

FIGURE 21.7: Multidimensional scaling allows us to compute locations on a screen that are consistent with inter-image distances, and so lay out images in a suggestive way. Frame 1 shows 500 images, the response to a query for a desert landscape. Multidimensional scaling has been used to compute locations for the thumbnails. Notice how strongly different images are far apart (this image distance places strong weight on global color distances, and the purple images are to the left of this frame, while more yellow images are to the right). The user then clicks on the black dot (near top right of the frame), and the 100 images closest to that point are selected; a new multidimensional scaling is computed for this subset of images, and they are laid out to give frame 2. The layout changes because the statistics of distances have changed. Again, the user clicks on the black dot (lower center of the frame), to select a subset of 20 images; again, a new scaling is computed for this subset, and they are laid out to give frame 3. *This figure was originally published as Figure 4 of “A Metric for Distributions with Applications to Image Databases,” by Y. Rubner, C. Tomasi, and L. Guibas, Proc. IEEE ICCV 1998, © IEEE, 1998.*

cluster center, then see all the elements of the cluster.

## 21.4 PREDICTING ANNOTATIONS FOR PICTURES

Appearance-based searches for images seem to be useful only in quite special applications. In most cases, people appear to want to search for images using more general criteria, like what objects are present, or what the people depicted are doing (Jörgensen 1998). These searches are most easily specified with words. Relatively few pictures come with keywords directly attached to them. Many pictures have words nearby, and a fair strategy is to treat some of these words as keywords (Section 21.4.1). More interesting to us is the possibility of learning to predict good annotating words from image features. We could do so by predicting words from the whole image (Section 21.4.2). Words tend to be correlated to one another, and prediction methods that take this into account tend to perform better (Section 21.4.3). Linking names in a caption to faces in an image is an important special case (Section 21.4.4), which suggests a general strategy of thinking about correspondences between image regions and words (Section 21.4.5).

Image annotation is important for two reasons: first, there are useful practical

applications in image search; and second, it emphasizes a question that is crucial for object recognition—what should we say about a picture? The main distinction between methods is how they deal with the fact that image annotations tend to be quite strongly correlated. Some methods model correlations explicitly, and others allow words to be conditionally independent given image structures, and allow the correlation between image structures to encode the correlation between words.

#### 21.4.1 Annotations from Nearby Words

The name of the image might yield words. A picture called `mydog.jpg` likely shows a dog, but it is hard to tell what might be in `13789.jpg`. If the image appears on a web page, then it is most likely in an `IMG` tag. The standard for these tags requires an `alt` attribute, which is the text that should appear when the image can't be displayed. Words that appear in this attribute could also be used as keywords. Unfortunately, HTML does not have a single standard way of captioning images, but a text matcher could identify some of the many methods used to display images with captions. If a caption is found, words that appear in the caption might be used as keywords. Finally, one could use words that render on the web page near to the image. One could also cluster any or all of these sources of words to try to suppress noise, and attach cluster tags rather than keywords to images. Notice that some kinds of web page might be more fruitful for this kind of analysis than others. For example, a catalog might contain a lot of images whose identity is quite obvious.

If you experiment informally with commercial image search engines, you will notice that most pictures returned for simple one-word object queries are pictures in which a single object is dominant. This underlines an important point. The images that we are dealing with in Internet search applications may not be at all like the images that appear at the back of your eye. There are quite strong relationships between object recognition and image search, but they're not the same problem. Apart from this very important point, these pictures are there either because people want such pictures (and so the search results are biased to place them at the top), or because search engines are biased toward finding them (because they look in places where such pictures are prominent and easily obtained).

#### 21.4.2 Annotations from the Whole Image

The simplest way to attach words to pictures is to use a classifier to predict one word for the whole picture (methods described in Chapter 16). The vocabulary might need to be quite big, and the approach might not work well for images that don't contain a dominant object. A more attractive approach is to try to predict more than one word. A simple and natural way to try and do this is to annotate each image with a binary code. This code is the length of the vocabulary. Each bit in the code corresponds to a word in the vocabulary, and there is a one if the word is present and a zero if it is absent. We then find the set of codes that actually appear (a set that will be much smaller than the set that could appear. Typical vocabularies might run to 400 words or more, and the number of possible codes is then  $2^{400}$ ). We treat each code that actually appears as a word, and build a multi-class classifier. This sounds easy, but is wholly impractical, because there

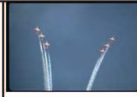

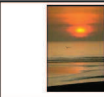


					
Predicted keywords	sky, jet, plane, smoke, formation	grass, rocks, sand, valley, canyon	sun, water, sea, waves, birds	water, tree, grass, deer, white-tailed	bear, snow, wood, deer, white-tailed
Human annotation	sky, jet, plane, smoke	rocks, sand, valley, canyon	sun, water, clouds, birds	tree, forest, deer, white-tailed	tree, snow, wood, fox

FIGURE 21.8: A comparison of words predicted by human annotators and by the method of Makadia *et al.* (2008) for images from the Corel5K dataset. *This figure was originally published as Figure 4 of “A New Baseline for Image Annotation,” by A. Makadia, V. Pavlovic, and S. Kumar, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 5304, 2008 © Springer, 2008.*

will be very few examples for each code. By failing to pool data, we are wasting examples. For example, our strategy would treat an image labeled with “sheep,” “field,” “sky,” and “sun” as completely different from an image labeled “sheep,” “field,” and “sky,” which is absurd.

So we must treat words individually; but words tend to be correlated, and we should exploit this correlation in our prediction methods. Straightforward methods are extremely strong. Makadia *et al.* (2008) describe a method based around  $k$ -nearest neighbors, which performs as well as, or better than, more complex methods in the literature (see also Makadia *et al.* (2010)). They use color and texture features in a straightforward labeling algorithm (Algorithm 21.1). They compare their method to a number of more complicated methods. It is highly competitive (see Table 21.2 for comparative performance information).

To predict  $n$  tags:

obtain the  $k$ -nearest neighbors of the query image

sort the tags of the closest image in order of frequency, then report the first  $n$

If the closest image has fewer than  $n$  tags:

rank tags associated with the other  $k - 1$  neighbors according to:

(a) their cooccurrence with the tags already chosen and

(b) their frequency in the  $k$ -nearest neighbor set.

The remaining tags are the best in this ranked set.

**Algorithm 21.1:** Nearest Neighbor Tagging.

Some of the tags on the nearest neighbor might be much rarer than the tags on the second nearest neighbor. To account for this, we can modify the tagging algorithm to account for both similarity between nearby neighbors and tag frequency. One plausible strategy is due to Kang *et al.* (2006). For a given query image, they build a confidence measure associating each tag with that image. Larger values of the measure associate the tag to the image more strongly. To do so, they require a

ranking of the tags in importance; this ranking is given by a vector of values, one per tag. Write  $\alpha_j$  for the ranking value of the  $j$ th tag;  $\mathbf{x}_i$  for the feature vector of the  $i$ th training example image, and  $\mathbf{x}_t$  for the feature vector of the test image;  $K(\cdot, \cdot)$  for a kernel comparing images;  $\Omega(\cdot, \cdot)$  for a kernel comparing *sets* of tags; and  $z_{t,k}$  for the confidence with which the  $k$ th tag is associated with the test image. We must compute  $z_{t,k}$ . Kang *et al.* use a submodular function argument to derive an algorithm for concave  $\Omega$ , though in their examples they use

$$\Omega(\mathcal{S}, \mathcal{S}') = \begin{cases} 0 & \text{if } \mathcal{S} \cap \mathcal{S}' \neq \emptyset \\ 1 & \text{otherwise} \end{cases}.$$

Their method is a straightforward greedy algorithm, which appears in Algorithm 21.2. Once we have the confidence for each tag for a test image, we can choose the tags to report with a variety of strategies (top  $k$ ; top  $k$  if all confidences exceed a threshold; all whose confidence exceeds a threshold; and so on). This method works well (see Table 21.2).

Using the notation of the text

For  $k = 1, \dots, m$ :

Let  $\mathcal{T}_k = \{1, 2, \dots, k\}$

$f(\mathcal{T}_k) = \sum_{i=1}^n K(\mathbf{x}_i, \mathbf{x}_t) \Omega(\mathcal{S}_i, \mathcal{T}_k)$

$z_{t,k} = f(\mathcal{T}_k) - f(\mathcal{T}_{k-1})$

**Algorithm 21.2:** Greedy Labeling Using Kernel Similarity Comparisons.

### 21.4.3 Predicting Correlated Words with Classifiers

When word annotations are heavily correlated, we could predict some words based on image evidence, and then predict others using the original set of word predictions. A more efficient method is to train the classifiers so that their predictions are coupled. For example, we will train a set of linear support vector machines, one per word. Write  $N$  for the number of training images;  $T$  for the size of the tag vocabulary;  $m$  for the number of features;  $\mathbf{x}_i$  for the feature vector of the  $i$ th training image;  $\mathcal{X}$  for the  $m \times N$  matrix  $(\mathbf{x}_1, \dots, \mathbf{x}_N)$ ;  $\mathbf{t}_i$  for the tag vector of the  $i$ th image (this is a 0-1 vector whose length is the vocabulary size, with a 0 value if the tag corresponding to that slot is absent and a 1 if it is present); and  $\mathcal{T}$  for the  $T \times N$  matrix  $(\mathbf{t}_1, \dots, \mathbf{t}_N)$ . Training a set of linear SVMs to predict each word independently involves choosing a  $T \times m$  matrix  $\mathcal{C}$  to minimize some loss  $\mathcal{L}$  that compares  $\mathcal{T}$  to the predictions  $\text{sign}(\mathcal{C}\mathcal{X})$ . If we chose to use the hinge loss, then the result is a set of independent linear SVMs.

Loeff and Farhadi (2008) suggest that these independent linear SVMs can be coupled by penalizing the rank of  $\mathcal{C}$ . Assume for the moment that  $\mathcal{C}$  does have low rank; then it can be factored as  $\mathcal{G}\mathcal{F}$ , where the inner dimension is small. Then  $\mathcal{C}\mathcal{X} = \mathcal{G}\mathcal{F}\mathcal{X}$ . The term  $\mathcal{F}\mathcal{X}$  represents a reduced dimension feature space

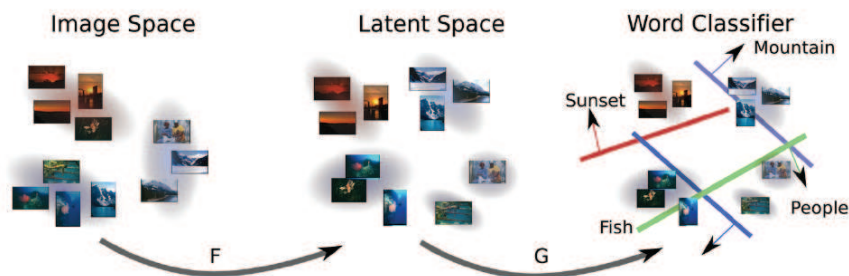


FIGURE 21.9: One way to build correlated linear classifiers is to learn a matrix of linear classifiers  $\mathcal{C}$  while penalizing the rank of  $\mathcal{C}$ . A low rank solution factors into two terms as  $\mathcal{C} = \mathcal{G}\mathcal{F}$ . The term  $\mathcal{F}$  maps image features to a reduced dimensional space of linear features, and  $\mathcal{G}$  maps these features to words. The word predictors must be correlated, because the number of rows of  $\mathcal{G}$  is greater than the dimension of the reduced dimensional feature space. *This figure was originally published as Figure 1 of “Scene Discovery by Matrix Factorization,” by N. Loeff and A. Farhadi, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 5304, 2008 © Springer, 2008.*

(it is a linear map of the original feature space to a lower dimensional feature space; Figure 21.9). Similarly,  $\mathcal{G}$  is a set of linear classifiers, one per row. But these classifiers have been coupled to one another (because there are fewer linearly independent rows of  $\mathcal{G}$  than there are classifiers, see Figure 21.9).

Penalizing rank can be tricky numerically. One useful measure of the rank is the *Ky-Fan norm*, which is the sum of the absolute values of the singular values of the matrix. An alternative definition is

$$\lambda \|\mathcal{C}\|_{kf} = \inf_{\mathcal{U}, \mathcal{V} | \mathcal{U}\mathcal{V} = \mathcal{C}} (\|\mathcal{U}\|^2 + \|\mathcal{V}\|^2).$$

Loeff and Farhadi learn by minimizing

$$\mathcal{L}(\mathcal{T}, \mathcal{C}\mathcal{X}) + \lambda \|\mathcal{C}\|_{kf}$$

as a function of the matrix of classifiers  $\mathcal{C}$ , and they offer several algorithms to minimize this objective function; the algorithm can be kernelized (Loeff *et al.* 2009). These correlated word predictors are close to, or at, the state of the art for word prediction (see Table 21.2). Results in this table support the idea that correlation is important only rather loosely; there is no clear advantage for methods that correlate word predictions. To some extent, this is an effect of the evaluation scheme. Image annotators very often omit good annotations (see the examples in Figure 21.10), and we do not have methods that can score word predictions that are accurate and useful but not predicted by annotators. Qualitative results do suggest that explicitly representing word correlation is helpful (Figure 21.10).

#### 21.4.4 Names and Faces

Rather than predict all tags from the whole image, we could cut the image into pieces (which might or might not overlap), then predict the tags from the pieces.



FIGURE 21.10: Word predictions for examples from the Corel 5K dataset, using the method of Loeff and Farhadi (2008). Blue words are correct predictions; red words are predictions that do not appear in the annotation of the image; and green words are annotations that were not predicted. Notice that the extra (red) words are strongly correlated to those that are correctly predicted, and are often good annotations. Image annotators often leave out the obvious—sky is present in the center image of the top row—and current scoring methods do not account for this phenomenon well. *his figure was originally published as Figure 5 of “Scene Discovery by Matrix Factorization,” by N. Loeff and A. Farhadi, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 5304, 2008 © Springer, 2008.*

This will increase the chance that that individual tag predictors could be trained independently. For example, it is unlikely that names in news captions are independent (in 2010, the name “Elin Nordegren” was very likely to co-occur with “Tiger Woods”). But this doesn’t mean we need to couple name predictors when we train them; instead, we could find each individual face in the image and then predict names independently for the individual faces. This assumes that the major reason that the names are correlated is that the faces tend to appear together in the images.

Linking faces in pictures with names and captions is a useful special case because news images are mainly about people, and captions very often give names. It is also a valuable example for developing methods to apply to other areas because it illustrates how correspondence between tags and image components can be exploited. Berg *et al.* (2004) describe a method to take a large dataset of captioned images, and produce a set of face images with the correct names attached. In their dataset, not every face in the picture is named in the caption, and not every name in the caption has a face in the picture (see the example in Figure 21.11). The first step is to detect names in captions using an open source named entity recognizer (Cunningham *et al.* 2002). The next is to detect (Mikolajczyk n.d.), rectify, and represent faces using standard appearance representations. We construct the feature vector so that Euclidean distance in feature vector space is a reasonable metric. We can now represent the captioned images as a set of data items that

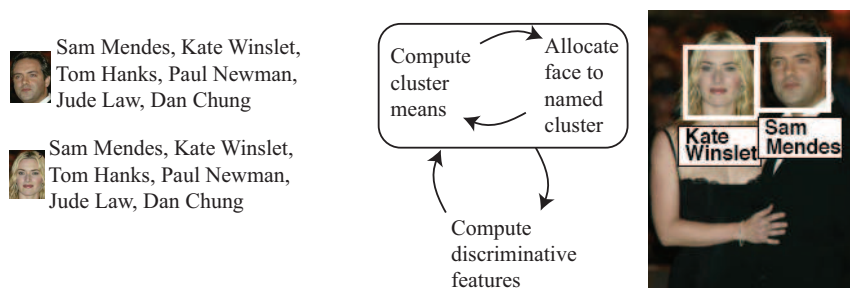


FIGURE 21.11: Berg *et al.* (2004) take a collection of captioned news images and link the faces in each image to names in the caption. They preprocess the images by detecting faces, then rectifying them and computing a feature representation of the rectified face. They detect proper names in captions using an open source named entity recognizer (Cunningham *et al.* 2002). The result is a set of data items that consist of (a) a face representation and (b) a list of names that could be associated with that face. *Part of this figure was originally published as Figure 2 of “Names and Faces in the News,” by T. Berg, A. Berg, J. Edwards, M. Maire, R. White, Y-W. Teh, E. Learned-Miller and D. Forsyth, Proc. IEEE CVPR 2004, © IEEE 2004.*

consist of a feature representation of a face, and a list of names that could go to that face (notice that some captioned images will produce several data items, as in Figure 21.11).

We must now associate names with faces (Figure 21.12). This can be seen as a form of k-means clustering. We represent each name with a cluster of possible appearance vectors, represented by the cluster mean. Assume we have an initial appearance model for each name; for each data item, we now allocate the face to the closest name in its list of possible names. Typically, these lists are relatively short, so we need only tell which item in a short list the face belongs to. We now re-estimate the appearance models, and repeat until labels do not change. At this point, we can re-estimate the feature space using the labels associated with the face images, then re-estimate the labeling. A natural variant is to allocate a face only when the closest name is closer than some threshold distance. The procedure can be started by allocating faces to the names in their list at random, or by exploiting cases where there is just one face and just one name. This strategy is crude, but works quite well, because it exploits two important features of the problem. First, on the whole, multiple instances of one individual’s face should look more like one another than like another individual’s face. Second, allocating one of a short list of names to a face is a lot easier than recognizing a face.

#### 21.4.5 Generating Tags with Segments

The most attractive feature of these names-and-faces models is that by reasoning about correspondence between pieces of image (in the models above, faces) and tags, we can learn models for tags independently. The fact that some tags co-occur strongly with others is caused by some pieces of image co-occurring strongly with others, so it doesn’t need to be accounted for by the model. There is now a

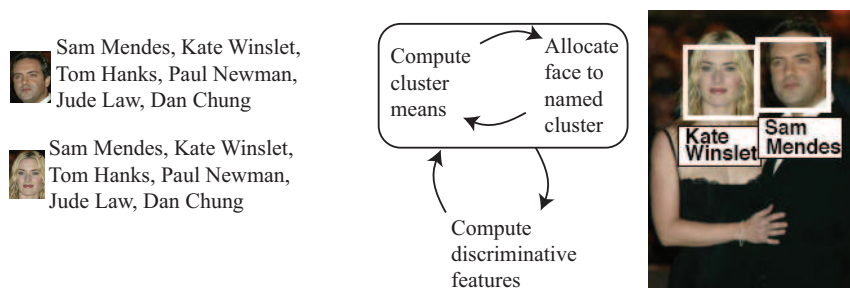


FIGURE 21.12: We associate faces with names by a process of repeated clustering. Each name in the dataset is associated with a cluster of appearance vectors, represented by a mean. Each face is then allocated to the closest name *in that face's list of names*. We now re-estimate the cluster means, and then reallocate faces to clusters. Once this process has converged, we can re-estimate the feature space using linear discriminants (Section 16.1.6), then repeat the labeling process. The result is a correspondence between image faces and names (**right**). *Part of this figure was originally published as Figure 2 of "Names and Faces in the News," by T. Berg, A. Berg, J. Edwards, M. Maire, R. White, Y-W. Teh, E. Learned-Miller and D. Forsyth, Proc. IEEE CVPR 2004, © IEEE 2004.*

huge variety of such models for tagging images with words. Generally, they can be divided into two classes: in one class, we reason explicitly about correspondence, as in the names and faces examples; in the other, the correspondence information is hidden implicitly in the model.

Explicit correspondence models follow the lines of the names and faces example. Duygulu *et al.* (2002) describe a model to which many other models have been compared. The image is segmented, and a feature descriptor incorporating size, location, color, and texture information is computed for each sufficiently large image segment. These descriptors are vector quantized using k-means. This means each tagged training image can be thought of as a bag that contains a set of vector quantized image descriptors and a set of words. There are many such bags, and we think of each bag as a set of samples from some process. This process generates image segments and then some image segments generate words probabilistically. This problem is analogous to one that occurs in the discipline of machine translation. Imagine we wish to build a dictionary giving the French word that corresponds to each English word. We could take the proceedings of the Canadian Parliament as a dataset. These proceedings conveniently appear in both French and English, and what a particular parliamentarian said in English (resp. French) is carefully translated into French (resp. English). This means we can easily build a rough paragraph-level correspondence. Corresponding pairs of paragraphs are bags of French words generated by (known) English words; what we don't know is which English word produced which French word. The vision problem is analogous if we replace English words with vector quantized image segments, and French words with words (Figure 21.13).

Brown *et al.* (1990) give a series of natural models and corresponding algorithms for this problem. The simplest model that applies is known as model 2 (there are five in total; the more complex models deal with the tendency of some



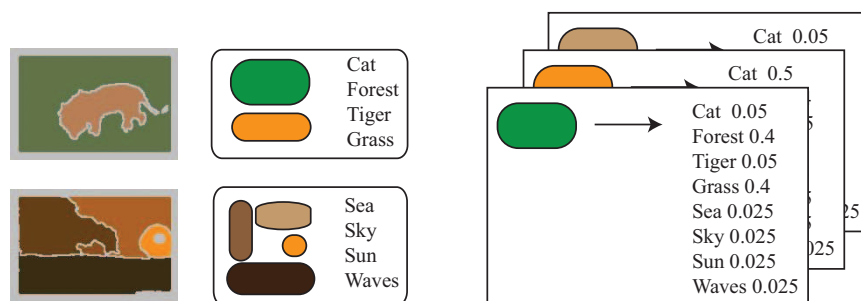


FIGURE 21.13: Duygulu *et al.* (2002) generate annotations for images by segmenting the image (**left**) and then allowing each sufficiently large segment to generate a tag. Segments generate tags using a lexicon (**right**), a table of conditional probabilities for each tag given a segment. They learn this lexicon by abstracting each annotated image as a bag of segments and tags (**center**). If we had a large number of such bags, and knew which tag corresponded to which segment, then building the lexicon just involves counting; similarly, if we knew the lexicon, we could estimate which tag corresponded to which segment in each bag. This suggests using an EM method to estimate the lexicon. *This figure was originally published as Figure 1 of “Object Recognition as Machine Translation: Learning a lexicon for a fixed image vocabulary,” by P. Duygulu, K. Barnard, N. deFreitas, and D. Forsyth, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 2353, 2002 © Springer, 2002.*

languages to be wordier than others, or to have specific word orders, and do not apply). We assume that each word is generated by a single blob, and associate a (hidden) correspondence variable with each bag. We can then estimate  $p(w|b)$ , the conditional probability that a word type is generated by a blob type (analogous to a dictionary), using EM.

Once we have a lexicon, we can tag each sufficiently large region with its highest probability word; or do so, but refuse to tag regions where the predicted word has too low a probability; or tag only the  $k$  regions that predict words with the highest probability; or do so, but check probabilities against a threshold. This method as it stands is now obsolete as an image tagger, but is widely used as a comparison point because it is natural, quite easily beaten, and associated with an easily available dataset (the Corel5K dataset described in Section 21.5.1).

The cost of reasoning explicitly about correspondence between individual regions and individual words is that such models ignore larger image context. An alternative is to build a form of generative model that explains the bag of segments and words without reasoning about which segment produced which word. An example of such an implicit correspondence model is the cross-media relevance model of Jeon *et al.* (2003). We approximate words as conditionally independent given an image, which means we need to build a model of the probability of a single word conditioned on an image,  $P(w|I)$ . We approximate this as  $P(w|b_1, \dots, b_n)$ , and must now model this probability. We will do so by modelling the joint probability  $P(w, b_1, \dots, b_n)$ . We assume a stochastic relationship between the blobs and the image; and we assume that, conditioned on the image, the blobs and the words are

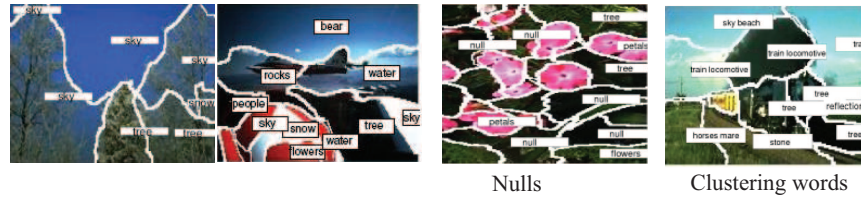


FIGURE 21.14: The basic correspondence method we described in the text can produce reasonable results for some image tags (**left**), but tends to perform better with tags that describe “stuff” rather than tags that describe “things” (**center left**). Some of this is because the method has very weak shape representations, and cannot fuse regions. However, it is extremely flexible. Improved word predictions can be obtained by refusing to predict words for regions where the conditional probability of the most likely word is too low (**center right**, “null” predictions), and by fusing words that are predicted by similar image regions (**right**, “train” and “locomotive”). *This figure was originally published as Figures 8, 10, and 11 of “Object Recognition as Machine Translation: Learning a lexicon for a fixed image vocabulary,” by P. Duygulu, K. Barnard, N. deFreitas, and D. Forsyth, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 2353, 2002 © Springer, 2002.*

independent. If we write  $\mathcal{T}$  for the training set, we have

$$\begin{aligned} P(w, b_1, \dots, b_n) &= \sum_{j \in \mathcal{T}} P(J) P(w, b_1, \dots, b_n | J) \\ &= \sum_{j \in \mathcal{T}} P(J) P(w | J) \prod_{j=1}^{\#w} P(b_j | J), \end{aligned}$$

and these component probabilities can be estimated by counting and smoothing. Jeon *et al.* assume that  $P(J)$  is uniform over the training images. Now write  $c(w, J)$  for the number of times the word  $w$  appears as a tag for image  $J$  and  $c_w(J)$  for the total number of words tagging  $J$ . Then we could estimate

$$P(w | J) = (1 - \alpha) \frac{c(w, J)}{c_w(J)} + \alpha \frac{c(w, \mathcal{T})}{c_w(\mathcal{T})}$$

(where we have smoothed the estimate so that all words have some small probability of being attached to  $J$ ). Notice that this is a form of non-parametric topic model. Words and blobs are not independent in the model as a result of the sum over training images, but there is no explicit tying of words to blobs. This model is simple and produces quite good results. As a model, it is now obsolete, but it is a good example of a very large family of models.

## 21.5 THE STATE OF THE ART OF WORD PREDICTION

Word prediction is now an established problem that operates somewhat independently from object recognition. It is quite straightforward to start research because good standard datasets are available (Section 21.5.1), and methods are quite easy to compare quantitatively because there is at least a rough consensus on appropriate evaluation methodologies (Section 21.5.2). Finally, there are numerous good

open questions (Section 21.5.3), which are worth engaging with because the search application is so compelling.

### 21.5.1 Resources

Most of the code that would be used for systems described in this chapter is feature code (Section 16.3.1) or classifier code (Section 15.3.3). Code for approximate nearest neighbors that can tell whether k-d trees or locality sensitive hashing works better on a particular dataset, and can tune the chosen method, is published by Marius Muja at <http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN>.

The Corel5K dataset contains 5,000 images collected from a larger set of stock photos, split into 4,500 training and 500 test examples. Each image has 3.5 keywords on average, from a dictionary of 260 words that appear in both the training and the test set. The dataset was popularized by Duygulu *et al.* (2002). As of the time of writing, an archive of features and tags for this dataset can be found at <http://lear.inrialpes.fr/people/guillaumin/data.php>.

The IAPRTC-12 dataset contains 20,000 images, accompanied by free text captions. Tags are then extracted from the text by various parsing methods. As of the time of writing, the dataset can be obtained from <http://imageclef.org/photodata>. Various groups publish the features and tags they use for this dataset. See <http://lear.inrialpes.fr/people/guillaumin/data.php>, or <http://www.cis.upenn.edu/~makadia/annotation/>.

The ESP dataset consists of 21,844 images collected using a collaborative image labeling task (von Ahn and Dabbish 2004); two players assign labels to an image without communicating, and labels they agree on are accepted. Images can be reassigned, and then only new labels are accepted (see <http://www.espgame.org>). This means that the pool of labels for an image grows, with easy labels being assigned first.

MirFlickr is a dataset of a million Flickr images, licensed under creative commons and released with concrete visual tags associated (see <http://press.liacs.nl/mirflickr/>).

### 21.5.2 Comparing Methods

Generally, methods can be compared using recall, precision, and  $F_1$  measure on appropriate datasets. Table 21.2 gives a comparison of methods applied to Corel5K using these measures. The performance statistics are taken from the literature. Some variations between experiments mean that comparisons are rough and ready: CorrLDA predicts a smaller dictionary than the other methods; PicSOM predicts only five annotations; and the  $F_1$  measure for Submodular is taken by eye from the graph of figure 3 in (Kang *et al.* 2006), in the method's most favorable configuration. Table 21.2 suggests that (a) performance has improved over time, though the nearest neighbor method of Section 21.4.2 is simultaneously the simplest and the best performing method; and (b) that accounting for correlations between labels helps, but isn't decisive (for example, neither Submodular nor CorrPred decisively beats JEC). This suggests there is still much to be learned about the image annotation problem.

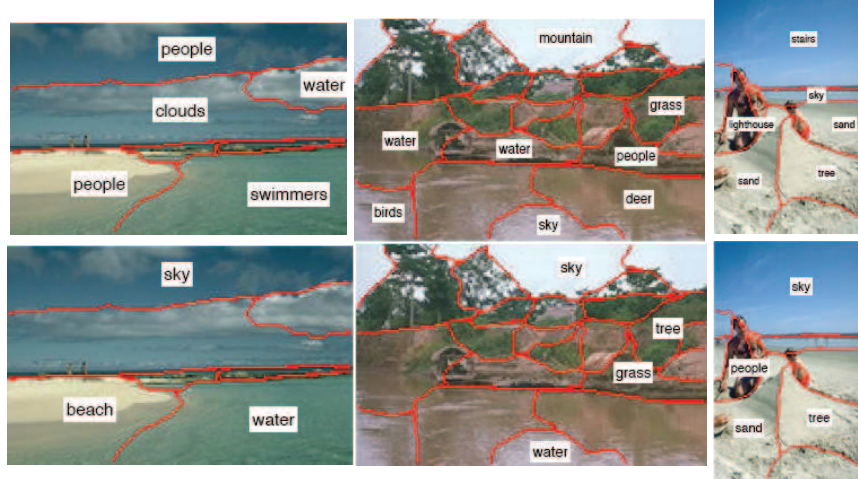
Method	P	R	$F_1$	Ref
Co-occ	0.03	0.02	0.02	(Mori <i>et al.</i> 1999)
Trans	0.06	0.04	0.05	(Duygulu <i>et al.</i> 2002)
CMRM	0.10	0.09	0.10	(Jeon <i>et al.</i> 2003)
TSIS	0.10	0.09	0.10	(Celebi 30 Nov. - 1 Dec. 2005)
MaxEnt	0.09	0.12	0.10	(Jeon and Manmatha 2004)
CRM	0.16	0.19	0.17	(Lavrenko <i>et al.</i> 2003)
CT-3×3	0.18	0.21	0.19	(Yavlinsky <i>et al.</i> 2005)
CRM-rect	0.22	0.23	0.23	(Feng <i>et al.</i> 2004)
InfNet	0.17	0.24	0.23	(Metzler and Manmatha 2004)
MBRM	0.24	0.25	0.25	(Feng <i>et al.</i> 2004)
MixHier	0.23	0.29	0.26	(Carneiro and Vasconcelos 2005)
CorrLDA <sup>1</sup>	0.06	0.09	0.072	(Blei and Jordan 2002)
JEC	0.27	0.32	0.29	(Makadia <i>et al.</i> 2010)
JEC <sup>2</sup>	0.32	0.40	0.36	(Makadia <i>et al.</i> 2010)
Submodular	-	-	0.26	(Kang <i>et al.</i> 2006)
CorrPred	0.27	0.27	0.27	(Loeff and Farhadi 2008)
CorrPredKernel	0.29	0.29	0.29	(Loeff and Farhadi 2008)
PicSOM <sup>3</sup>	0.35	0.35	0.35	(Viitaniemi and Laaksonen 2007)

TABLE 21.2: Comparison of the performance of various word annotation prediction methods by precision, recall, and F1-measure, on the Corel 5K dataset. The methods described in the text are: *Trans*, which is the translation model of Section 21.4.5; *CMRM*, which is the cross-media relevance model of Section 21.4.5; *CorrPred*, which is the correlated classifier method of Section 21.4.3; *JEC*, which is the nearest neighbor method of Section 21.4.2; and *Submodular*, which is the submodular optimization method of Section 21.4.2. Other performance figures are given for information, and details of the models appear in the papers cited.

### 21.5.3 Open Problems

One important open problem is *selection*. Assume we wish to produce a textual representation of an image—what should it contain? It is unlikely that a list of all objects present is useful or helpful. For most pictures, such a list would be much too long and dominated by extraneous detail; preparing it would involve dealing with issues like whether the nut used to hold the chairleg to the chair is a separate object, or just a part of the chair. Several phenomena seem to affect selection; some objects are interesting by their nature and tend to get mentioned if they occur in an image. Spain and Perona (2008) give a probabilistic model that can often predict such mentions. Other objects are interesting because of where they occur in the image, or how big they are in the image. Yet other objects are interesting because they have unusual properties (say, a glass cat or a car without wheels), and identifying this remains difficult. Some objects are depicted in unusual circumstances (for example, a car that is upsidedown). This means that context cues might help tell what is worth mentioning. Choi *et al.* (2010) show a variety of contextual cues that can be computed and identify an object as unusual for context.

## Without spatial relations



## With spatial relations

FIGURE 21.15: Gupta and Davis (2008) show that image labeling can be improved by representing spatial relations between regions, and taking these relations into account when labeling. The top row shows labelings predicted using the method of (Duygulu *et al.* 2002), which considers only individual regions when labeling; the bottom row shows predictions made by their method. Notice that, for example, the “lighthouse” and the “steps” in the image on the right are overruled by the other patches and given better labels by the context. *This figure was originally published as Figure 7 of “Beyond nouns; exploiting prepositions and comparative adjectives for learning visual classifiers,” by A. Gupta and L. Davis, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 5302, 2008 © Springer, 2002.*

**Modifiers**, such as adjectives or adjectival phrases, present interesting possibilities to advance learning. Yanai and Barnard (2005) demonstrated that it was possible to learn local image features corresponding to color words (e.g., “pink”) without knowing what parts of the image the annotation referred to (Yanai and Barnard 2005). This raises the interesting possibility that possessing phrases can help learning: for example, it is easier to learn from “pink cadillac” than from “cadillac,” because the “pink” helps tell where the “cadillac” is in the image. Small improvements have been demonstrated using this approach (Wang and Forsyth 2009). A discipline in linguistics, known as *pragmatics*, studies the things that people choose to say; one guideline is that people mention things that are unusual or important. This suggests that, for example, there is no particular value in mentioning that a sheep is white or a meadow is green. This means that we face two problems: first, we must determine what modifiers apply to a particular object, and second, we must determine whether that modifier is worth mentioning. Farhadi *et al.* (2009a) have demonstrated a method that can identify instances of objects that either have unusual attributes, or lack usual ones. This method may be capable of predicting what modifiers are worth mentioning.

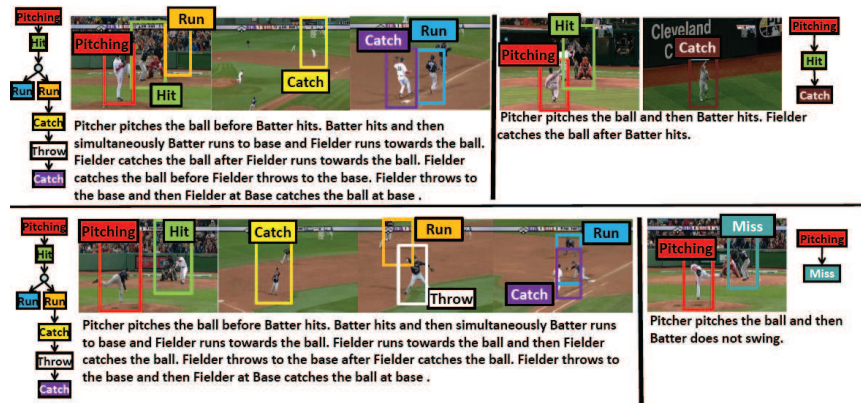


FIGURE 21.16: In structured video, we can predict narratives by exploiting the structure of what can happen. This figure shows examples from the work of Gupta *et al.* (2009), who show that by searching possible actors and templates for sports videos, one can come up with matches that are accurate enough to build a reasonable narrative from templates. *This figure was originally published as Figure 7 of “Understanding Videos, Constructing Plots Learning a Visually Grounded Storyline Model from Annotated Videos,” by A. Gupta, P. Srinivasan, J. Shi, and L.S. Davis, Proc. IEEE CVPR 2009, © IEEE 2009.*

Another way to obtain richer descriptions of images is to use spatial **relations** between objects. Heitz and Koller (2008) improve the performance of object detectors by identifying patches of *stuff*—materials such as grass, tarmac, sky and so on, where the shape of the region has little to offer in identifying what it is—that lie nearby. They train a probabilistic graphical model to enhance the detector response when appropriate materials lie in appropriate places (and weaken the response when they don’t); the result is a small improvement in detector performance. Gupta and Davis (2008) use images labeled with relational phrases (e.g., “bear in water”) to learn to label regions with noun tags together with models of spatial relations. Relational cues could improve learning by disambiguating correspondence quite strongly; for example, if one has a good model of “grass” and of the spatial relation “on,” then “sheep on the grass” offers strong cues as to which region is “sheep.” Experiments suggest that both of these effects are significant and helpful; the paper shows significant improvements in region labeling (Figure 21.15).

The natural goal of all this generalization is to produce **sentences** from images. Even short sentences can represent a great deal of information in a compact form. To produce a sentence, we would need to select what is worth mentioning; we would need to decide what was happening, what was doing it, and to what it was being done; and we would need to know what modifiers to attach where. In some kinds of video (for example, of a sport), the narrative structure of what is likely to happen is quite stylized, and so quite good sentences can be produced (Figure 21.16). Gupta *et al.* (2009) have shown this means that we can search for sets of actors that fit a template of an action, then report that action in quite a rich sentence form. Yao *et al.* (2010) have been able to link image parsing strategies to text generation strategies to generate informative sentences about video. For



	(transportation, move, track)	A man stands next to a train on a cloudy day
	(bike, ride, track)	A backpacker stands beside a green train
	(transportation, move, road)	This is a picture of a man standing next to a green train
	(pet, sleep, ground)	There are two men standing on a rocky beach, smiling at the camera.
	(bike, ride, road)	This is a person laying down in the grass next to their bike in front of a strange white building.
	(display, place, table)	This is a lot of technology.
	(furniture, place, furniture)	Somebody's screensaver of a pumpkin
	(furniture, place, furniture)	A black laptop is connected to a black Dell monitor
	(bottle, place, table)	This is a dual monitor setup
	(furniture, place, home)	Old school Computer monitor with way to many stickers on it

FIGURE 21.17: Farhadi *et al.* (2010b) link sentences to pictures by first computing an affinity with an intermediate representation, then using this to compute a score for a sentence-image pair; the sentence that produces the best score is the annotation. On the **left**, two example images; in the **center**, the top five intermediate representations, given as triples of (actor, scene, action); on the **right**, the top five sentences for each image. Notice how sentence details tend to be inaccurate, but the general thrust of the sentence is often right. *This figure was originally published as Figure 3 of “Every picture tells a story: Generating sentences from images,” by A. Farhadi, M. Hejrati, M.A. Sadeghi, P. Young, C. Rastchian, J. Hockenmaier, and D. Forsyth, Proc. European Conference on Computer Vision. Springer Lecture Notes in Computer Science, Volume 6314, 2010 © Springer, 2009.*

static images, the problem remains very difficult; Farhadi *et al.* (2010b) describe one method to link static images to sentences, using an intermediate representation to manage difficulties created by the fact that we have no detector for most of the words encountered (Figure 21.17).

## 21.6 NOTES

There are many datasets of images with associated words. Examples include: collections of museum material (Barnard *et al.* 2001b); the Corel collection of images, described in (Barnard and Forsyth 2001, Duygulu *et al.* 2002, Chen and Wang 2004), and numerous other papers; any video with sound or closed captioning (Sato and Kanade 1997, Sato *et al.* 1999, Wang *et al.* 2000); images collected from the Web with their enclosing web pages (Berg and Forsyth 2006); or captioned news images (Berg *et al.* 2004). It is a remarkable fact that, in these collections, pictures and their associated annotations are complementary. The literature is very extensive, and we can mention only the most relevant papers here. For a more complete review, we refer readers to (Datta *et al.* 2005), which has 120 references. There are three natural activities: One might wish to cluster images; to search for images using keywords; or to attach keywords to new images. Typically, models intended for one purpose can produce results for others.

**Search:** Belongie *et al.* (1998b) demonstrate examples of joint image-keyword searches. Joshi *et al.* (2004) show that one can identify pictures that illustrate a story by searching annotated images for those with relevant keywords, then ranking the pool of images based on similarity of appearance. **Clustering:** Barnard and Forsyth (2001) cluster Corel images and their keywords jointly to produce a browsable representation; the clustering method is due to Hofmann and Puzicha (1998). Barnard *et al.* (2001b) show that this form of clustering can produce a useful, browsable representation of a large collection of annotated art in digital