INDIANA UNIVERSITY

TAG SELECTION AND PROPAGATION FOR LARGE-SCALE VISUAL LANDMARK RECOGNITION

by

Manu Singh

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the
SCHOOL OF INFORMATICS AND COMPUTING

May 2016

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Master of Science

Master's Thesis Committee

David J. Crandall,

Associate Professor.

Michael S. Ryoo,

Assistant Professor.

Sriraam Natarajan,

Associate Professor.

INDIANA UNIVERSITY

Abstract

SCHOOL OF INFORMATICS AND COMPUTING

Master of Science

by Manu Singh

The exponentially growing quantity of online social sharing content has created vast amount of readily available annotated multimedia resources like images. These resources provide exciting opportunities to build computer models that can perform complex tasks and potentially match human cognitive skills. However, the social content poses great challenges with respect to the inconsistency and inaccuracy within these annotations. A solution to this problem is to build data driven approaches that can assist to identify and resolve this inconsistent behaviour.

In this work, we investigate the large scale landmark classification task and propose enhancements that can help to improve the selection of annotated text (tags) in images and assist the visual features to build powerful machine recognition models. We first analyse the noisy content of these image tags, and propose a filtering criteria that helps to remove inaccurate data. Further, we use similarity and semantic matching techniques to make corrections within the textual content. To deal with the inconsistent nature of

this class of data, where annotations seem to be missing or insufficient, we apply tag propagation to find closely related content by searching for visually similar images in an auxiliary dataset of annotated images. To make this search process tractable for a large scale dataset, we use locality sensitive hashing. Our methodology to find the optimal parameters with these heuristics is data driven and can be applied to other related datasets and tasks. In our experiments, we observe and report both improved results on different classification categories and a faster training system.

To my parents, Arun and Shail.

Acknowledgements

I am extremely fortunate and indebted to Dr. David Crandall for introducing me to Computer Vision. This work would not have been enjoyable and complete without his motivation, companionable outlook, and kindness. I thank him for giving me the time and space to bring my ideas together and provide invaluable suggestions at every step of the work. I am grateful to Dr. Apu Kapadia, for his encouragement to pursue a research problem early in my academic career. I would like to thank Dr. Michael Ryoo, for introducing me to different areas of study in Computer Vision. I thank Indiana University, especially the School of Informatics and Computing, for giving me this wonderful opportunity to enrich my learning experience.

I would like to thank Dr. Michael Ryoo and Dr. Sriraam Natarajan for being a part of my thesis committee and for the valuable feedback.

I thank my parents, Arun and Shail, for their unconditional love and support. I owe them everything that I have achieved. Finally, I thank Monika and Rakhi for their enduring friendship and care in all aspects of my life.

Contents

List of Figures

List of Tables

Abbreviations

EXIF	**E**xchangeable **I**mage **F**ile **F**ormat
ANN	**A**pproximate **N**earest **N**eighbor
SVM	**S**upport **V**ector **M**achine
rLDA	**R**egularized **L**atent **D**irichlet **A**llocation
AFSVM	**A**ugmented **F**eatures **S**upport **V**ector **M**achine
ReLU	**R**ectified **l**inear **u**nit
CNN	**C**onvolutional **N**eural **N**etwork
LCS	**L**ongest **C**ommon **S**ubsequence
k-NN	**k** **N**earest **N**eighbor
LSH	**L**ocality **S**ensitive **H**ashing
FT	**F**iltering
WM	**W**ord **M**atching
TP	**T**ag **P**ropagation

Chapter 1

Introduction

Over the years, clicking and sharing digital photographs has increased at an unprecedented scale. Cheaper, faster, more reliable and readily available resources like digital cameras, internet, and mobile phones have made it possible to collect images on such a scale. Mary Meeker [3] reported that each day in 2014, an average of 1.8 billion photographed images were uploaded to social sharing platforms like Facebook, Flickr, Picasa, Instagram, etc. [4–7]. These social platforms also store metadata information associated with the image, like in Exif [8] format as shown in Figure 1.1 and metadata of the user, like their social connections. Humans are good at extracting and understanding the concepts associated with an image, such as scene, context, geolocation (visual features) and textual themes (tags) naturally, but for a machine this is complicated. Humans can also learn distinct visual and textual resemblance from a collection of images which may not be implicit with a single image. Machine can learn concepts from local and global visual features as well as user annotations such as tags, but their performance is restricted due to limited data. Before the revolution of social data, these tasks were practically infeasible. Nowadays, readily available public large-scale data provides

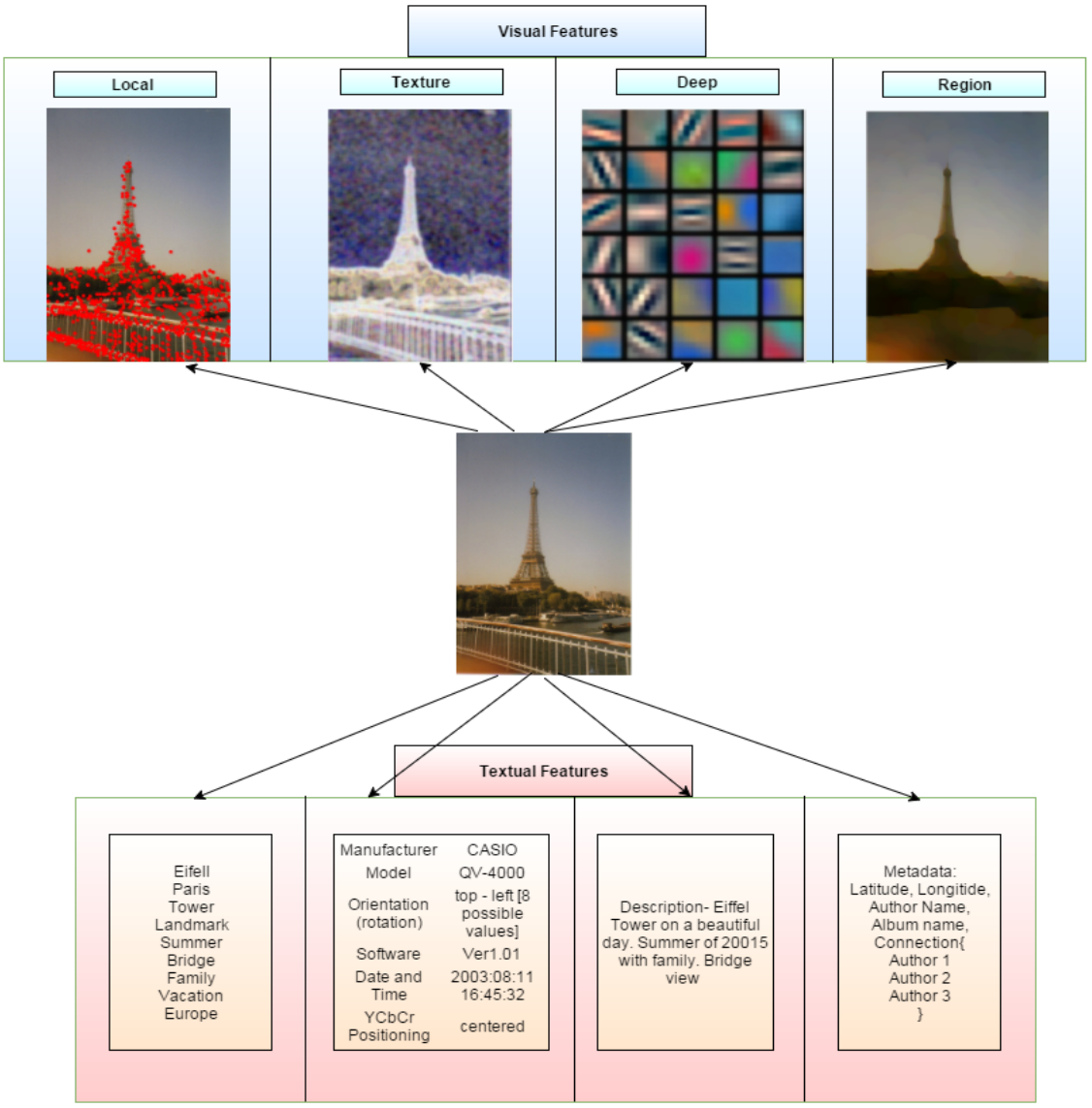


FIGURE 1.1: Visual and textual features in an image.

exciting opportunities to generalize and build better models that are able to work with a larger number of classes and thus have significant practical application value.

1.1 Landmark Recognition Overview

Landmark recognition is the task of locating or identifying landmarks across the world from a given query image. The definition of what a landmark is can vary based on the perception of different people, and communities. In this work we consider landmarks

as discovered by [9], which considers the density of images taken within a radius of a geolocation. Work by [10] also uses the similar idea to discover landmark but additionally uses landmark information from travel guides such as Wikitravel [11]. The geotag information with the images have also enabled us to label [9, 10] these large datasets in an unsupervised approach which otherwise would have taken a very large number of human hours or would have been impractical altogether.

Landmark Landmark recognition in large scale image collections can help to summarize, index, and visualize images across the world. This can directly help in creating virtual tourism environments [10], 3D reconstruction models [12], and geographical and climatic exploration. Learning the location of an image can also provide a useful prior for other recognition tasks.

1.2 Motivation: Tag selection and propagation

The visual data collected from these social platforms have lots of variation with respect to the landmarks. These variations can occur due to viewpoint, scale, illumination and background changes. Further, there are occlusions and deformations in the images which can obfuscate the distinct visual features of the landmark. Landmarks with less intra-class variations can add more confusion in the models, for example *London Bridge*, *Golden Gate Bridge*, and *Brooklyn Bridge* would have large overlap in visual features. Another challenge with such a dataset is that it contain images of landmarks that are taken indoors and do not contain significant visual information to make any inference. In such cases associated tags can possess much more discriminative power than visual features. Li et al's [13] work on large-scale image classification shows empirically that

non-visual data (textual tags) provide strong cues and even outperform visual features for all categories of classification tasks. But tags also have some shortcomings:

1. Unfortunately, the tags do not have any predetermined distribution and the range of the tags often varies from one landmark to another. This can affect the efficiency of the model especially when the range of tags is high and the classification task involves hundreds of landmarks.

2. Kennedy et al. [14] reported that the datasets collected from the web (Flickr) have a 50% chance of incorrect annotations.

3. Crandall et al's [9, 13] large-scale landmark dataset shows about 15% data with no annotations.

1.3 Terminology

In this section we look at some of the key terms and represenations that are used in this work.

1.3.1 Visual data

The local features associated with the visual content of an image describes the salient features of an image such as SIFT [15]. These features are quantised and represented by a feature vector:

$$F_V = \left\{ F_{V1}, F_{V2}, ..., F_{VD} \right\}, \tag{1.1}$$

where each F_{Vd} represents a feature in a D dimensional space. Other examples include texture [16], color based features and recently developed deep features [17].

1.3.2 Textual data

The textual data includes human and machine annotated text with the images such as tags, description, Exif [8] content and user metadata such as social connections etc. These features are represented by a feature vector:

$$F_T = \Big\{ F_{T1}, F_{T2}, ..., F_{TK} \Big\}, \tag{1.2}$$

where each F_{Vk} represents a feature in a K dimensional space. In this work we look only at user annotated tags from Flickr [4] and consider them as textual features.

1.3.3 User tags (textual feature)

In this work the textual features are derived from the tags associated in an image. The feature representation is equivalent to equation 1.2. For a set of K available tags in a given task we associate a value 1 to F_{Tk} in the feature vector if the k^{th} tag exist in an image, otherwise a value 0. Our feature vector is binary and represented as:

$$F_T = \Big\{ F_{T1}, F_{T2}, ..., F_{TK} \Big\} (\in 0, 1)^K \tag{1.3}$$

1.3.4 Classification category

To quantify our results we use five out of six classification categories which were introduced by Li et al's work [13] on landmark classification. In that work, they partition the data downloaded from Flickr [4] into different geographical clusters, and select those clusters which have the highest peak of images. This selection is formalised as top 10, top 20, top 50, top 100, top 200 and top 500 category classification task which contain the top number of peaks, e.g. top 10 category has 10 most photographed places in the dataset and so on. In this work we show our results on top 10 - top 200 classification categories.

1.4 Objective

In this work we investigate landmark classification on a large scale dataset collected by Crandall et al. [9, 13] and particularly focus on the shortcomings of the tags associated with the images which can help in the overall classification task. Here we evaluate our results on 5 category tasks - top 10, top 20, top 50, top 100 and top 200 landmarks, which collectively contain about 730,000 images.

1.5 Proposed framework

Our main contributions are the enhancements that we propose:

1. We employ a filtering process (section 4.2) to remove noisy tags, which significantly reduces the dimensionality and sparsity of the textual features and increases the efficiency of the model.

2. We use similarity and semantic matching (section 4.3) to identify morphological, structural, spelling and contextual variations within the tags which further reduces the sparsity of the features vectors and enhances accuracy.

3. For the missing annotations, we use tag propagation [18] to generate tags (section 5.2). We also extend tag propagation [18] to existing annotated images based on our selection criteria (section 5.3.1). This improves the accuracy further and gives us the best results (section 8.4).

In this work we emphasize a large scale data driven approach which differentiates it from many approaches which consider relatively small datasets and classification tasks [19–21]. This raises an important question about how much data is actually required to generalize these tasks, as there can be an infinite number of social images possible. We do not evaluate or answer this in this work, but the reason that since people usually take a picture from similar viewpoint and capture relatively similar objects at these landmarks the data aggregation approaches like [9, 10] should reasonably capture the useful visual features from a practical standpoint. We select [9] and evaluate our results on this dataset and do not select [10] as the ground truth testing dataset is very small.

1.6 Organisation

We organise our work in chapters starting with the related work in **Chapter 2**. In this chapter we describe some background work in landmark recognition and popular techniques in image retrieval system. **Chapter 3** describes some of the key concepts in deep learning and then explicates the deep features and finetuning procedure that are used in this work. **Chapter 4** and **Chapter 5** illustrates the key idea of this work and proposes a tag selection and propagation criteria respectively. We also explain the

experimental setup and the parameter selection process in these chapters. **Chapter 6** explains the workflow and key components of our system. We show the visual and textual examples in the dataset in **Chapter 7** and show some of the key problems with such large dataset. Finally, we show the results, unsuccessful experiment and future work in **Chapter 8**.

Chapter 2

Related Work

2.1 Introduction

Here, we survey some of the background work related to the thesis. We start with the introduction of image classification background and then particularly look at the large scale landmark classification works. We also investigate some image retrieval based techniques that deal with combinations of visual and textual features. Since our work is focused on learning and improving the textual features associated with large scale image classification tasks, we use them as a metric for comparison with this related work.

2.2 Image Classification

Image classification using the bag of words [19–21] model is a simple approach which typically uses handcrafted visual features. It was very popular until Krizhevsky et al's [22] work on Convolutional Neural Networks reported significant improvement on these tasks using features learned from data. Since the bag of words model loses spatial

information between the features, variants of the models were introduced which focused on weighing the regions using spatial pyramids [23] and using receptive fields [24] to determine important regions in an image. One of the key challenges with the bag of words approach is learning the optimum dictionary size. To make the process tractable for large datasets, approximate nearest neighbors (ANN) [25] approaches are used.

2.3 Landmark Classification

Zheng et al's [10] work evaluates landmark recognition for a very large collection of landmarks - 5312 landmarks from about 20 million images - but their testing dataset is rather very small - 728 handpicked images from 124 landmarks - which may not statistically represent the generalization of their model for large scale images. They use textual features only to learn the landmark name and do not use them as features in classification. Further, they use visual features to filter the data which brings an implicit bias when these features are again used in classification. They use image retrieval method to find the closest match to a query image from a database of images by indexing using k-d tree [26]. This method may not be tractable when the querying images increase.

Im2GPS [27] also uses image retrieval methods to find a closest geographical match for a given query image. They apply nearest neighbor matching in a database of about 6 million images taken from Flickr. Though this work is related, the goal there is locating images with geographical precision, whereas we are trying to locate images in specific geographical location cells. They use a combination of six global visual descriptors to index images but do not consider textual features in their work. Similar to [10], the test set is very small (237 images only) and not quite scalable when the database size grows as the feature vectors have dimensionality of 100,000.

Li et al's [13] experiments on large-scale landmarks using the bag of words model show the impact of different dictionary sizes for different classification categories using multiclass SVM [28]. They also create binary features using tags and train a separate multiclass SVM which outperforms the visual features using the bag of words model. Finally, they show that a combination of both the visual and textual features works the best. They report the results for 500 landmarks with around 2 million ground truth images. Ground truth images collected by [9, 13] are purely based on the geo locations which prevent any implicit bias in the model as geolocation do not represent the feature. Training and testing sets are equally split in their experiments which are fairly large sized and the results give good generalization for these large scale tasks. We, therefore, use this work as our baseline and evaluate our results on the same dataset. However with regards to textual features, they perform minimal filtering discarding the tags that occur less than 5% in a geographical cluster. They do not account for morphological, language, context and spelling variations that occur in tags and also do not infer any additional information for images that have no tags, as a result, unannotated images have empty feature vector.

2.4 Image Retrieval Systems

Torralba et al's [29] work shows that there exists visual consistency between images with contextual coherent textual content, and they conclude this by performing experiments on a large dataset of about 79 million images. However, these images were collected by searching images from the web for the common 75,062 non-abstract nouns in English. So it is not clear whether these images would truly correspond the social images tagged by users such as in Flickr [4], as the images collected in their work would depend on the

criteria of the indexing in the search engine. Further, the textual information used here are common english words which may or may not resonate with the tagging behaviour of social users, as they can have loosely labeled content. In our work, we do not make any assumptions about the textual content and consider the socially annotated data from Flickr [4] which contains different types of noise and common English words.

Work by [30] looks at the problem of inconsistency of tags and proposes a graphical retagging model. In this model, they first learn the tag specific visual vocabulary for each tag and then construct the similarity graph amongst these vocabularies. In their approach retagging is a collaborative process where the visual features of a query image are used to find the closest tags from the visual vocabulary and simultaneously the user tags of the query image are used to find the contextually related tags by constructing the similarity graph between these vocabularies. So they simultaneously look to add relevant tags based on the visual similarity as found important by [29] and also consider the co-occurrence relation of tags by finding relevant tags. One of the challenges in this approach is to learn the visual vocabulary for each tag especially in problems similar to large-scale landmark classification where the number of unique tags even after filtering is around tens of thousands. Compared to this work our approach is both simple and scalable especially for large classification tasks, as we use the visual similarity for tag propagation and find contextual similar tags independently without a constraint relationship model.

Several other approaches have been designed as well to improve image search on social images. Though the end goal of this work is different than ours, they are related, as they give insights into the relationship between visual and textual content. These approaches can be broadly divided into three categories:

2.4.1 Separated technique

Separated methods consider either visual or textual content to refine tags. In [31], the authors propose a new graphical regularized Latent Dirichlet Allocation (rLDA) model to learn the textual similarity iteratively based on tag similarity and relevance. They process tags associated with each image as the contents of a document and then regularize with topics learned from tags of separately collected similar visual images. Liu et al. [32] proposes a tag ranking procedure where existing tags are re-ranked based on their relevance with the images content. At first, they estimate the relevance score of tags using probability density estimation and then refine those scores by performing a random walk on the tag graph. The tag graph that they create takes into account the tag occurrence score based on the visual features of the image. Li et al. [33] employ a tag relevance score scheme where, given an image and a tag, they first find visually similar images, accumulate the tags associated with them, and use voting to decide the score for the tags. These methods and similar techniques like [34, 35] indicate that visual features and textual features in noisy social images can resonate strongly, and using one of them can help to refine the inconsistencies of the other.

2.4.2 Sequential technique

Image search based on sequential techniques employs visual content and the tags sequentially. Liu et al. [36] proposes a ranking method where they first determine the scores based on the tags of the images and then refine the scores again on visual content of images. They propose an optimization problem based on relevance scores on visual and textual similarity. Visual similarity is formulated with a Gaussian kernel function and the textual similarity is based on the overlap of tags. The overlap is a measure of

the contextual similarity amongst tags. Chen et al. [**?**] estimate the initial score for tags by training a Support Vector Machine (SVM) classifier for each tag with loosely labeled positive and negative samples collected using the inverted file method. In that work they observe some noisy tags on the loosely positive labeled sample, which they filter out by training a separate SVM with Augmented Features (AFSVM). They further refine the scores with a graph-based method that simultaneously considers the similarity of photos and semantic correlation amongst the tags.

2.4.3 Joint technique

In [37] the authors proposes a joint model to learn the importance of user tagged images based on visual and textual cues with a graphical approach. The relationships are represented in a hypergraph, where each hyperedge represents two or more vertices. The graph is represented as G= {V, E, w}, where V and E are the vertex set and edge set of the graph respectively and w represents the weight of the edges. In this model, every social image is a vertex in a hypergraph constructed by the combination of a bag of visual words and bag of textual words model. The idea is to associate similar visual and textual features in the same hyperedge. Semi-supervised approaches are applied to build the hyperedges based on the pairwise score between the visual and textual features. In [37] fixed edge weights are used, whereas in [38] the authors learn the weights based on an optimization problem. In [38] the authors reports that their approach performs better than separated and sequential approaches in many category of tags, but is comparatively slower.

Chapter 3

Deep Learning

3.1 Introduction

Recently, deep learning based techniques have outperformed many classical computer vision approaches, which involve developing algorithms to design hand crafted features. Deep learning models are essentially neural networks with greater numbers of hidden layers and that use non-linear transformations to learn the representation in data. In this work, we use deep learning based Convolutional Neural Networks to extract visual features (CNN features), which we use as a measure of visual similarity. In this chapter, we explain the building blocks of a neural network and then describe a Convolutional Neural Network architecture that we adopt in our work.

3.2 Perceptron

Deep learning is inspired by the working of the human brain which is capable of solving complex tasks very quickly. The basic building block of this network is a perceptron

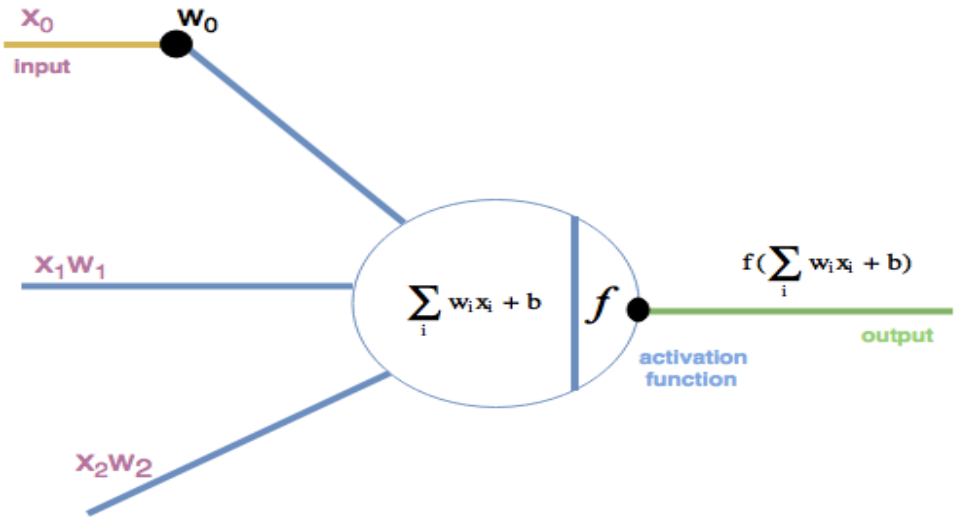


FIGURE 3.1: An artificial neuron (Perceptron).

(figure 3.1), a mathematical model similar to a neuron in the human brain.

Each perceptron consists of:

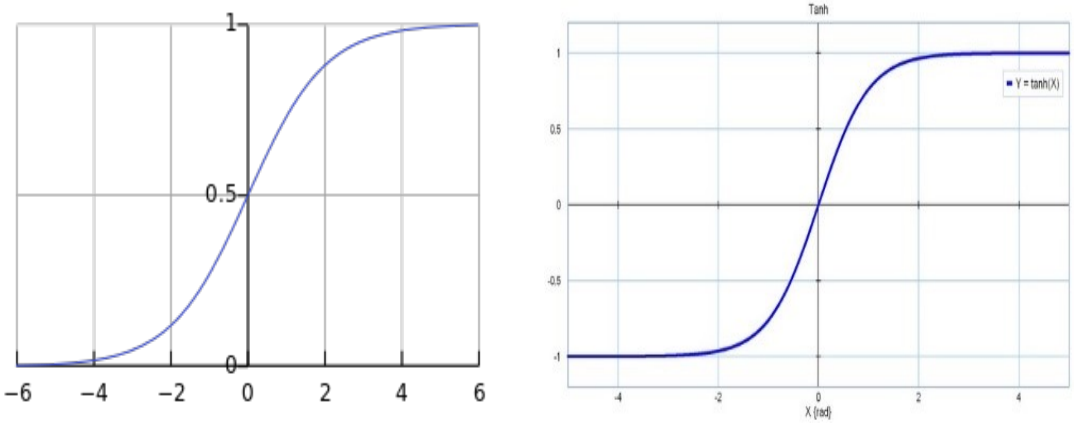
- Input or signals: each unit consists of various inputs from other similar units. All these input connections have weights which determine the magnitude or influence of the corresponding generated signal. These weights are learned by the system. The strength of a signal is a dot product of the weight and input added to the bias.

- Body or cell: The body sums up all the signals and passes them to the activation function.

- Activation Function: The activation function gets the aggregated signal from the body and is responsible for the firing or activation of the whole unit. Some popular choices of activation function are:

  - **Sigmoid**: The sigmoid function takes a real value and outputs a number between 0 and 1 (table 3.1 (a)). It is represented mathematically as $1/(1+$

e^{-x}) . The sigmoid mimics the nature of biological neuron by either firing (1) or not firing (0). One of the shortcomings of using sigmoid is that its long tail near 0 and 1 greatly reduces the local gradient and causes problems during backpropagation where a significantly small local gradient gradually kills the gradient flow to connecting neurons.



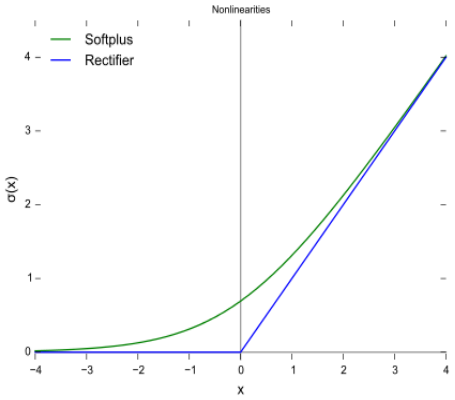
a) Sigmoid function b) Tanh function

TABLE 3.1: Activation functions.

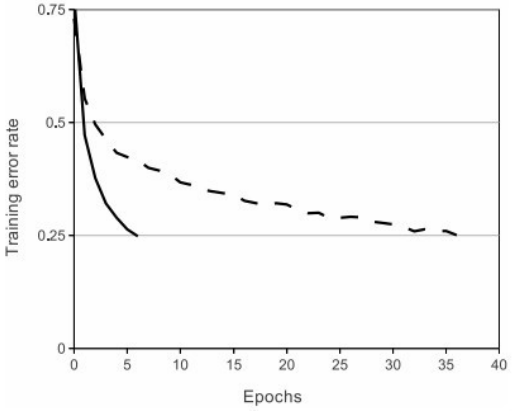
– **Tanh**: The Tanh function takes a real value and outputs a number between -1 and 1 (table 3.1 (b). It is represented mathematically as $(1-e^{-2x}/(1+e^{-2x})$. It also suffers from the gradient saturation problem like sigmoid function.

– **Rectified linear unit**: The rectified linear unit(ReLU) as described by Nair and Hinton [39] is a linear function with a threshold at zero as shown in table 3.2 (a). The mathematical equivalence is max(0, x). ReLU enhances the training speed greatly as reported by [22] and shown in table 3.2 where they reported 25% error rate 6 times faster than using a Tanh activation function. ReLU does not involve a complex exponential calculation like sigmoid and Tanh and thus is significantly faster.

ReLu suffers from a problem where a sudden large gradient very early in the training can inhibit any further activations. This is dangerous and can lead to



a) ReLU [40]



b) Figure [22] showing converge rate of Tanh (dotted) and ReLU (solid)

TABLE 3.2: Rectified linear unit (ReLU).

an entire portion of the network being dead. So tuning the learning rate here is very important. Generally low training rate helps alleviate this problem.

3.3 Network architecture

The Neural network is a large network of neurons spread across different layers. Since these networks are feed forward, these connections are acyclic in nature and occur across adjacent layers. There are no connections between the neurons in the same layer. A hidden layer, as shown in figure 3.2, is a fully connected layer as its neurons are connected to adjacent layers. There is an input layer with 3 neurons and an output layer with 2 neurons in figure 3.2.

3.4 Backpropagation

Backpropagation, as introduced by [41], is a learning algorithm that is used to tune the weights and biases of the neurons in a neural network. The algorithm computes the loss at the end of the network and then propagates the gradient change recursively back to

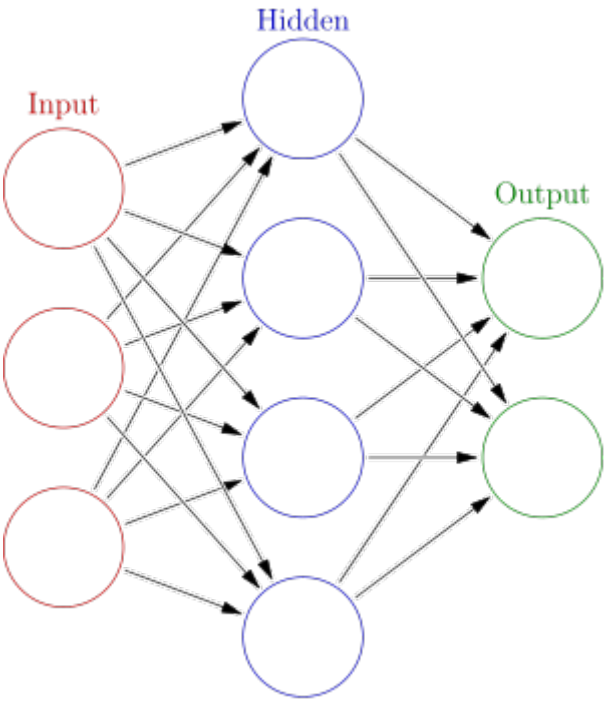


FIGURE 3.2: Neural network with 3 layer [1]

the network. The gradient distribution is dependent on the choice of activation function used and the magnitude change further depends on the local gradient at each of the neurons during the forward pass. This propagation step intuitively helps the hidden layers to adapt based on the interaction between the input and output layers, which thus learn to be sensitive to important features. This reduces the overall loss and helps in the generalization of the model.

3.5 Convolutional Neural Network (CNN)

The convolutional neural network [43] is a special case of the neural network where the neurons are 3D or 2D kernel filters. The kernels or filters have weights which are shared across the whole layer. This greatly reduces the parameters in the overall network, making it more efficient to work with images which otherwise would force each neuron

Algorithm 1 Backpropagation learning algorithm [42]

```
INITIALIZE
    Assign network weights (often small random values)
for data in training set do
    FORWARDS PASS
        Starting from the input layer, do a forward pass through the network, computing
        the activities of the neurons at each layer.
    BACKWARDS PASS
        Compute the derivatives of the error function with respect to the output layer
        activities
        for layer in layers do
            Compute the derivatives of the error function with respect to the inputs of the
            upper layer neurons
            Compute the derivatives of the error function with respect to the weights be-
            tween the outer layer and the layer below
            Compute the derivatives of the error function with respect to the activities of
            the layer below
        end for
        Updates the weights.
end for
```

to learn a large number of weights (height * width * color channels of an image). This reduction of weights also helps in generalization of the model and reduces overfitting. Each filter performs convolutional operation across the input with the same weights and the layer doing these operations is called the convolutional layer. We describe some components of the convolutional neural network architecture as described in [22].

- Input Layer: This layer ingests the raw pixel value from the image represented as a matrix (height * width * color channel of the image).
- Convolutional layer: This layer consists of the kernel (neurons) which do a convolution operation over local regions from the input layer, and outputs feature maps which are passed as input down the network.
- Activation: The ReLU will apply the thresholding to provide activations for the response maps. After each layer, the ReLU non-linearity is applied.

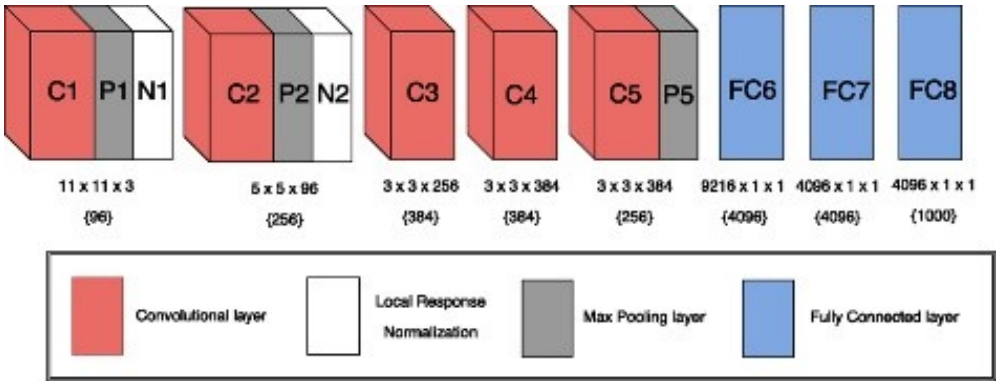


FIGURE 3.3: An example of 8 layer Convolution neural network [2] architecture.

- Pooling layer: This layer subsamples the feature maps, looking at small regions and representing them as typically a maximum value or an average.

- Fully connected layer: In this layer, every neuron is connected to all the neurons of the adjacent layers. This layer calculates the class scores in the end.

3.6 Dropout

Hinton et al. [44] proposed another regularization concept to reduce overfitting, in which they randomly hibernate a subset of neurons during training and show that the technique is equivalent to training multiple nets. During testing time, the whole network is used and activations are scaled with the dropout rate.

3.7 CNN features

The major difference between CNN features and classical hand designed features such as SIFT [15] and HOG [45] is that the hand designed features are not learned by the system, whereas the CNN features are learned by the networks at various stages or layers. Each layer in this hierarchical network extracts some features and forwards them down as an

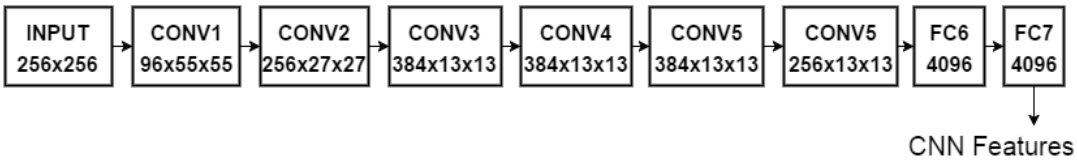


FIGURE 3.4: CNN feauture extraction.

input to the next layer. During the training phase, all the layers are learned jointly through back-propagation. We use the Caffe [46] framework to define a Convolutional Neural Network which is essentially AlexNet [22] architecture. The only difference in this implementation is that the biases are initialised to 0.1 instead of 1 and the training do not include relighting data-augmentation [47]. Here we take a pre-trained AlexNet [22] model on ILSVRC12 [48] dataset and fine-tune it on our training dataset. ILSVRC12 [48] has 1000 object categories and about 1.2 million images in the training set.

3.8 Fine-tuning

The idea behind fine-tuning is to reduce the high computation cost of training from scratch. So by picking a pre-trained model which has already learned important feature distinctions on a large dataset, the model is responsive to generic low-level features such as edges. Zeiler & Fergus's work [17] shows that once the model is trained, the low-level features are extracted by filters at the initial layer of the network, which are then aggregated through pooling and propagated down the network where it learns the high-level features. Thus, we reuse the weights already learned for all the layers except the last layer (fully connected layer) where weights are randomly initialized during fine-tuning so that the network is trained for the correct number of classes. The learning rate of the layers varies during the fine-tuning process, where we keep higher learning rate for the last layer which is initialized with random weights, to low learning rates for

other layers. This is again to minimize a large change in the features already learned by the lower layers and minimize over-fitting. The CNN feature we refer to is the 4096 dimensional vector from the 7^{th} layer of the AlexNet [22] architecture.

Chapter 4

Tag Selection

4.1 Introduction

Kennedy et al. [14] reported that the datasets collected from the web (Flickr) have a 50% chance of incorrect annotations. Moreover, tags also suffer from mispelling, contextual and language variations. In this chapter we propose a strategy to filter out the noisy tags. We also apply similarity and semantic matching to identify and correct the textual variations. The filtering process helps to reduce the dimensionality of the textual features, and correcting the mispelling, contextual and language helps to reduce the sparcity.

4.2 Tags Filtering

Li et al's [13] work on landmark classification uses a binary feature vectors on the textual data and shows that they are powerful and even outperform the bag of visual words [49] model. The dimensionality of the feature vectors varies with respect to the classification

task, and increases rapidly as the number of classes increases. As the dataset [9] contains images within a small radius (100m) of the landmark, there are images which have noisy annotations that do not correspond to actual landmark and rather have images of people eating food at a cafe near the landmark or riding a subway. We select some of these images and tags and show in section 7.5. Further, there are images that show landmark clearly but have been incorrectly annotated. For example, we found people have used tags such as *spain*, *espanha*, *bike* and *mexico* for a particular landmark in Paris, France (Eifel tower). This observation within the dataset is the noise that occurs specific to a user or an event but not related to the landmark. So we propose to perform a filtering step that can remove some of these tags and reduce the dimensionality of the binary feature vector.

The filtering step is based on a simple idea that accounts for the occurrence of the tag relative to the total range of tagged images per landmark. Instead of looking at each landmark individually and filtering out the tags, we look at all the landmarks collectively for a given tag to make a decision. This keeps the final set of tags consistent across all landmarks. We calculate the score $T_g^{(i,k)}$ of a tag with respect to all the landmarks. The score is essentially a probability score of the occurrence of the tag for a landmark.

$T_g^{(i,k)}$ is the score for i^{th} tag in the k^{th} landmark given by equation 4.1,

$$T_g^{(i,k)} = \frac{\sum_{j=1}^{n} P_{(i,j)}}{N}, \tag{4.1}$$

$$P_{(i,j)} = \left\{ \mathbf{1} \text{ if } i^{th} \text{ tag exists in } j^{th} \text{ image, } \mathbf{0} \text{ otherwise} \right\}, \tag{4.2}$$

and N is the total number of tagged images in k^{th} landmark.

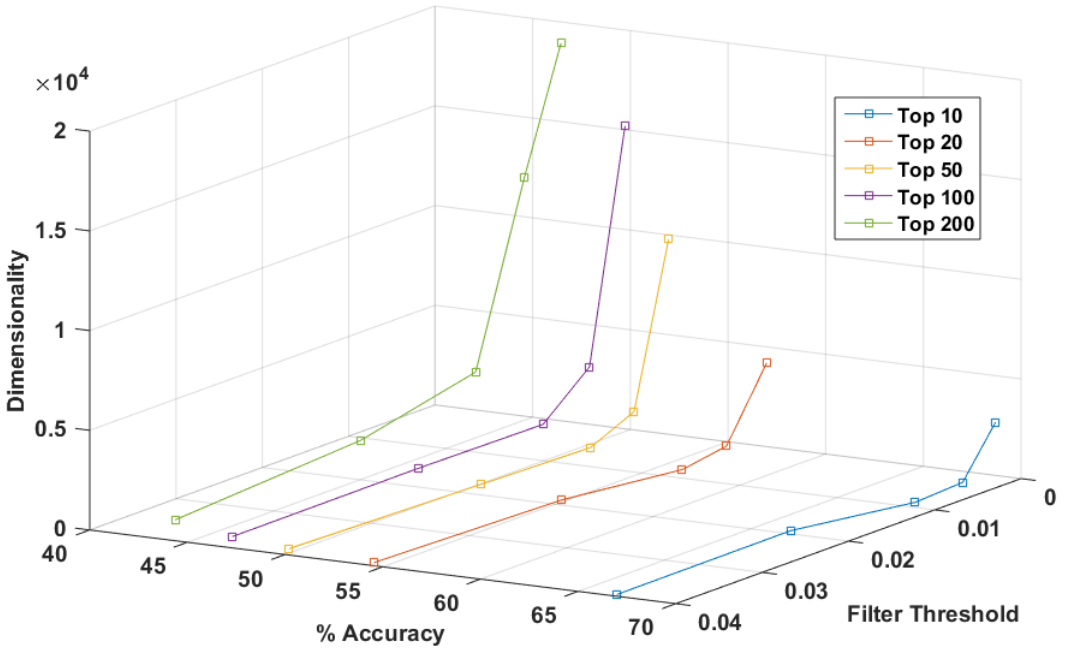


FIGURE 4.1: Dimensionality and accuracy variation with filtering.

For each tag, we create a vector of the scores which is the size of all the landmarks in the given classification task and select the maximum score. If the maximum score $T_{max}^{(i)}$ given by equation 4.3 is less than the threshold then we reject and filter the tag.

$$T_{max}^{(i)} = Max([T_g^{(i,1)}, T_g^{(i,2)}, T_g^{(i,3)}, T_g^{(i,4)}, ..., T_g^{(i,k)}]) \tag{4.3}$$

4.2.1 Impact of filtering

We run experiments and compare $T_{max}^{(i)}$ with various thresholds and observe following in figure 4.1:

1. The dimensionality reduces significantly as we move towards a stronger threshold, rejecting the tags aggressively, but the accuracy is also reduced since the information loss is greater.

2. There is a tipping point in the threshold value where the dimensionality increases significantly but with minimal accuracy gain. We take that value as our threshold value.

Though this filtering process reduces the dimensionality significantly and gives a speedup, we observe that the accuracy actually drops slightly for all categories by about 1%.

4.2.2 Filtered tags analysis

4.2.2.1 Useful tags

To understand the accuracy drop, we sample some random tags that were filtered out and try to classify them into different categories (table 4.1).

Filtered tags (Useful)		
Category	**Filtered tag**	**Closest match**
Morphological/structural variation	thetatemodern	tatemodern
	pyramids	pyramid
	monuments	monument
	thenationalgallery	nationalgallery
	theweatherproject	weatherproject
	churches	church
	notredameofparis	notredamedeparis
	blacknwhite	blackandwhite
Spelling variations	coloseo	colosseo
	notredam	notredame
	monumenti	monument
	catherdral	cathedral
	colleseum	colloseum
	piazzettasanmarco	piazzasanmarco
	greatbritan	greatbritain
	cattedrale	cathedrale
	skyscrapper	skyscraper
	engeland	england

Prefix/Suffix variations	europe99	europe
	chinesenewyear2008	chinesenewyear
	pars07	paris
	europe07	europe
Language variations	arte *(Spanish)*	art
	Italia *(Italian)*	Italy
	architektur *(German)*	architecture
	francja *(Polish)*	france
	cathedrale *(German)*	cathedral
	stadion *(Indonesian)*	stadium
	urbano *(Spanish)*	urban
Context variations	promenade	parade
	sight	view
	castle	palace
	pillar	tower
	horizon	view
	lighting	light
	sunshine	sun

TABLE 4.1: Reusable filtered tags.

To find language variations we use Google Translate [50] to auto detect language (table 4.1). To this we try to aggregate and reuse these filtered tags which could help in reducing the sparsity (see more in section 4.3).

4.2.2.2 Noisy tags

Though we find a substantial number of tags that seem useful, there exist a subset of this filtered set which is random, incorrect or personalized based on user (table 4.2), so we believe that filtering step is still important. For the incorrect tags category we show the ground truth landmark and its corresponding country. One or more tags from these categories is observed in all the classes.

Filtered Tags (noise)		
Category	**Tags**	
Personized	andrea,mom, alex, bryan, james, gary	
Random	stilllife,1870mmf3545g, man, new, drunk, nophotoshop, 1870mmf3545g, topv333, topv777, ltytr1,2006 , 2007, 2008	
Incorrect	**GT Landmark**	**Tags**
	eiffel tower, france	spain, espanha, newyork, mexico,uk
	bigben, england	germany, Deutschland, Portugal, nyc, America
	londoneye , england	spain, espanha, London, london, Ireland
	colosseum, italy	russia, mexico, Poland, espaa, Portugal

TABLE 4.2: Noisy Tags.

4.3 Matching in tags

The tags have been added by users from all around the world, which brings diverse perspective and context in these annotations. So a given image can be annotated by different users with a different set of tags, especially since users also use different languages based on their preference as shown in table 4.1. Some tags are also misspelled and sometimes even labeled incorrectly, as we observe them in table 4.2. We propose to search and include the useful tags based one of the two criteria:

4.3.1 Similarity matching

We measure the similarity of two strings using the pattern matching algorithm introduced by [51]. The algorithm recursively finds the longest common subsequence (LCS) [52] between the strings. For any string if the LCS is less than the length of the string then the sub-strings to the left and right of the LCS are again processed to find LCS amongst them. This process is repeated recursively on these substrings and finally, the matching score is returned as the ratio of the total matching characters divided by the

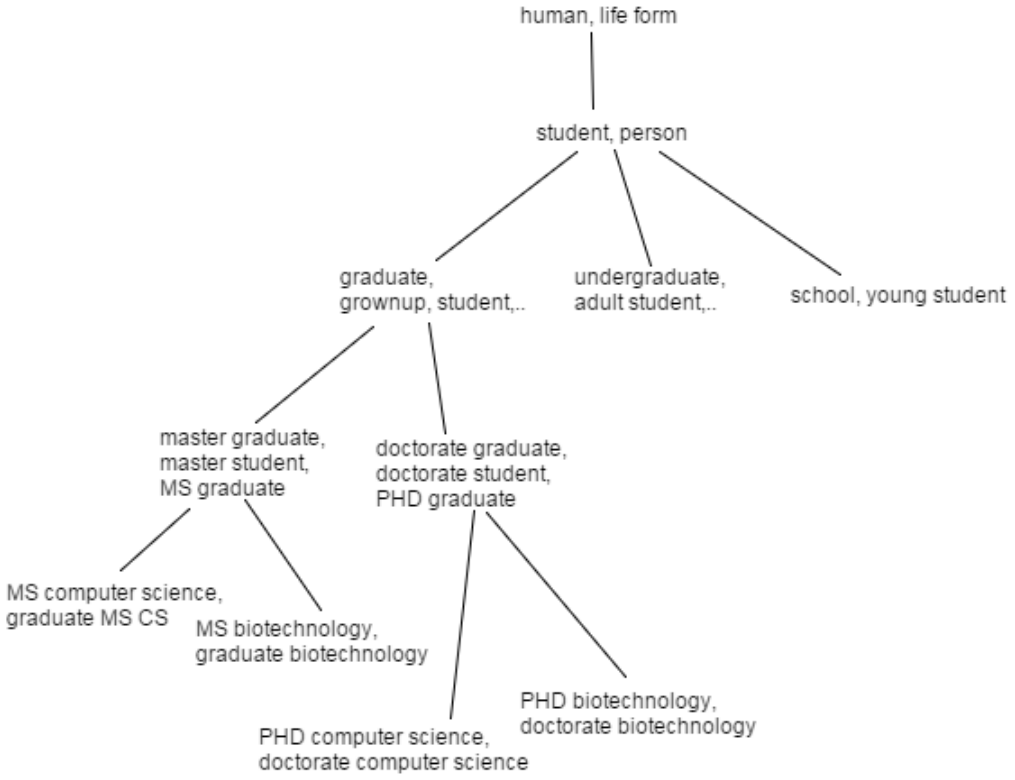


FIGURE 4.2: Hierarchical structure in a knowledge base (WordNet).

total length of the strings. The score is always between 0 and 1. We consider good matches to be above the score of 0.85.

4.3.2 Semantic matching

Here we use an already available knowledge base such as WordNet [53] which gives a hierarchical structure modeling of the English language. Each node or word in this knowledge base consists of a set of similar synonym words called a synset. We show a hypothetical example of a similar hierarchical structure in figure 4.2 to illustrate the concept.

Semantic similarity between words is a well-studied area in the field of text mining and information retrieval and is specially used to learn the similarity between sentences and documents. Li et al [54] proposes a method to learn the semantic similarity between

words by considering the path lengths of the common ancestor between two words in the hierarchical knowledge base and also the depth of the common ancestor. They formulate a similarity measure between two words as $(w_1, w_2) = f_L(l) \cdot f_D(h)$.

$f_L(l)$ is the transfer function for the path length and is monotonically decreasing with respect to the length. This can be illustrated by an example, suppose two words exist in the same synset, then the length of the common ancestor path is zero, otherwise it will be some positive number based on the path length of the common ancestor. For example in figure 4.2 the path length between *MS biotechnology* and *graduate biotechnology* is 0 as they lie in the same synset whereas the path length between MS computer science and PHD computer science is 4 (*MS computer science—master graduate—graduate—doctorate graduate—PHD computer science*). So the transfer function for the path is given by $f_L(l) = e^{\alpha l}$, where α is a constant dependent on the knowledge base and l is the path length.

$f_D(h)$ is the transfer function of the depth and is formulated based on the idea that generally the words at a higher layer of the hierarchy are more generic than the ones at a lower layer. So if a common ancestor match occurs at a low level then it is more specific context, match and thus it should weigh stronger than the match at an upper layer. For example in figure 4.2 the depth of match between *student* and *master student* is 1 as the common ancestor is *student* whereas the depth of match between *master student* and *doctorate student* is 2 as the ancestor is a *graduate*. Thus, the transfer function is a monotonically increasing function and formulated as

$$f_D(h) = \frac{(e^{\beta h} - e^{-\beta h})}{(e^{\beta h} + e^{-\beta h})}, \tag{4.4}$$

where β is a constant dependent on the knowledge base and h is the depth of the common ancestor. So our context similarity score between two tags is $(w_1, w_2) = f_L(l) \cdot f_D(h) = e^{\alpha l} \cdot \frac{(e^{\beta h} - e^{-\beta h})}{(e^{\beta h} + e^{-\beta h})}$ and it is always between 0 and 1. We use WordNet [53] as the knowledge base and use the values $\alpha = 0.2$ and $\beta = 0.45$ as found by [55]. We consider good matches to be above the score of 0.85.

We look for all pairs between filtered and unfiltered tags and apply similarity and semantic matching schemes. In the end, we collect those tags from the filtered set which score above 0.85 in either of the schemes as useful tags. As a result of this one-time processing step, we can learn the dictionary of all the filtered tags from the training data that are not noisy but useful and associate them with the corresponding tag from the unfiltered set. Now whenever we encounter any filtered tag during training and testing data, we can probe this dictionary to find the relevant tag instead, and thus reduce sparsity of the feature vector.

Chapter 5

Tag Propagation

5.1 Introduction

Li et al. [13] reported that about 15% of the images on Flickr have no tags. This leaves a scope of improvement, and the tag propagation concept applies naturally and helps to estimate tags by comparing visually similar tagged images from an auxiliary dataset. In this chapter, we apply tag propagation to circumvent the 'no tag' image problem, and we also extend it to other images based on our selection criteria. Comparing visually similar images in large scale datasets can be computationally expensive, so we apply locality sensitive hashing to make it efficient. We also conduct experiments and use a parameter selection criteria which is purely derived from the data. This makes the whole process generic and can be applied to other related tasks.

5.2 Tags Propagation

Wang et al. [18] introduced the concept of generating the tags for an image with no tags by comparing its visual features with a database of tagged images. In that work, they determine the tags for a query image with no tags by simply aggregating various visual features to find visually looking similar tagged images from a database of images called the auxiliary dataset. The tags extracted from these similarly matched images can then be used to create the text features (set of tags). They show that they outperform the results by combining these textual features with visual features on PASCAL VOC 2006 and PASCAL VOC 2007 datasets. In this [18] work the visual features used by the authors were a combination of four features namely-SIFT [15], GIST, color and gradient features. Figure 5.1 illustrates the process.

5.3 Selecting images for Tag Propagation

Apart from just using images with no tags for tag propagation, we believe that we can do better by trying to estimate the accuracy also based on the number of tags per image. Due to the noisy nature of the tags, some of the images with fewer tags could well have bad tags and thus equivalent to having no tags. So we believe that estimating the accuracy based on the number of tags per image could give stronger evidence to select images with tags in addition to the images with no tags.

5.3.1 Criteria

The criteria to select images for tag propagation follows a simple strategy where we first split the training set randomly into two sets. We use these sets for training and testing

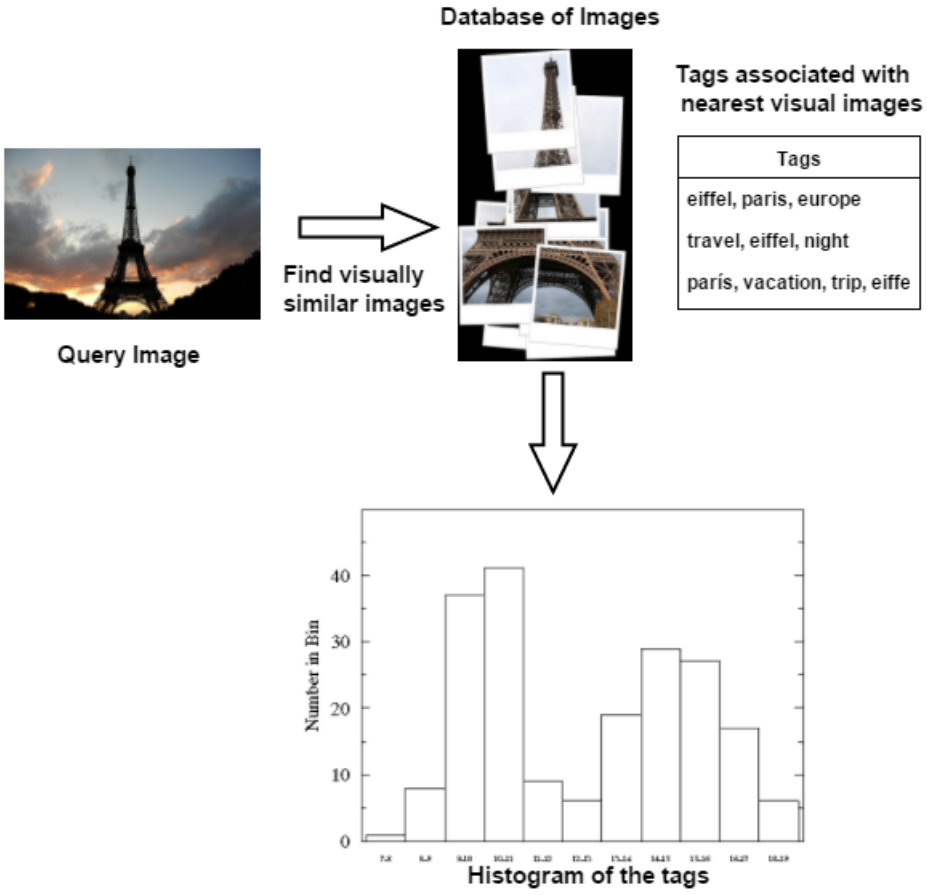


FIGURE 5.1: Tag Propagation.

purposes on a separately trained SVM based on the textual features of the images. The training set contains about 70 % of randomly selected images and about 30% for testing. The textual features are the binary vector based on the occurrence of the tags per image. We use this process to estimate the accuracy based on the numbers of tags per image.

5.3.2 Conclusion

We show the results of this process in figure 5.2 and find that most of the underperforming images in all the categories (top 10, top 20, top 50, top 100 and top 200) are the ones which have less than 5 tags per image. Images with more than 5 tags usually perform reasonably well. We notice some sporadic sharp dips in the accuracy, especially where

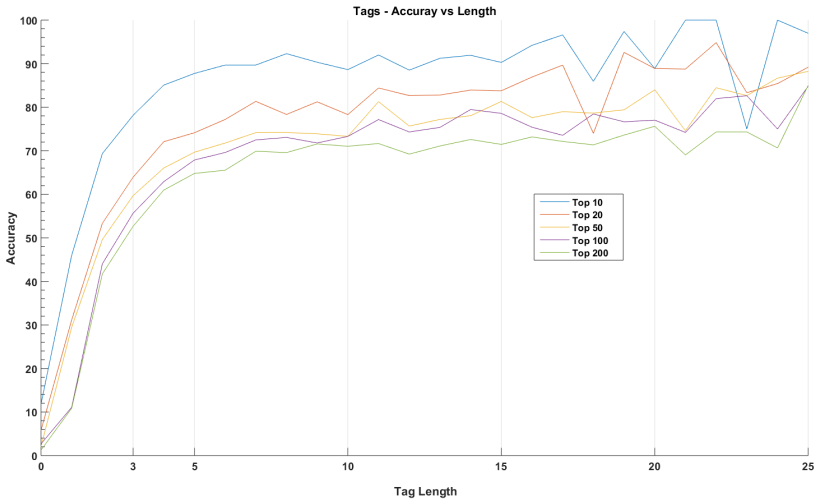


FIGURE 5.2: Accuracy variation with number of tag per image.

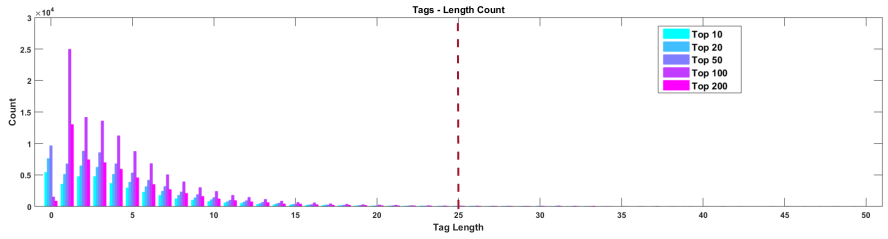


FIGURE 5.3: Distribution of number of tags per length.

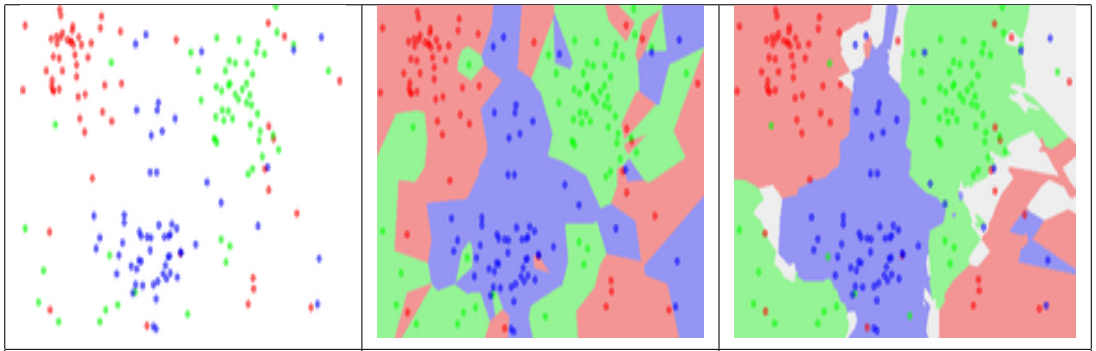
the number of tags per image is 18 and 23, the reason is that the data (images) for those particular number of tags are small and do not represent a large enough sample, and thus can be ignored. Figure 5.3 shows the histograms count for the complete training dataset across all the classification categories. Hence, for our criteria estimation, we ignore the number of tags per images more than 25.

5.4 K-NN Classification

5.4.1 K-NN concept

The concept behind k-Nearest neighbor (k-NN) is simple. We extract and identify some properties of the training data and match them with the properties of test data. The

data point with the closest matching property from the training data is inferred as the predicted label. Often looking at just the closest match to make a decision can cause errors, especially when the data has class imbalance or outliers. To illustrate this, we look at a particular example from [56], where we have a dataset with 3 classes each having 60 data point and the classification maps for different values of k (see table 5.1). The value of k is referred as the number of nearest neighbors that are accounted to make the decision.



a) The dataset	b) k=1	c) k=5

TABLE 5.1: KNN classification maps.

5.4.2 KNN classification maps and effect of k

For k =1, the decision boundaries seem more precise, but the outlier cases such as a green class inside the blue and red patch create small patches which often are incorrect generalizations, and cause an error. For k=5 there the decision boundaries learned are more smooth and provides a better generalization of the data. The white regions are where the decision can not be made due to a tie between nearest neighbors. So the classical problem in this approach is to find the optimal value of the hyperparameter k.

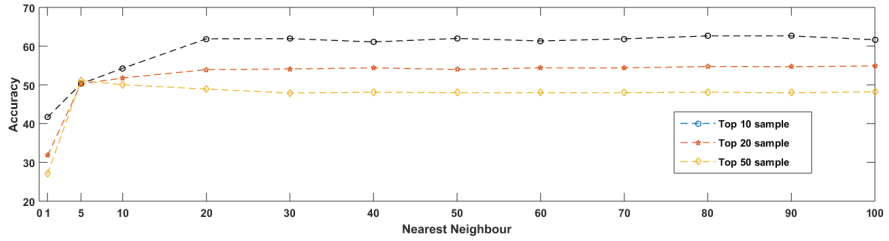


FIGURE 5.4: Selecting the hyperparameter for k-NN.

5.4.3 Parameter selection (setting k)

To find an optimal value of k we train on 30% randomly sampled training images and use 10% training data as validation set, and then run k-nearest neighbors for various values of k and observe the accuracy.We show the results on a portion of training data for 3 classification tasks in figure 5.4 (Top 10, Top 20 and Top 50) and observe stable result above 20 nearest neighbors. In our experiments on complete data, we use the value of k=20.

5.4.4 Curse of Dimensionality

One of the advantages of using k-Nearest Neighbor is its simplicity to implement and flexible decision boundaries, but it suffer from computational inefficiency during testing as computing distances with all the datapoints can be challenging. Also, the high dimensional features and large data would make this very inefficient and practically impossible to use. The curse of dimensionality in the k-NN context means that when the dimensionality and number of datapoints increases, the volume of search space in a neighbourhood of the query data point increases such that it contains a significant percentage of complete data, thus rendering the search process highly inefficient (which is comparable to a brute-force search). Using approximate nearest neighbors approach, we can boost the comparison cost at the expense of a little accuracy.

5.4.5 Fast approximate K-NN

In our system, we use Locality Sensitive Hashing [57, 58] mainly due to its sublinear running time. Though there are some hyper-parameters that need to be tuned for Locality Sensitive Hashing (LSH) based on its implementation and data, it is a one-time process and can be calculated by splitting a portion of the training data as a validation set and then running for a range of hyperparameters and selecting the best ones. We use the scikit [59] library for implementation, which provides good interface and reference [60] to tune these parameters. We explain details for the approximate nearest neighbor using LSH in section 5.5

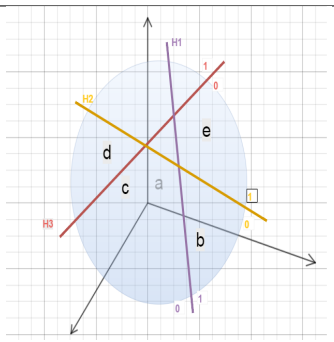
5.5 Locality Sensitive Hashing (LSH)

5.5.1 Introduction

Locality Sensitive Hashing (LSH) [57, 58] is a popular and efficient technique to find approximate nearest neighbors in a high dimensional space. The basic principle of this technique is that close or similar points in a high dimensional space should be relatively close when projected to a lower dimensional space. LSH is more efficient than a k-d tree [26] when the dimension become large, as its running time is sub-linear.

5.5.2 Method

LSH uses hash functions to project high dimensional points into a lower space and then ensures that the probability of collisions of similar points is large by using multiple hash functions with a sufficient number of bins. A hash function can be illustrated by a set of hyperplanes which divides a given space into multiple bins as in table 5.2. Supposing



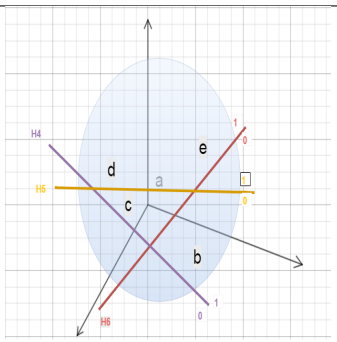


Table 1 for Hash1 = [H1 H2 H3]		Table 2 for Hash2 = [H4 H5 H6]	
Hashcode	Bin	Hashcode	Bin
000	a,c	000	-
001	d	001	-
010	-	010	-
011	-	011	-
100	b	100	b
101	-	101	c
110	e	110	-
111	-	111	a,d,e

TABLE 5.2: Locality Sensitive Hashing (LSH).

we have 4 [a, b, c, d, e] data-points in a three-dimensional space dividing the space by the respective hyperplanes of the two hash functions Hash1 and Hash2, then we can associate the data-points in respective bins.

Now given a query point in this space, we find the bin of the query point in all the hash tables, calculate the distance of the query point with all the points in the respective bins, and return the k nearest neighbors.

5.5.3 Running time

The running time can be split into two parts:

- T^H, which is the total time to hash a high dimensional data point.
- T^C, which is the total time to do comparisons within the bucket or bin of a hash function.

Let there be N data points with K dimensions and M hash functions, each having H hyperplanes then,

1. Cost of the hash operations: It is equivalent to the number of hash operations of a single data given by

$$\boldsymbol{T^H} = \mathcal{O}(M(KH)), \tag{5.1}$$

where $\mathcal{O}(KH)$= Cost of the dot product of K dimension data point with H hyperplanes, and M = Number of hash functions.

2. Cost of the comparisons: The cost of comparisons can be measured by the average number of collisions per bucket. The expectation of collision in each bin is given by $\mathbb{E}[Collision] = \frac{N}{2^H}$

The total comparison cost for each bin will have a factor of the dimensionality of the data point. Thus total cost of comparison of a K dimension collision points is $(\frac{KN}{2^H})$. Since there are M number of hash functions, so

$$\boldsymbol{T^C} = \mathcal{O}(M(\frac{KN}{2^H}) \tag{5.2}$$

The total running time is given by T^H(equation 5.1) + T^C(equation 5.2), and if we choose H = $\mathcal{O}(\log(N))$, then the running time reduces to $\boldsymbol{\mathcal{O}(\log(N))}$.

Chapter 6

System Design

6.1 Introduction

In this chapter, we describe the key components of our system. We first illustrate a thresholding step, which aims to remove the noisy tags that occur in the auxiliary dataset and can propagate after we apply tag propagation based techniques. This thresholding step can also help to alleviate the inconsistencies that can occur with approximate nearest neighbour matches. We present experimental results that show that applying thresholding step is helpful. We also illustrate our auxiliary dataset collection process. Finally, we present the workflow of the overall system.

6.2 Histogram thresholding

Since Approximate Nearest Neighbor (ANN) approaches may give false positives, we apply a thresholding step to minimize those negative effects and provide a better estimate. In this step, we first draw a histogram of the count of all the tags of the nearest

neighbours and then selectively remove tags which have counts less than K/factor, where K is the number of visually similar images already learned in section 5.4.3. Finally the remaining tags greater than the threshold are returned and converted into a binary vector.

6.2.1 Motivation

The visually similar tagged images from an auxiliary database can have all sorts of noise in the form of mislabeled tags, spelling inaccuracies, random noise, multi-lingual tags etc. It is thus important to get rid of the random noise and at the same time take advantage of contextual, spelling, and multilingual variance which can provide important information as seen in section 4.2.2.1 and table 4.1. We use both similarity and semantic matching (section 4.3) and collect useful tags while strictly rejecting all tags which do not fulfill the criteria. This also keeps the dimensionality of the textual feature consistent with the training set.

6.2.2 Selecting threshold factor

The threshold factor is another parameter that needs to be tuned and we run a simple experiment for three category task (Top 10, Top 20 and Top 50) and sample 30% images from the training dataset. We also keep 10% training data for validation. We have reported in section 5.4.3 and figure 5.4 that looking at 20 nearest neighbor fetches optimal results and thus keep the value of K=20 in our experiment. We run the experiment for a range of valid factor values and show the result in figure 6.1.

The threshold factor with value 1 would imply considering all tags that occur at least 20 times. This is a very strong bound and only selects those tags that occur in all the

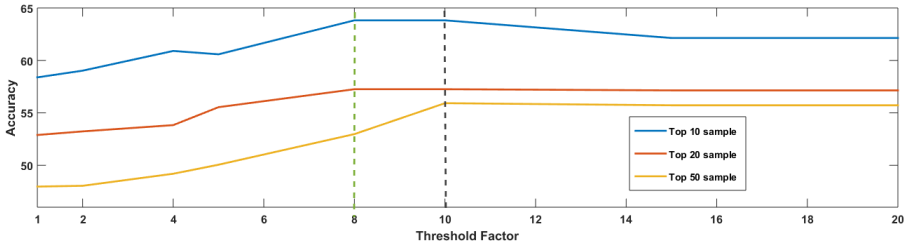


FIGURE 6.1: Affect of threshold factor on the accuracy.

nearest neighbors. On the other hand, a factor value of 20 would imply selecting all tags that occur at least once. This is a loose bound which considers all tags in nearest neighbors and this bound can reduce the accuracy if the nearest neighbor has lots of false positives.

In figure 6.1 we observe that the stronger bound performs poorly, which is expected since this discard many useful tags. Considering all the tags and keeping a loose threshold would also not be the best choice, it is prone to errors when there are false positives - we can observe this for top 10 category where accuracy falls after threshold factor 10. In our case, selecting tags which occur at least twice (factor=10, K =20) gives optimal performance.

6.3 Auxiliary dataset

We build the auxiliary dataset using a crawler which interfaces with the Flickr API services [4].

6.3.1 Crawler

The crawler is initialized with the GPS co-ordinates of the landmarks and then samples the top 50 tags from the training set for each of the landmarks. The crawler looks at

three criteria:

1. It looks at all geotagged images within a 300m radius from the GPS coordinates of the landmark and selects images which have 3 or more tags in the pool of sampled top 50 tags for that particular landmark. This process is repeated until 200 images are collected per landmark.

2. In case the crawler is unable to find 200 geotagged images, it selects any image which has 3 or more tags amongst the top 10 tags for that landmark. Here we put a tighter constraint which is to look within the top 10 tags per landmark, as the images otherwise would pick up more noise since they are not bounded by the region near the landmark and can be selected with no relevance to the landmark.

3. If any image already exists in training or testing dataset then we discard it from the auxiliary dataset.

The intention behind picking images which have at least 3 or more top tags is based on our results in figure 5.2 which show that images with less than 3 tags perform poorly if we consider only the textual features. So this ensures that when the images are picked from the auxiliary dataset the tags associated with them have less entropy and high information gain.

We believe that a dataset as large as [9] should be sufficient to identify important tags, and thus other tags from a relatively smaller auxiliary set would contribute more as noisy tags. We filter these noisy tags in the histogram threshold step (section 6.2.2). The images collected from step 1) are more precise as they have geotagged images and lie near the landmark. We put a much stronger constraint in step 2) where we match

with the top 10 tags because these images are not geotagged and thus are prone to more noise.

6.4 Workflow

Inspired by Wang et al's [18] concept we build a similar system but use features extracted from a Convolution Neural Network (CNN) [22] model instead of local features. Also instead of just using this method for images with no tags as shown in [18], we also apply this approach on some selected images which have tags but underperform on just the textual features. Our selection process is described in section 5.3.1. We use the Caffe [46] framework to build the CNN architecture, which acts as a feature extractor. For any query image, the CNN features are extracted and then used to find the closest visual matching images by searching its k-nearest neighbors in the auxiliary database. For finding these k-nearest neighbors, we investigated several approaches and selected locality sensitive hashing [51, 57] mainly because of its efficiency with respect to higher dimensions and large datasets. During training and testing time we do the following:

1. Select the images that meet the criteria (see section 6.2.2).
2. Generate new tags by applying tag propagation.
3. Create the text binary feature vector after histogram thresholding.

Finally, we train a SVM for classification. The workflow process is shown in Figure 6.2.

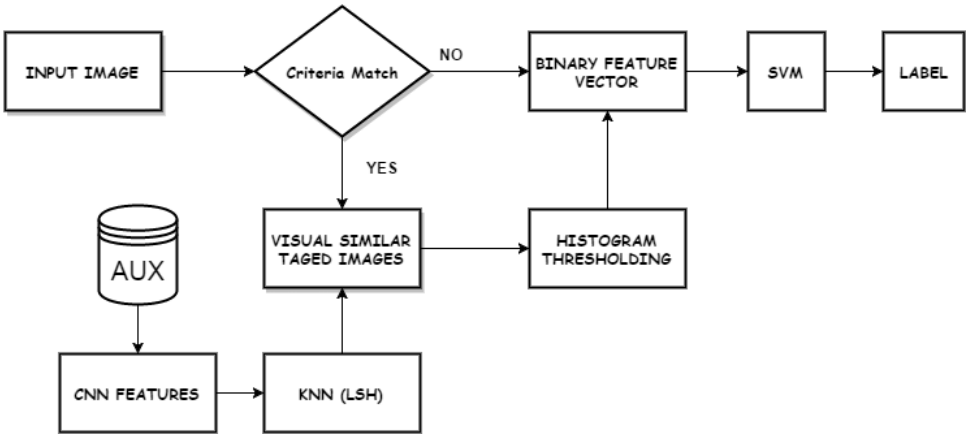


FIGURE 6.2: Workflow.

Chapter 7

Dataset

7.1 Introduction

The dataset that is used in this work was originally collected by Crandall et al. [9] as an initiative to create a large publically available dataset containing images from the social website. The motivation to select this dataset is its size, visual diversity and the image and corresponding label selection process. Below are some more details for each of them.

7.2 Crawler and dataset size

The data was collected using a crawler which interfaces with Flickr API services [4]. The crawler sampled a random photo-id from all possible ranges, looked up its photographer and all the social connections of the photographer, and then downloaded all geotagged images and metadata amongst them. The metadata consists of textual tags, geolocation, time, and date. This process was repeated for other photo ids and resulted in a collection

of 60 million images. They then filtered out images above a certain geo precision (city block size) reducing the dataset to 30 million images.

7.3 Selecting images and ground truth

Usually, dataset labeling involves analyzing the visual features, non-visual feature (metadata) or both and then assigning some ground truth value. Typically this is done by an individual or via crowdsourcing through Mechanical Turk [61] for larger datasets as shown by Imagenet [62]. This, unfortunately, brings some implicit bias based on people's perception of the visual and non-visual world. For instance, if you were to create a dataset of historical places, then a person who is well travelled would have a better chance of labeling accurately compared to a person who is less well traveled. The quality of labeling is another concern, and it becomes complicated when the intra-class variance is low. Expert knowledge especially plays a key role in these low intraclass variation tasks and often is not scalable for large dataset. However, the authors [9] use geotag information of images to cluster the photographs using the mean shift [63] algorithm. The peak of this density distribution is what they use as labels or landmarks. At the end they examine the top 500 peaks/landmarks in the distribution and collect the images within these clusters to build up approximately 2 million ground truth labeled images. Note that we do not use this geotag information again in our modeling, and thus overcome any bias that the process may have otherwise incurred.

Landmark	**Training(data per class)**	**Testing(data per class)**	**Images with no tags (%)**	**Images with one tags (%)**
Top 10	3503	3452	15.4	10.09
Top 20	2408	2338	15.78	10.59
Top 50	1281	1323	15.05	10.56
Top 100	1043	1023	1.43	23.96
Top 100	544	541	1.55	23.91

TABLE 7.1: The dataset.

7.4 Data partitioning

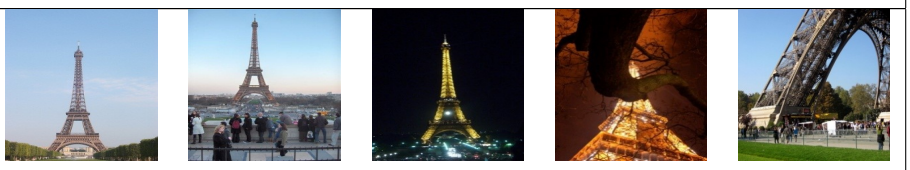
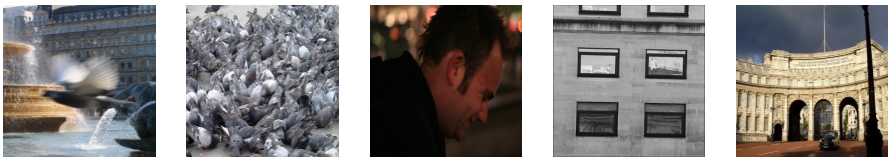
They [9] partition the data approximately equally for both training and testing while making sure that a user contributes to either training or testing. So training and testing are disjoint sets based on the user. No filtering is carried out based on the visual content or tags, and in fact the dataset contains a significant number of images with no or few tags. The data is finally normalized based on the number of images per class, and a clean set contains the partitioning shown in table 7.1.

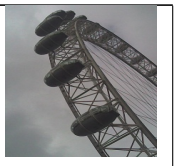
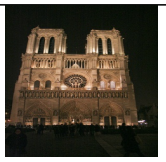
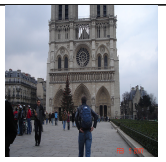
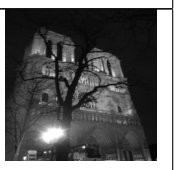
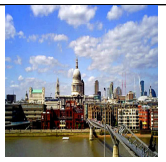
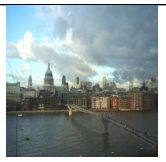
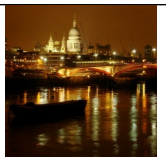
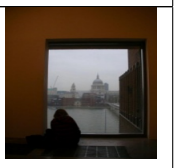
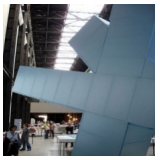
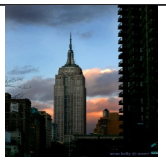
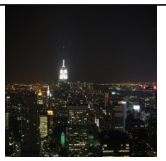
The resultant data is both visually and non-visually diverse and challenging. Since each landmark contains pictures within a cluster over a radius of 300 m, they pick up lots of visual and non-visual noise. For instance, there are pictures of events like marriages, parades etc, and nearby public or private places such as cafes, subway, buses, hotels, houses, etc. which do not hold any visual resemblance to an actual landmark other than they lie within a 300m radius. For tags, the variance occurs additionally due to language, spelling, context and morphological variations.

7.5 Data variance

In table 7.2 we show some examples of high confidence visual images (first row) and low confidence images (second row) for the top 10 landmarks. We also illustrate a snapshot

of the variance in textual data of each landmark by showing the top 10 occurring tags and the least 10 occurring tags.

Landmark		
	Top 10 Tags ———————- Least 10 Tags	Images (second rows of each landmark shows visual variance)
1.	paris, france, eiffel, eiffeltower, tower, toureiffel, europe,tour, night, travel	
	leaves,beauty, spain, espanha, bike, mexico, picture, nophotoshop, amateur, andrea	
2.	london, trafalgarsquare, england,europe, uk, trafalgar, square,fountain, unitedkingdom, nationalgallery	
	topv333,topv777, alex, bryan, bridge, bradford, foto, future, nophotoshop, aplusphoto	
3.	london, england, bigben, uk, westminster, parliament, big londres, ben, unitedkingdom,	
	fall,fave, germany, deutschland, photograph, cruise, earth, nophotoshop, james,portugal	

4.	london, londoneye, england, uk, eye, thames, unitedkingdom, wheel, southbank, europe					
	leaves,bird, gary, spain, merrygoround, leaf, earth, planet, andrea, james					
5.	paris, france, notredame, cathedral, europe, notre, dame, church, travel, notredamedeparis					
	topv333, topv777, beauty, espanha, ltytr1, palace, cruise, lovers, london, mom					
6.	london, tatemodern, tate, england, uk, art, modern, thames,bridge, southbank					
	fave,espaa, beach, russia, lovers, single, james, lambeth, noite, boats					
7.	newyork, nyc, manhattan, empitestate, newyorkcity, usa,ny,new, york, building					

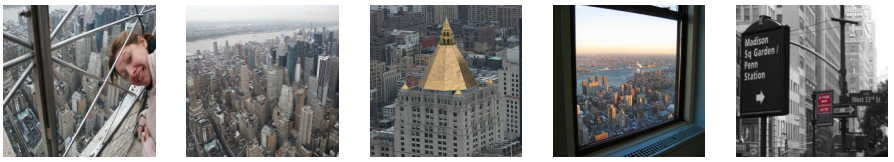
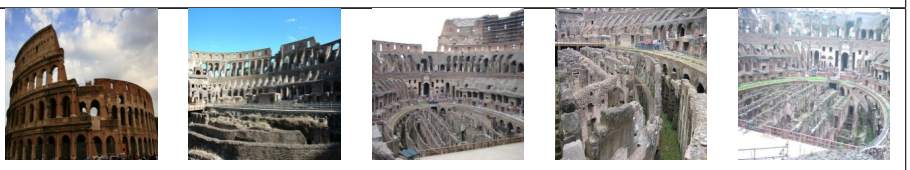
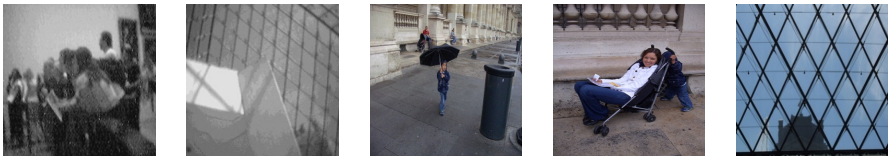
	wind,fall, germany, photograph, ltytr1, vakantie, single, james, church, nature	
8.	venice, italy, venezia, italia, europe, sanmarco, veneto,venedig, venecia, piazzasanmarco	
	green,photograph, andrea, james, london, mom, ave, arcos, england, superhearts	
9.	rome, italy, roma, colosseum, italia, colosseo, europe, coliseum, travel, geotagged	
	taiwan, deutschland, beauty, park, espaa, russia, mexico, poland, foto, earth	
10.	paris, louvre, france, museum, pyramid, europe, travel,pyramide, architecture, musedulouvre	
	deer,bike, belgium, lovers, london, mom, boat, noite, superhearts, tonemapped	

TABLE 7.2: Visual variance in the dataset.

In table 7.3 we show 3 high, 25 moderate and 25 low frequency occurring tags in the top 10 landmarks. Interestingly we can observe some repeating patterns:

- **High-frequency tags** capture the name of landmark 9 out of 10 times,as highlighted in bold. But the landmark name itself is not the top used tag (except Venice), rather the city name seems to be more dominant. These tags seem very important with respect to each landmark considering most of these landmark occur in different cities or even countries, but conversely, they could even be deemed very generic and less important if most or all the landmarks lie in the same county or city.

- **Moderate frequency tags** contain some interesting categories of tags:

  - Some of them are distinctive for each landmark. Specifically for Bigben, people often use 'clocktower', 'tower', 'bridge'. Though 'bridge' may not be a visual feature of Bigben, it is adjacent to the Westminster Bridge [64] and many images contain a visual of the bridge, so this tag adds useful information. Likewise, we can observe the same for London eye where people have used 'millenniumwheel', 'riverthames' and 'Ferris wheel'.

  - Language variations occur based on the geographical location of the landmark or the preferred language of the user, like Notre-Dame has tags such as 'francia' (Spanish) and 'frankreich' (German).

  - Some of the tags are random or seems based on some temporal event. Examples of such tags are '2003', '2006', '2007', '2008' etc. An example of a possible explanation may be global events such as the FIFA world cup [65] which was

held in 2006 in Germany, which could have increased tourists in Europe who also visited other landmarks as well in nearby European counties.

- There are random noisy tags as well e.g. 'Nikon', 'bw' , 'st', 'favorites' etc.
- **Low-frequency tags** are mostly random noise due to spelling mistakes or personalized tags which contain the name of the people or emotions e.g. 'gary', 'bottle', '1870mmf3545g', 'charlie', 'drunk', 'mom' etc.

	Landmark		
	High Frequency	**Moderate Frequency**	**Low Frequency**
1.	paris france **eiffel**	Nikon, 2006, blackandwhite, latoureiffel, trip, pars, lights, bw, seine, eiffelturm, frankreich, 2008, Europa, city, torreeiffel, vacation, bird, kids, july, boy, martin, traffic, politics, cameraphone, muslim	leaves, beauty, spain, espanha, bike, mexico, picture, nophotoshop, amateur, andrea, mom, tonemapped, fire, parade, mountains, uk, mobile, images, drunk, cars, ice, horse, york, newyork, stilllife
2.	London **Trafalgarsquare** england	gallery, blue, demonstration, architecture, march, bigben, trip, sky, street, vacation, water, sculpture, lion, city, column, protest, 2008, people, canon, 2006, londra, greatbritain, night, statue, nelson	topv333, topv777, alex, bryan, bridge, bradford, foto, future, nophotoshop, aplusphoto, fire, butterfly, charlie, cars, feathers, 1870mmf3545g, central, game, colors, colours, door, poor, wait, around, edited
3.	london england **bigben**	houses, station, 2008, building, trip, british, canon, bridge, palaceofwestminster , 2006, eye, sky, vacation, clocktower, river, geotagged, city, architecture, 2007, tube, greatbritain, londra, underground, britain, tower	fall, fave, germany, deutschland, photograph, cruise, earth, nophotoshop, james, portugal, noite, superhearts, tonemapped, atardecer, images, charlie, beautiful, pretty, america, 1870mmf3545g, man, landscapes, nyc, empire, times

4.	london **londoneye** England	capsule, architecture, bridge, british, big, pod, housesofparliament, inglaterra, 2008, millenniumwheel, londra, clouds, blue, city, 2006, canon, vacation, riverthames, ferriswheel, travel, greatbritain, britain, parliament, sky, 2006	leaves, bird, gary, spain, merrygoround, leaf, earth, planet, andrea, james, church, museum, nokia, noite, superhearts, ships, wood, mobile, manipulated, drunk, boarding, brighton, ice, horse, garden
5.	paris france **notredame**	blackandwhite, notredamecathedral, pars, canon, cathedral, statue, nikon, city, trip, frankreich, Europa, holiday, stainedglass, 2008, 2006, bw, night, cathdrale, parigi, seine, gargoyle, vacation, francia, frankrijk, cathedralenotredame	topv333, topv777, beauty, espanha, ltytr1, palace, cruise, lovers, london, mom, arcos, england, nature, superhearts, faces, ireland, mobile, abstract, manipulated, images, orange, south, metal, bottle, 1870mmf3545g
6.	london **tatemodern** tate	street, 2008, riverthames, slide, maman, blackandwhite, millennium, pauls, st, slides, streetart, sculpture, nikon, millennium, modernart, canon, louisebourgeois, travel, spider, greatbritain, southwark, 2007, people, night, dorissalcedo	fave, espaa, beach, russia, lovers, single, james, lambeth, noite, boats, atardecer, rust, batman, feathers, shopping, new, centre, side, stilllife, toy, posters, picnik, frost, moon, ceiling
7.	newyork nyc **empirestatebuilding**	lights, canon, nuevayork, nikon, trip, 2006, geotagged, panorama, sky, 2005, us, skyscrapers, cityscape, holiday, urban, midtown, chryslerbuilding, bigapple, 2007, unitedstates, travel, vacation, buildings, view, architecture	wind, fall, germany, photograph, ltytr1, vakantie, single, james, church, nature, fire, wild, abstract, woman, gothamcity, metal, feathers, lift, 1870mmf3545g, man, landscapes, village, centre, side, mid

8.	**Venice** italy venezia	pigeon, 2008, bw, canon, pigeons, holiday, vacation, mask, church, architecture, europa, campanile, 2007, square, italien, basilica, carnevale, carnival, italie, marco, places, olympus, favorites, 2003, rome	green, photograph, andrea, james, london, mom, ave, arcos, england, superhearts, tonemapped, mobile, artistic, sign, beer, drinking, town, colours, bridges, port, roman, windows, center, national, tree
9.	rome italy roma	kolosseum, honeymoon, tourism, friends, landscape, portrait, piazza, canoneos400d, stadion, viaje, fabio, anfiteatro, colleseum, pistillo, bisbi, music, street, d50, abigfave, blackwhite, 1999, arquitectura, highschool, monuments, eurotrip	taiwan, deutschland, beauty, park, espaa, russia, mexico, poland, foto, earth, past, 200000000stagelovers, london, mom, portugal, river, england, cathedral, wales, augustus, flowers, trash, trashcan, plant, Ireland
10.	paris **louvre** france	pars, iledefrance, honeymoon, sky, reflection, city, musee, impei, piramide, people, 2006, europa, museedulouvre, holiday, louvremuseum, lelouvre, frankreich, canon, 2008, trip, bw, nikon, museo, statue, sculpture	deer, bike, belgium, lovers, london, mom, boat, noite, superhearts, tonemapped, butterfly, flowers, medieval, artistic, sign, 1870mmf3545g, shopping, garden, town, tunnel, stilllife, vintage, ivory, monochromia, colors

TABLE 7.3: Textual variance in the dataset.

Chapter 8

Results

8.1 Introduction

In this chapter, we show the results for 5 classification categories (top 10, top 20, top 50, top 100, and top 200) which collectively contains around 730,000 images. This set of images is divided equally into training and testing sets as described in section 7.4. We also show the speedup gain with our approach using a linear SVM for classification.

8.2 Overall results

The baseline for comparison in all the categories is the results which are obtained on binary features with all tags. Our results in table 8.1 show that performing filtering (FT) on the tags significantly reduces the dimensionality for all categories, up to 96% for top 10 categories. But filtering can reduce the available information and impact the accuracy as well, though marginally observe a drop in accuracy around 0.25% to 0.7%.

8.3 Effect of tag matching

We also apply word matching (WM) which is a combination of similarity and semantic matching (section 4.3) to gain useful information from the rejected features or tags during filtering. We observe an overall gain of 1.5% in the accuracy. Note that in word matching (WM), after the filter step, we match tags from rejected set that have a high correlation with tags in the majority set. Since we just find matches and represent them with existing tags, the dimensionality of the system is unaffected and remains same. Intuitively we are reducing the sparsity of the feature vectors.

Categories	**Feature Dimension**	**Accuracy (%)**	**Dimension Reduction (%)**	**Delta Accuracy (%)**
Top 10 (base)	6659	69.25	-	-
Top 10 + FT	256	69	96	- 0.25
Top 10 + FT + WM	256	70.51	96	+ 1.26
Top 10 + FT + TP	256	**80.04**	96	+ 10.79
Top 20 (base)	8552	57.65	-	-
Top 20 + FT	422	57.07	95	- 0.58
Top 20 + FT + WM	422	59.22	95	+ 1.57
Top 20 + FT + TP	422	**64.87**	95	+ 7.22
Top 50 (base)	10976	52.65	-	-
Top 50 + FT	948	52.37	91	- 0.28
Top 50 + FT + WM	948	54.44	91	+ 1.79
Top 50 + FT + TP	948	**57.49**	91	+ 4.84
Top 100 (base)	16976	50.4	-	-
Top 100 + FT	1817	49.95	89	- 0.45
Top 100 + FT + WM	1817	51.86	89	+ 1.46
Top 100 + FT + TP	1817	**53.88**	89	+ 3.48
Top 200 (base)	19197	47.2	-	-
Top 200 + FT	4015	46.5	79	- 0.7
Top 200 + FT + WM	4015	48.6	79	+ 1.4
Top 200 + FT + TP	4015	**49.23**	79	+ 2.03

TABLE 8.1: Results.

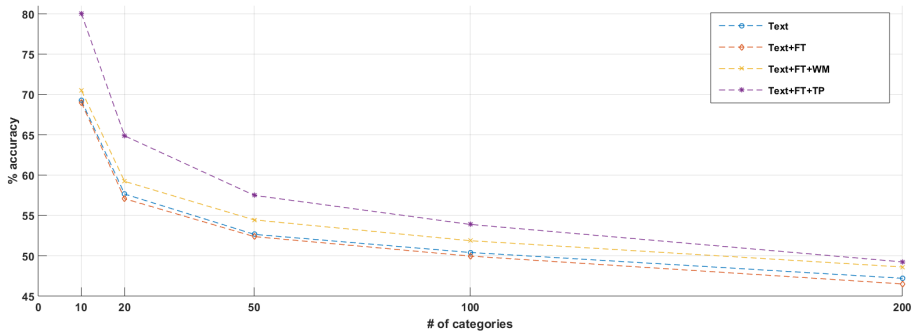


FIGURE 8.1: Results.

8.4 Effect of tag propagation

Applying Tag propagation (TP) gives the maximum gain in accuracy. For top 10 classifications we observe a 10.79 percentage gain. The dataset has about 25% of images with either no tag or 1 tag (table 7.1) which perform poorly on textual data as shown in figure 5.2. We apply tag propagation to resolve this problem and see significant improvement for images with 3 or fewer tags. Figure 8.1 shows the accuracy boost (dotted line) after performing tag propagation. We observe 5 to 10 times better results for images which have no tags and on average 7 percentage gain on images with one tag.

8.5 Speedup

Dimension reduction with filtering greatly speeds up the training time. Fig 8.2 shows the training time on a 4 core CPU machine for a linear SVM. For the 200 classification category task, we get up to 7 times reduction in training speed.

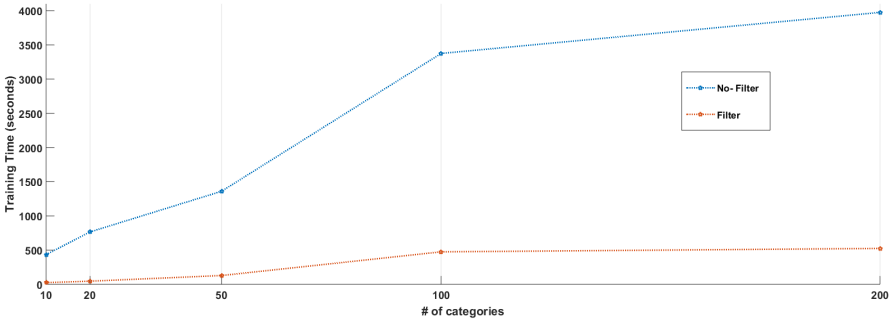


FIGURE 8.2: Speedup.

8.6 Unsuccessful experiment

8.6.1 Pairwise tags

Our work with the crawler (section 6.3.1) revealed some interesting thoughts that motivated us to use pairwise tags. We observed that searching images using the pair of relevant top tags (no geolocation) gives better results than just using a single top tag. Though the number of images retrieved is lower in that case, but the quality of images retrieved is better. Using a single tag retrieves more images with artifacts or objects related to the landmark name rather than the actual place or nearby area. Based on this intuition that there exists some relationship amongst tags, we tried a basic approach where we consider all pairwise tags as additional features along with the single tags. So for a 10-way classification task which originally had 256 dimensions after initially filtering it now had an additional (256x255)/2 = 32640 dimension or pairwise relations. The accuracy after including these pairwise relations underperformed compared to the baseline (Figure 8.3).

Possible Reason for failures:

- As observed from figure 5.2 the low performing data are tags with lengths 0, 1,

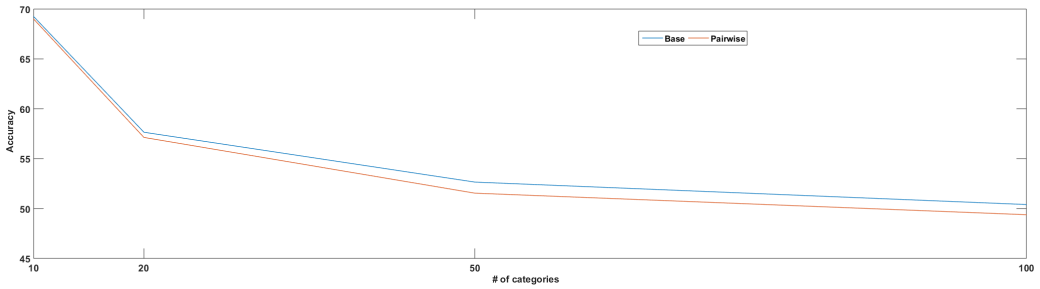


FIGURE 8.3: Accuracy with pairwise.

2, 3 and 4. The respective number of pairwise relations are 0, 0, 1, 3 and 6 for those lengths. The data with 5 or more number of tags are performing fairly well already and therefore this is one of the reason why pairwise may not help.

- All the pairwise relations are binary, which means that if two tags exist, we associate a value 1 and otherwise 0. This may not be the best strategy considering the fact that a relation amongst the two top tags is much stronger than the relation between a top tag and a noisy tag. So learning the weights amongst the relationship may be more helpful.

Chapter 9

Conclusion

9.1 Conclusion

The focus of the thesis was to explore and understand the textual representations (user annotated tags) in large scale image collections, the ultimate goal being to improve overall landmark classification. In the first part of this work, we aim to separate the noisy and useful tags, and apply tag propagation to learn missing tags in the images. Particularly in chapter 4, we propose a filtering criteria which separates the noisy and useful tags. We use similarity and semantic matching to fix irregularities in tags such as spelling, context and language variations. In chapter 5, we first apply tag propagation for missing tags and then extend it for the underperforming images on textual features. Our selection criteria and parameter tuning is data driven and thus can be applied generically with many datasets. We use locality sensitive hashing to construct the nearest neighbor search in an efficient and scalable manner.

The second half of this work illustrates the dataset, the system components, and shows experimental results. In chapter 6, we study how we can apply a simple thresholding

strategy to counter the noisy variations in the auxiliary dataset. This strategy can also help to remove the effect of inconsistent matches in nearest neighbor search. We describe the crawler rules that were used to build the auxiliary dataset. We describe the dataset in chapter 7, and give examples of visual and textual variance. Finally, in chapter 8 we show the experimental results of our system. We show results and observe improvement in five classification categories when we apply word matching and tag propagation. We also show that applying filtering can reduce the the dimensionality of the feature vector and thus make our system significantly faster.

9.2 Future work

Some interesting extensions of this work would include:

1. Extending semantic and semantic matching for multiple languages to help to cover an even more diverse set of locations around the world. In this work, the dataset had predominantly English speaking geographical locations.

2. In this work we used binary weights for all tags in our textual feature vector, i.e. we treat all the tags equally. Instead, dynamically learning the weights of tags based on some preference criteria like tag ranking should be more powerful.

3. Combining the state of the art convolutional neural net features with these improved textual features should provide interesting insights which can further help to understand the correlation amongst these features.

Bibliography

[1] Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg.

[2] CNN block diagram. http://www.frontiersin.org/files/Articles/172022/frobt-02-00036-HTML/image_m/frobt-02-00036-g001.jpg.

[3] Internet trends. http://www.kpcb.com/internet-trends.

[4] FlickrApi. https://www.flickr.com/services/api/.

[5] facebook. https://www.facebook.com/.

[6] Google photos. https://photos.google.com/.

[7] Instagram. https://www.instagram.com/?hl=en.

[8] EXIF format. https://en.wikipedia.org/wiki/Exchangeable_image_file_format.

[9] David J Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg. Mapping the world's photos. In *Proceedings of the 18th international conference on World wide web*, pages 761–770. ACM, 2009.

[10] Y. T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T. S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *Computer Vision and Pattern Recognition*, pages 1085–1092, 2009.

[11] Wikitravel. http://wikitravel.org/en/Main_Page.

[12] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *International Conference on Computer Vision*, pages 1–8. IEEE Computer Society, 2007.

[13] Yunpeng Li, David J Crandall, and Daniel P Huttenlocher. Landmark classification in large-scale image collections. In *International Conference on Computer Vision*, pages 1957–1964, 2009.

[14] Lyndon S Kennedy, Shih-Fu Chang, and Igor V Kozintsev. To search or to label?: predicting the performance of search-based automatic image classifiers. In *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 249–258. ACM, 2006.

[15] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[16] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(8):837–842, August 1996.

[17] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[18] G. Wang, D. Hoiem, and D. A. Forsyth. Building text features for object image classification. In *Computer Vision and Pattern Recognition*, pages 1367–1374, 2009.

[19] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

[20] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. *Local features and kernels for classification of texture and object categories: An in-depth study*. PhD thesis, INRIA, 2005.

[21] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, pages 1470–1477. IEEE, 2003.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[23] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, pages II: 2169–2178, 2006.

[24] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition*, pages 3370–3377. IEEE Computer Society, 2012.

[25] Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 271–280, Austin, Texas, 25–27 January 1993.

[26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[27] J. H. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[28] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[29] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, November 2008.

[30] Dong Liu, Shuicheng Yan, Xian-Sheng Hua, and Hong-Jiang Zhang. Image retagging using collaborative tag propagation. *Multimedia, IEEE Transactions on*, 13 (4):702–712, 2011.

[31] Hao Xu, Jingdong Wang, Xian-Sheng Hua, and Shipeng Li. Tag refinement by regularized lda. In *Proceedings of the 17th ACM International Conference on Multimedia*, pages 573–576. ACM, 2009.

[32] Dan Guo and Pengfei Gao. Complex-query web image search with concept-based relevance estimation. *International Conference on World Wide Web*, pages 1–18, 2015.

[33] Xirong Li, Cees GM Snoek, and Marcel Worring. Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, 11(7):1310–1322, 2009.

[34] Meng Wang, Bingbing Ni, Xian-Sheng Hua, and Tat-Seng Chua. Assistive tagging: A survey of multimedia tagging with human-computer joint exploration. *ACM Comput. Surv.*, 44(4):25:1–25:24, September 2012.

[35] Guangyu Zhu, Shuicheng Yan, and Yi Ma. Image tag refinement towards low-rank, content-tag prior and error sparsity. In *Proceedings of the International Conference on Multimedia*, pages 461–470. ACM, 2010.

[36] Dong Liu, Meng Wang, Linjun Yang, Xian-Sheng Hua, and HongJiang Zhang. Tag quality improvement for social images. In *International Conference on Multimedia and Expo*, pages 350–353. IEEE, 2009.

[37] S Arya. Optimized hypergraph based social image search using visual-textual joint relevance learning. *IOSR Journal of Computer Engineering*, 1(16):40–49.

[38] Yue Gao, Meng Wang, Zheng-Jun Zha, Jialie Shen, Xuelong Li, and Xindong Wu. Visual-textual joint relevance learning for tag-based social image search. *Image Processing, IEEE Transactions on*, 22(1):363–376, 2013.

[39] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[40] ReLU. https://en.wikipedia.org/wiki/Rectifier_(neural_networks)#/media/File:Rectifier_and_softplus_functions.svg.

[41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[42] Backpropagation. https://en.wikipedia.org/wiki/Backpropagation.

[43] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[44] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[45] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[46] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[47] Caffe based ALexNet architecture. https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet.

[48] ImageNet. http://www.image-net.org/challenges/LSVRC/2012/results.html.

[49] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, pages 1470–1477, 2003.

[50] Google Translate. https://translate.google.com/.

[51] John W Ratcliff and David E Metzener. Pattern-matching-the gestalt approach. *Dr. Dobbs Journal*, 13(7):46, 1988.

[52] Longest Common Subsequence. https://en.wikipedia.org/wiki/Longest_common_subsequence_problem/.

[53] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[54] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1150, 2006.

[55] Yuhua Li, Zuhair Bandar, and David McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge Data Engineer*, 15(4):871–882, 2003.

[56] K-nearest scikit. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[57] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Very Large Data Bases*, volume 99, pages 518–529, 1999.

[58] Locality Sensitive Hashing. https://www.slaney.org/malcolm/yahoo/Slaney2008-LSHTutorial.pdf.

[59] Scikit home. http://scikit-learn.org/stable/, .

[60] Scikit approximate nearest neighbors. http://scikit-learn.org/stable/auto_examples/neighbors/plot_approximate_nearest_neighbors_hyperparameters.html, .

[61] Mechanical Turk. https://www.mturk.com/mturk/welcome.

[62] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[63] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 603–619, May 2002.

[64] Westminster Bridge. https://en.wikipedia.org/wiki/Westminster_Bridge.

[65] 2006 FIFA World Cup. https://en.wikipedia.org/wiki/2006_FIFA_World_Cup.