Learning A Semantic Space: From Image Annotation to Music Similarity

Samy Bengio, Google
with co-authors
Jason Weston, Nicolas Usunier, Philippe Hamel

International Workshop on Document Analysis Systems
March 27-29, 2012

PART I. Image Annotation: What is it?

Goal: Label a new image using a predefined set of possible annotations.

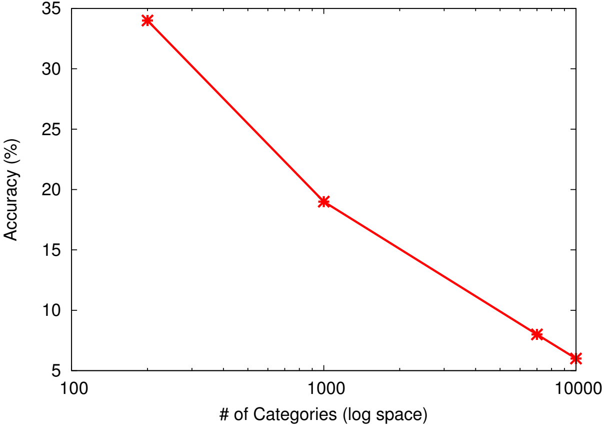
→ **obama**

→ **eiffel tower**

- Computer vision literature has mostly focused on getting better features to represent images.
- The number of possible annotations (dictionary) is usually small (from 20 to 1000 or even 10,000 very recently).
- In this work, we consider dictionaries of size 100,000 and more.

Size Matters!

(Figure inspired by Fei Fei Li)



Despite several research advances, performance of best systems degrades significantly as the number of possible categories grows.

Datasets (to grasp the scale)

Statistics	ImageNet	Web
Number of Training Images	2,518,604	9,861,293
Number of Test Images	839,310	3,286,450
Number of Validation Images	837,612	3,287,280
Number of Labels	15,952	109,444

About our version of Imagenet

This was taken from the website about 2 years ago, but since then nobody has published anything using the whole dataset...

Classical Approach To Image Annotation

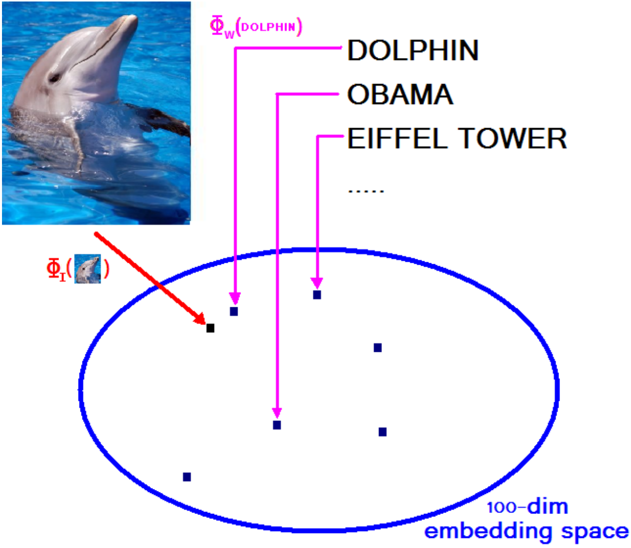
Feature Extraction

1. Interest point detection: which points in the image should we analyze.
2. Feature extraction: how do we represent each point. Examples: color histograms, edges (SIFT, HoG).
3. Aggregation of features: from a dictionary of commonly seen features, count how many of each common feature was in the image.

Model Training

1. Gather a large training set of labeled images.
2. Extract features for each training image.
3. Train a classifier for each label (so-called one-vs-rest).
4. Example of an often-used classifier: Support Vector Machine.
5. Does not scale well...

Our Proposed Solution: Wsabie



Learn $\Phi_I(\cdot)$ ***and*** $\Phi_W(\cdot)$ ***to optimize precision@k.***

Joint Word-Image Embedding Model

Images: $d = 10,000$ dimensional sparse "visterms". Learn map:

$$\Phi_I(x) = Vx : \mathbb{R}^d \to \mathbb{R}^D.$$

Annotations: Y possible annotations, indexed by i. Learn map:

$$\Phi_W(i) = W_i : \{1, \dots, Y\} \to \mathbb{R}^D.$$

Our model compares the degree of match between the image and annotations in the embedding space:

$$f_i(x) = sim(\Phi_W(i), \Phi_I(x)) = W_i^\top Vx$$

We also constrain the weights (regularize):

$$||V_i||_2 \leq C, \quad i = 1, \dots, d, \qquad ||W_i||_2 \leq C, \quad i = 1, \dots, Y.$$

To Label an Image is Equivalent to a Ranking Problem

- Label an image means selecting a few relevant labels from a large set of potential labels.
- That amounts to ranking (ordering) labels given the image.
- Learning-To-Rank is a known setting in machine learning.
- Classical approach to learning-to-rank:
  - for each image x,
  - for each proper label for that image y,
  - and for each wrong label for that image $\bar{y}$:
  - make sure the distance between x and y is smaller (by a margin) than the distance between x and $\bar{y}$.

Ranking Annotations: AUC is Suboptimal

Classical approach to learning to rank is maximize AUC by minimizing:

$$\sum_{x} \sum_{y} \sum_{\bar{y} \neq y} |1 + f_{\bar{y}}(x) - f_y(x)|_+$$

A scalable version of this is via stochastic gradient descent (SGD): sample triplets $(x, y, \bar{y})$ and make a gradient step on the hinge loss.

Problem: All pairwise errors are considered the same.

Example:

Function 1: true annotations ranked 1st and 101st.

Function 2: true annotations ranked 50st and 52st.

AUC prefers these *equally* as both have 100 "violations".

We want to optimize the top of the ranked list!

Ordered Weighted Pairwise Classification (OWPC) Loss

A class of ranking error functions recently defined in [Usunier et al. '09]:

$$err(f(x), y) = L(rank_y(f(x))),$$

where

$$L(k) = \sum_{j=1}^{k} \alpha_j, \text{ with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0.$$

and $rank_y(f(x))$ is the rank of the true label y given by $f(x)$:

$$rank_y(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) \geq f_y(x))$$

Different choices of $L(\cdot)$ have different minimizers:

$\alpha_j = \frac{1}{Y-1} \rightarrow$ minimize mean rank

$\alpha_j = \frac{1}{j} \rightarrow$ more weight on optimizing the top of list.

Example from before: $\alpha_j = \frac{1}{j} \rightarrow$ err(func1)=5.18, err(func2)=8.99.

SVM$_{struct}$ with OWPC = State-of-art on small text retrieval tasks.

Weighted Approximate-Rank Pairwise (WARP) Loss

Problem: we would like to apply SGD:

$$err(f(x), y) = L(rank_y^1(f(x))), \qquad rank_y^1(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) + 1 \geq f_y(x))$$

...but this is expensive to compute per (x, y) sample when Y is large.

Solution: approximate by sampling $f_i(x)$ until we find a violating label $\bar{y}$

$$rank_y^1(f(x)) \approx \left\lfloor \frac{Y-1}{N} \right\rfloor$$

where N is the number of trials in the sampling step.

Online WARP Loss

Input: labeled data (x_i, y_i), $y_i \in \{1, \ldots, Y\}$.
repeat
  Pick a random labeled example (x_i, y_i)
  Set $N = 0$.
  repeat
    Pick a random annotation $\bar{y} \in \{1, \ldots, Y\} \setminus y_i$.
    $N = N + 1$.
  until $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ or $N > Y - 1$
  if $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ **then**
    Make a gradient step to minimize:

$$L(\lfloor \frac{Y-1}{N} \rfloor)|1 - f_y(x) + f_{\bar{y}}(x)|_+$$

  end if
until validation error does not improve.

Other Approaches

Methods Compared:

- One-Vs-Rest: $f_i(x) = w_i \cdot x$ - trained with Hinge loss.
- Multiclass: $f_i(x) = w_i \cdot x$ - trained with AUC [Grangier & Bengio, '08].
- Approximate k-NN - speed/accuracy trade-off:
  we tried: bal. tree of depth p, calc distance of all $\frac{n}{2^p}$ points.

Other Related Work

- Unsupervised text embedding, e.g. LSI, pLSI, LDA, etc.
- Supervised text embedding: e.g. [Bai et al. '09]
- Optimizing Precision@k/MAP for text: e.g. ListNet [Cao et al. '07], SVM_{map} [Yu et al., '07], LambdaRank [Burges et al., '07] and more.

Test Set Performance Results

On ImageNet

Algorithm	p@1	p@10	MAP
Approx. k-NN	1.55%	0.41%	2.32%
One-vs-Rest	2.27%	1.02%	5.17%
Multiclass	3.14%	1.26%	6.43%
Wsabie	4.03%	1.48%	7.75%

On Web Images

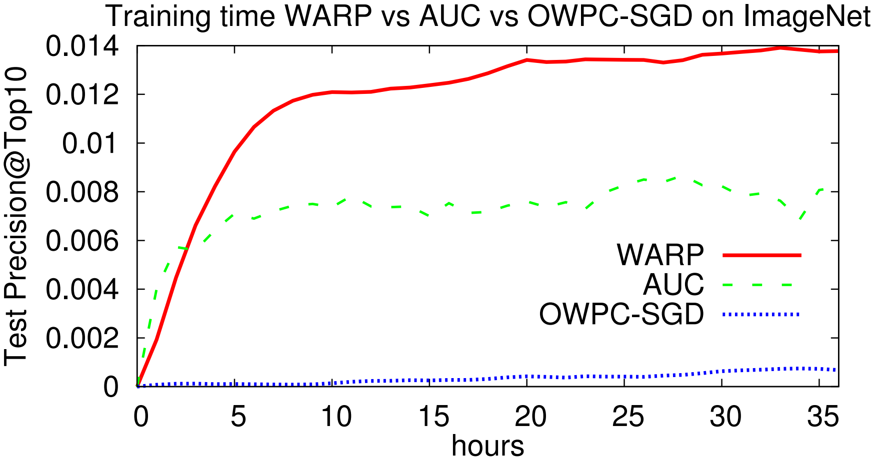
Algorithm	p@1	p@10	MAP
Approx. k-NN	0.30%	0.34%	1.52%
One-vs-Rest	0.52%	0.29%	1.45%
Multiclass	0.32%	0.16%	0.83%
Wsabie	1.03%	0.44%	2.27%

WARP vs. AUC optimization

For each model choice, WARP consistently improves over AUC

Model	Loss	p@1	p@10
Dataset: ImageNet			
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	AUC	1.65%	0.91%
	WARP	**4.03%**	**1.48%**
$f_i(x) = w_i \cdot x$	AUC	3.14%	1.26%
	WARP	**4.25%**	**1.48%**
Dataset: Web			
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	AUC	0.19%	0.13%
	WARP	**1.03%**	**0.44%**
$f_i(x) = w_i \cdot x$	AUC	0.32%	0.16%
	WARP	**0.94%**	**0.39%**

Training time: WARP vs. OWPC-SGD & AUC



Test Time and Memory Constraints

Test Time and Memory requirement needed to return the top ranked annotation on the test set of Imagenet and Web, not including feature generation.

Algorithm	ImageNet		Web	
	Time	Space	Time	Space
k-NN	255 days (26.2s)	6.9 GB	3913 days (103s)	27.1 GB
Approx. kNN	2 days	7 GB	-	-
One-vs-Rest	17 h (0.07s)	1.2 GB	19 days (0.5s)	8.2 GB
Multiclass	17 h	1.2 GB	19 days	8.2 GB
Wsabie	5.6 h (0.02s)	12 MB	6.5 days (0.17s)	82 MB

Changing the Embedding Size on ImageNet

Test error metrics when we change the dimension D of the embedding space used in Wsabie.

Embedding Dim.	p@1	p@10	MAP
100	3.48%	1.39%	7.12%
200	3.91%	1.47%	7.66%
300	4.03%	1.48%	7.75%
500	3.95%	1.44%	7.58%

Training an Ensemble of WSABIEs

Ensemble learning is known to improve performance.

Several WSABIEs can be trained and combined, giving improved performance, but still give a reasonably low memory usage + fast model.

Model	p@1	p@10	MAP
Approx. k-NN	1.55%	0.41%	2.32%
One-vs-Rest	2.27%	1.02%	5.17%
Multiclass	3.14%	1.26%	6.43%
Wsabie	4.03%	1.48%	7.75%
Wsabie Ensemble (2 models)	5.74%	1.97%	10.17%
Wsabie Ensemble (3 models)	6.14%	2.09%	11.23%

Using Better Features..

This paper is not about feature representations.
But, clearly better features lead to better performance.

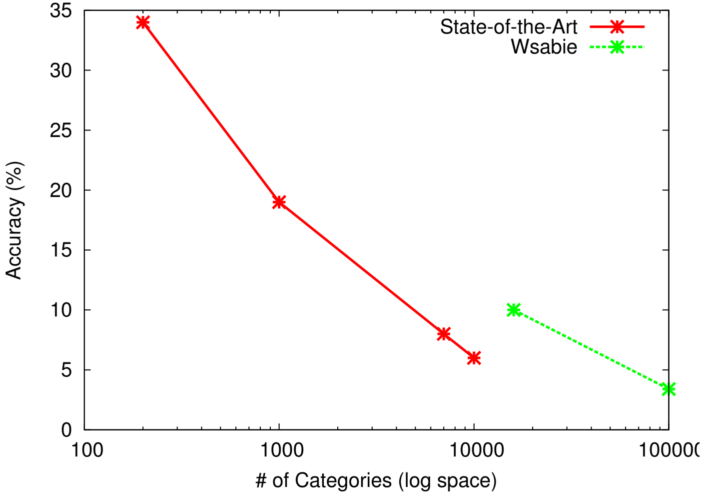
ImageNet: bag-of-words "visterms" features

Algorithm	p@1	p@10	MAP
Wsabie	4.03%	1.48%	7.75%
Wsabie Ensemble (3 models)	6.14%	2.09%	11.23%

ImageNet: visterms + position + others

Algorithm	p@1	p@10	MAP
Exact Nearest Neighbor	7.73%		
Wsabie	8.83%	2.71%	14.97%
Wsabie Ensemble (3 models)	9.82%	2.88%	16.24%
Wsabie Ensemble (10 models)	10.03%	3.02%	17.02%

Size Matters - Revisited



Learned Annotation Embedding (on Web Data)

Annotation	Neighboring Annotations
barack obama	*barak obama*, obama, barack, barrack obama, bow wow
david beckham	*beckham*, *david beckam*, alessandro del piero, *del piero*
santa	*santa claus*, *papa noel*, *pere noel*, santa clause, joyeux noel
dolphin	delphin, dauphin, *whale*, *delfin*, *delfini*, *baleine*, blue whale
cows	*cattle*, *shire*, dairy cows, kuh, *horse*, cow, *shire horse*, *kone*
rose	rosen, *hibiscus*, *rose flower*, rosa, roze, pink rose, *red rose*
pine tree	*abies alba*, abies, *araucaria*, pine, neem tree, *oak tree*
mount fuji	mt fuji, fuji, fujisan, fujiyama, *mountain*, *zugspitze*
eiffel tower	*eiffel*, tour eiffel, la tour eiffel, *big ben*, *paris*, blue mosque
ipod	i pod, *ipod nano*, apple ipod, ipod apple, new ipod
f18	f 18, eurofighter, f14, fighter jet, tomcat, mig 21, f 16

Image Annotation Examples: Dolphin

Wsabie: delfini, orca, dolphin, mar, delfin, dauphin, whale, cancun, killer whale, sea world

One-Vs-Rest: surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, mahi mahi

Wsabie: blue whale, whale shark, great white shark, underwater, white shark, shark, manta ray, dolphin, requin, blue shark, diving

One-Vs-Rest: freediving, blau, deep sea, azul, caretta caretta, manta ray, leopard seal, taucher, dolphin, underwater scene, business background

Image Annotation Examples: Obama & Eiffel Tower

Wsabie: barrack obama, barak obama, barack hussein obama, barack obama, james marsden, jay z, obama, nelly, falco, barack

One-Vs-Rest: falco, barack, daniel craig, obama, barack obama, kanye west, pharrell williams, 50 cent, barrack obama, bono, smoking

Wsabie: eiffel, paris by night, la tour eiffel, tour eiffel, eiffel tower, las vegas strip, eifel, tokyo tower, eifel tower

One-Vs-Rest: tour eiffel, eiffel tower, eiffel, la tour eiffel, paris by night, paris france, advent, paris, warhammer

Image Annotation Examples: Ipod

Wsabie: ipod, ipod nano, nokia, i pod, nintendo ds, nintendo, lg, pc, nokia 7610, vino

One-Vs-Rest: wine, ipod, i pod, zippo, brochure, moleskine, nintendo ds, book, nokia, ipod classic

Wsabie: radioactive, ipod ad, post it, smiley, yellow, smiley face, smile, iowa hawkeyes, a style, caution, soda stereo, kill bill, idance

One-Vs-Rest: pacman, pac man, a style, amarillo, smiley face, smile, enjoi, gelb, radioactive, be happy, yellow caution, soda stereo

PART II: Music Similarity With Wsabie

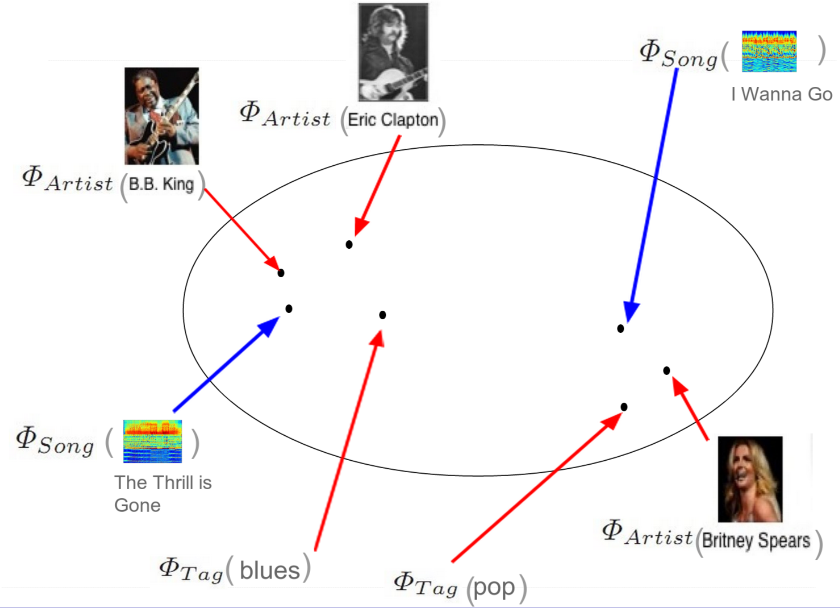
Organizing the world's music information:

- Similar artists: given an artist name, provide a list of similar artists.
- Similar song: given a song track (audio), provide a list of similar songs.
- Genre prediction: given a song track (audio), label with appropriate genre(s).
- Any other combination between these 3 types of data.

Can we use Wsabie to do so?

- Jointly embed audio representations of tracks, artist names, genres.

Wsabie For Music



Wsabie For Music

Mapping for a given artist name:

$$\Phi_{Artist}(i) : \{1, \ldots, |\mathcal{A}|\} \rightarrow \mathbb{R}^d = A_i.$$

Mapping for a given genre:

$$\Phi_{Genre}(i) : \{1, \ldots, |\mathcal{G}|\} \rightarrow \mathbb{R}^d = G_i.$$

Mapping for the audio content of a given song:

$$\Phi_{Song}(s') : \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^d = Vs'.$$

We also constrain the weights (regularize):

$$||A_i||^2 \leq C, \quad ||G_i||^2 \leq C, \quad ||V_i||^2 \leq C.$$

Multi-Tasking Wsabie

Artist prediction: the artists are ranked according to the magnitude of $f_i(x)$, largest first:

$$f_i^{ArtistPred}(s') = \Phi_A(i)^\top \Phi_S(s') = A_i^\top V s'$$

Similarly, for song or genre prediction and similar artists or songs:

$$\begin{aligned}
f_{s'}^{SongPred}(i) &= \Phi_S(s')^\top \Phi_A(i) &= (Vs')^\top A_i \\
f_j^{SimArtist}(i) &= \Phi_A(j)^\top \Phi_A(i) &= A_j^\top A_i \\
f_{s'}^{SimSong}(s'') &= \Phi_S(s')^\top \Phi_S(s'') &= (Vs')^\top V s'' \\
f_i^{GenrePred}(s') &= \Phi_G(i)^\top \Phi_S(s') &= G_i^\top V s'
\end{aligned}$$

We can multi-task all these tasks sharing parameters: *map all problems to the same semantic embedding space.*

Train with WARP as before.

Feature Representation

- **MFCC** These are the same features used in speech recognition. We extract 13 MFCC every 10ms, with 1st and 2nd derivative. We computed a dictionary of 2000 typical MFCCs and represent each track as a vector of counts of the number of times each typical MFCC was seen in the track.
- **SAI** These were developed at Google: Stabilized Auditory Image features.

Other Approaches

Methods Compared:

- One-Vs-Rest: $f_i(x) = w_i \cdot x$ - trained with Hinge loss on our features.
- All the entrants of the TagaTune Competition.
- Cosine similarity for song similarity.

Other Related Work

- Unsupervised text embedding, e.g. LSI, pLSI, LDA, etc.
- Supervised text embedding: e.g. [Bai et al. '09]
- Optimizing Precision@k/MAP for text: e.g. ListNet [Cao et al. '07], SVM$_{map}$ [Yu et al., '07], LambdaRank [Burges et al., '07] and more.

Music Datasets

Summary statistics of the datasets.

Statistics	TagATune	Big-data
Number of Training Songs/Clips	16,289	771,901
Number of Test Songs	6498	185,994
Number of Style Labels	160	-
Number of Artist Labels	-	26,972

The TagaTune dataset was used in a recent competition, we can compare to those methods.

Results on TagATune

Summary of Test Results

Approach	Features	p@3	p@6	p@9	p@12	p@15
Zhi	mfcc	0.224	0.192	0.168	0.146	0.127
Manzagol	mfcc	0.255	0.194	0.159	0.136	0.119
Mandel	cepstral + temporal	0.323	0.245	0.197	0.167	0.145
Marsyas	spectral + mfcc	0.440	0.314	0.244	0.201	0.172
1-vs-rest	mfcc	0.349	0.244	0.193	0.154	0.136
Wsabie	mfcc	0.382	0.275	0.219	0.182	0.157
1-vs-rest	mfcc + sai	0.362	0.261	0.221	0.167	0.151
Wsabie	mfcc + sai	0.473	0.330	0.256	0.211	0.179

Results on TagATune

Related tags in the embedding space learnt by Wsabie ($d = 400$, using features mfcc+sai) on the TagATune data. We show the closest five tags (from the set of 160 tags) in the embedding space.

Style Tag	Neighboring Style Tags
female opera	opera, operatic, woman, male opera, female singer
hip hop	rap, talking, funky, punk, funk
middle eastern	eastern, sitar, indian, oriental, india
flute	flutes, wind, clarinet, oboe, horn
techno	electronic, dance, synth, electro, trance
ambient	new age, spacey, synth, electronic, slow
celtic	irish, fiddle, folk, medieval, female singer

Results on TagATune

Changing the Embedding Size on TagATune. Test Error metrics when we change the dimension d of the embedding space used in Wsabie.

Algorithm	Features	p@3	p@6	p@9	p@12
Wsabie ($d = 100$)	mfcc	0.371	0.267	0.212	0.177
Wsabie ($d = 200$)	mfcc	0.379	0.273	0.216	0.180
Wsabie ($d = 300$)	mfcc	0.381	0.273	0.217	0.181
Wsabie ($d = 400$)	mfcc	0.382	0.275	0.219	0.182
Wsabie ($d = 100$)	mfcc+sai	0.452	0.319	0.248	0.205
Wsabie ($d = 200$)	mfcc+sai	0.465	0.325	0.252	0.208
Wsabie ($d = 300$)	mfcc+sai	0.470	0.329	0.255	0.209
Wsabie ($d = 400$)	mfcc+sai	0.473	0.33	0.256	0.211
Wsabie ($d = 600$)	mfcc+sai	0.477	0.334	0.259	0.212
Wsabie ($d = 800$)	mfcc+sai	0.476	0.334	0.259	0.212

Results on Web-Data

Summary of Test Set Results on Big-data.

Algorithm	Artist Prediction		Song Prediction		Similar Songs	
	p@1	p@6	p@1	p@6	p@1	p@6
one-vs-rest^{ArtistPrediction}	0.087	0.036	-	-	-	-
cosine similarity	-	-	-	-	0.054	0.021
Wsabie^{SingleTask} (d=100)	0.091	0.034	0.099	0.056	0.040	0.020
Wsabie^{AllTasks} (d=100)	0.107	0.038	0.123	0.069	0.056	0.026
Wsabie^{AllTasks} (d=400)	0.125	0.041	0.133	0.073	0.065	0.028

PART III: Document Similarity With Wsabie

Large Scale Wsabie Experiments

- Using hundreds of millions of text documents.
- Each document is a bag-of-words with a dictionary of 1M words.
- Train a Wsabie model (d=100) for about one week, using pairs of similar documents.

Resulting Embedding Space

car	audi, ferrari, lamborghini, chassis, steering, roadster, renault, nissan, volkswagen
nobel	prize, peace, prizes, ig, physiology, laureates, dynamite
cheese	recipe, butterfat, gouda, cappuccino, creamy, toast, snacking
france	nantes, loire, hainaut, burgundian, hugues, alsace, rhone

Conclusion and Future Work

Conclusion

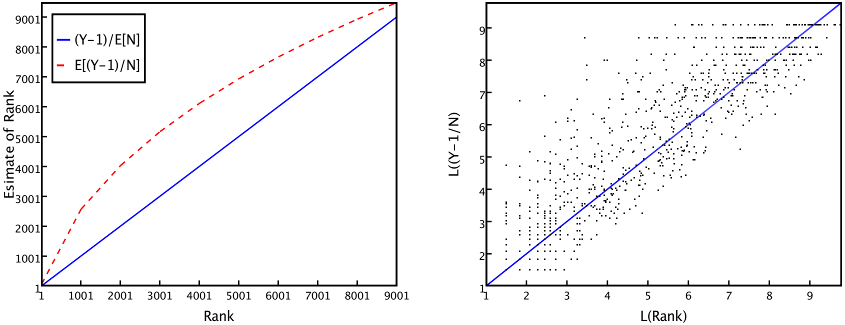
- Embedding model is scalable + performs quite well.
- WARP loss applicable to many large scale retrieval/ranking tasks.
- Annotating Images with more than 100k labels is feasible.
- Music Embedding model beats other approaches on similar tasks.
- Multi-Tasking the tasks of interest helps and also makes a compact model for all tasks of interest.
- Wsabie scales reasonably well with the number of features when sparse.

Future Work

- More multi-tasking:

  large set of genres, co-listen data, free text? . . .
- Use label embedding tree ideas for scalability.

WARP Loss: Approximation Accuracy



$$\frac{(Y-1)}{E[N]} = (Y-1)/\sum_{i=1}^{\infty} i(1-p)^{i-1}p,$$

$$E[\frac{(Y-1)}{N}] = \sum_{i=1}^{\infty} \frac{Y-1}{i}(1-p)^{i-1}p$$

$$p = Pr(violation) = \frac{rank}{Y-1}$$

Algorithm Time and Space Complexity

Time and space complexity needed to return the top ranked annotation on a single test set image, not including feature generation. Denote by Y the number of classes, n the number of train examples, d the image input dimension, $\bar{d}$ the average number of non-zero values per image, D the size of the embedding space, and p the depth of the tree for approximate k-NN.

Algorithm	Time Complexity	Space Complexity
k-NN	$\mathcal{O}(n \cdot \bar{d})$	$\mathcal{O}(n \cdot \bar{d})$
Approx. k-NN	$\mathcal{O}((p + n/2^p) \cdot \bar{d})$	$\mathcal{O}(n \cdot d)$
One-vs-Rest	$\mathcal{O}(Y \cdot \bar{d})$	$\mathcal{O}(Y \cdot d)$
Multiclass	$\mathcal{O}(Y \cdot \bar{d})$	$\mathcal{O}(Y \cdot d)$
Wsabie	$\mathcal{O}((Y + \bar{d}) \cdot D)$	$\mathcal{O}((Y + d) \cdot D)$

Bag of Visterms Representation

input	image
block segmentation	set of overlapping blocks
block descriptors	each block is described with color and edge (LBP) histograms
block quantization	each block is mapped to a discrete index, through kmeans learned over the training blocks.
bag of visterms	set of block indexes = set of visual words
output	*tf idf* weighted vector

Visterm Feature Representation

We use the sparse vector representation of [Grangier & Bengio '08]:

- Each image segmented into overlapping blocks at various scales.
- Each block represented by color+edge features.
- Discretized by training kmeans (10,000 "visterms").

Each image represented as a *bag of visual words*: a histogram of the number of times each visual word was present in the image.

10k dim sparse vectors an average of $\bar{d} = 245$ non-zero values.

It takes on average 0.4 seconds to extract these features per image.