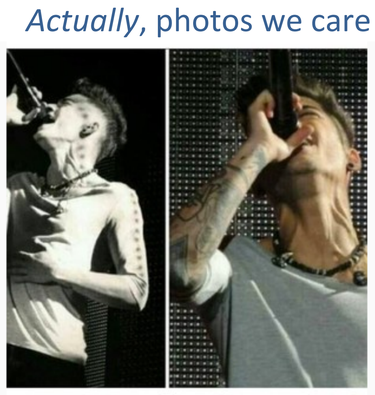
Different query can result in an entirely different subset of #tags from the phylogenetic tree.

Background

Most image indexing solely relies on SIFT features.



Actually, photos we care about are like these:



❤ 1 💬 1

#beautiful, #cute, #niall, #love, #liampayne, #harry, #1direction, #louistomlinson, #directioner, #zayn, #louis, #niallhoran, #sexy, #onedirection, #guys, #liam, #boys, #zjmalik, #iphonesia, #instagood, #tagsforlikes, #hot, #harrystyles, #1d, #leeyum, #zaynmalik, #photooftheday, #happy,

♥ @_dem_accents_,

28 #tags, **1**@

❤ 9 💬 2

#cute, #love, #adorable, #tbt, #misshim, #so, #lovehim, #cousin, #miss, #him, #braces, #happy,

♥ @shortiie4life, @_itsemiiily, @_abbylovespenguins10, @kaylahodea,

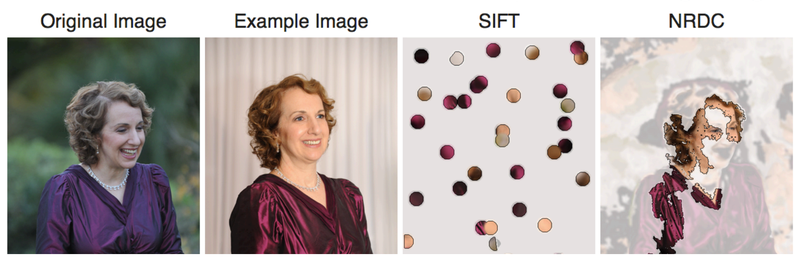
12 #tags, **4**@

#tags are part of the images, just like colors and pixels.

Non-Rigid Dense Correspondence

NRDC: advanced patch-matching algorithm for image.

Pros: better performance over SIFT in real world scenarios like differences in facial expressions, lighting.



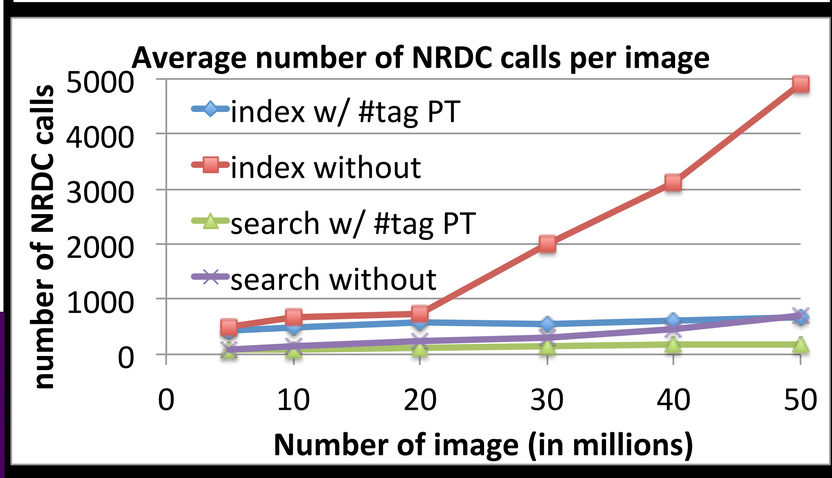
NRDC can identify critical visual frequent keywords like the Starbucks Mermaid logo.

Cons: NRDC operates on pairs of images. Suppose we are indexing 1B img, 1B*1B=*5e17* pairwise computations are needed – not scalable.

Data and Preliminary Results

Table 1: Caranx experiment data as of 2013 June.

Source	Instagram public data
Images	54.8M
#tags (unique)	~200M (~5M)
Unique users	15.4M



Phylogenetic Tree of #tags

Use a **phylogenetic tree (PT)** to organize #tags and enable search over them.

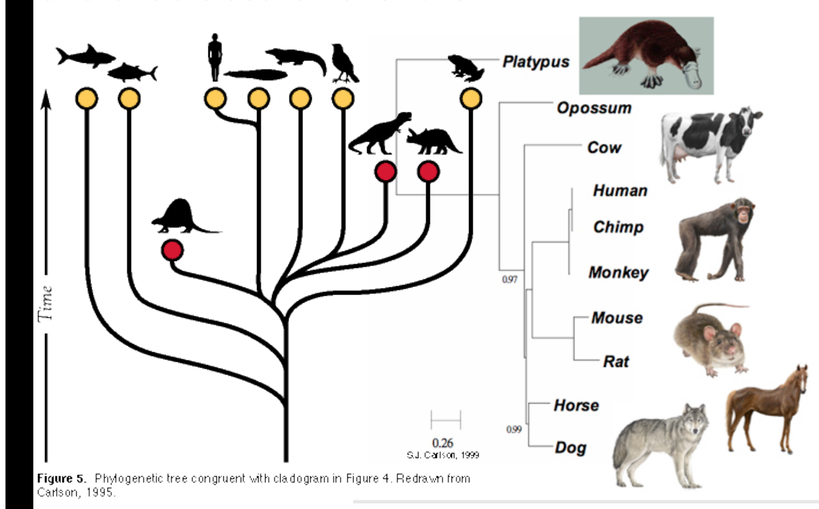


Figure 5. Phylogenetic tree congruent with cladogram in Figure 4. Redrawn from Carlson, 1995.

#tags are like species: they thrive and die out as time evolves.

#summer is popular during May~Aug

#bostonStrong quickly gains&loses presence

#happyNewYear reoccurs every new year.

NRDC+PT=*Credibility*+Scalability

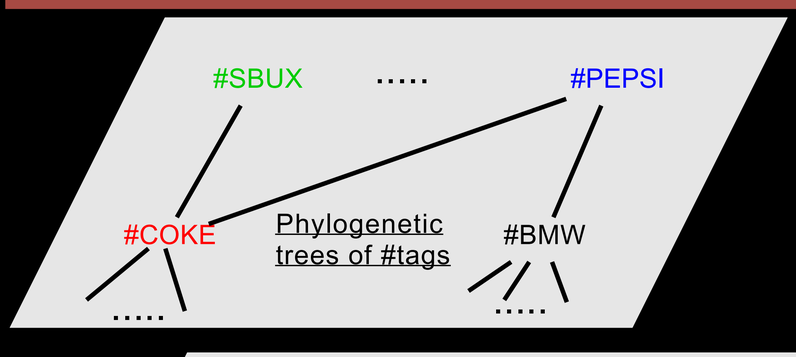
INDEXING

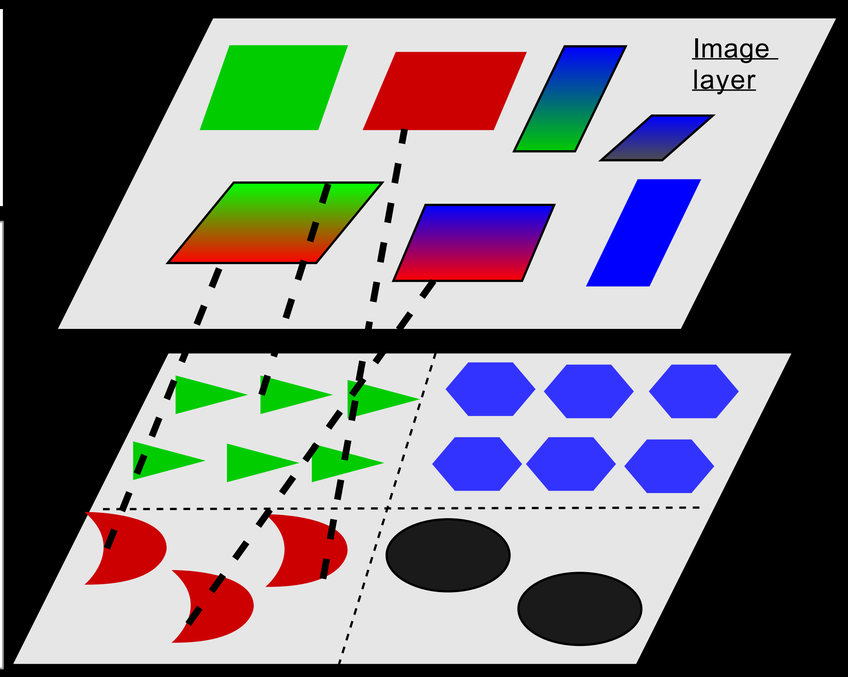
1. Organize #tags and associated images into a phylogenetic tree. #tag pairwise distance matrix D:
   D(t,v) = |images w/ #t| ∧ |images w/ #v|
   Use UPGMA algorithm to compute phylogenetic tree from D.
2. Perform pairwise NRDC to images under each close sub-branch (each sub-branch probabilistically contains 2~10 #tags.).
3. Quantize the dense patches (visual keywords) discovered from NRDC. Store the patches and their minHash signatures.

QUERYING

1. Use query #tag to quickly narrow down branches to search.
2. Match query image with registered patches using NRDC.
3. Return result images based on matched patches.

Caranx Overview





Adobe

voxsup

NORTHWESTERN UNIVERSITY