Object Recognition as Machine Translation: Learning a Lexicon for a Fixed Image Vocabulary

P. Duygulu[1], K. Barnard[1], J.F.G. de Freitas[2], and D.A. Forsyth[1]

[1] Computer Science Division, U.C. Berkeley, Berkeley, CA 94720
[2] Department of Computer Science, University of British Columbia,Vancouver
{duygulu, kobus, daf}@cs.berkeley.edu, nando@cs.ubc.ca

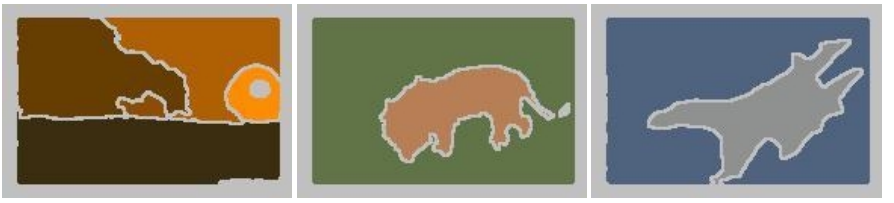
Abstract. We describe a model of object recognition as machine translation. In this model, recognition is a process of annotating image regions with words. Firstly, images are segmented into regions, which are classified into region types using a variety of features. A mapping between region types and keywords supplied with the images, is then learned, using a method based around EM. This process is analogous with learning a lexicon from an aligned bitext. For the implementation we describe, these words are nouns taken from a large vocabulary. On a large test set, the method can predict numerous words with high accuracy. Simple methods identify words that cannot be predicted well. We show how to cluster words that individually are difficult to predict into clusters that can be predicted well — for example, we cannot predict the distinction between `train` and `locomotive` using the current set of features, but we can predict the underlying concept. The method is trained on a substantial collection of images. Extensive experimental results illustrate the strengths and weaknesses of the approach.

Keywords: Object recognition, correspondence, EM algorithm.

1 Introduction

There are three major current types of theory of object recognition. One reasons either in terms of geometric correspondence and pose consistency; in terms of template matching via classifiers; or by correspondence search to establish the presence of suggestive relations between templates. A detailed review of these strategies appears in [4]. These types of theory are at the wrong scale to address core issues: in particular, **what counts as an object?** (usually addressed by choosing by hand objects that can be recognised using the strategy propounded); **which objects are easy to recognise and which are hard?** (not usually addressed explicitly); and **which objects are indistinguishable using our features?** (current theories typically cannot predict the equivalence relation imposed on objects by the use of a particular set of features). This paper describes a model of recognition that offers some purchase on each of the questions above, and demonstrates systems built with this model.

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2353, pp. 97–112, 2002.



sea sky sun waves | cat forest grass tiger | jet plane sky

Fig. 1. Examples from the Corel data set. We have associated keywords and segments for each image, but we don't know which word corresponds to which segment. The number of words and segments can be different; even when they are same, we may have more than one segment for a single word, or more than one word for a single blob. We try to align the words and segments, so that for example an orange stripy blob will correspond to the word tiger.

1.1 Annotated Images and Auto-Annotation

There are a wide variety of datasets that consist of very large numbers of annotated images. Examples include the Corel dataset (see figure 1), most museum image collections (e.g. `http://www.thinker.org/fam/thinker.html`), the web archive (`http://www.archive.org`), and most collections of news photographs on the web (which come with captions). Typically, these annotations refer to the content of the annotated image, more or less specifically and more or less comprehensively. For example, the Corel annotations describe specific image content, but not all of it; museum collections are often annotated with some specific material — the artist, date of acquisition, etc. — but often contain some rather abstract material as well.

There exist some methods that cluster image representations and text to produce a representation of a joint distribution linking images and words [1, 2]. This work could predict words for a given image by computing words that had a high posterior probability given the image. This process, referred to as **auto-annotation** in those papers, is useful in itself (it is common to index images using manual annotations [7,12]; if one could predict these annotations, one could save considerable work). However, in this form auto-annotation does not tell us *which* image structure gave rise to *which* word, and so it is not really recognition. In [8], Mori *et.al.* proposed a method for annotating image grids using cooccurences. In [9,10], Maron *et al.* study automatic annotation of images, but work one word at a time, and offer no method of finding the correspondence between words and regions. This paper shows that it is possible to learn which region gave rise to which word.

Recognition as translation. One should see this process as analogous to machine translation. We have a representation of one form (image regions; French) and wish to turn it into another form (words; English). In particular, our models will act as *lexicons*, devices that predict one representation (words; English),

given another representation (image regions; French). Learning a lexicon from data is a standard problem in machine translation literature (a good guide is Melamed's thesis [11]; see also [5,6]). Typically, lexicons are learned from a form of dataset known as an **aligned bitext** — a text in two languages, where rough correspondence, perhaps at the paragraph or sentence level, is known. The problem of lexicon acquisition involves determining precise correspondences between words of different languages. Datasets consisting of annotated images are aligned bitexts — we have an image, consisting of regions, and a set of text. While we know the text goes with the image, we don't know which word goes with which region. As the rest of this paper shows, we can learn this correspondence using a variant of EM.

This view — of recognition as translation — renders several important object recognition problems amenable to attack. In this model, we can attack: **what counts as an object?** by saying that all words (or all nouns, etc.) count as objects; **which objects are easy to recognise?** by saying that words that can be reliably attached to image regions are easy to recognise and those that cannot, are not; and **which objects are indistinguishable using our features?** by finding words that are predicted with about the same posterior probability given any image group — such objects are indistinguishable given the current feature set.

2 Using EM to Learn a Lexicon

We will segment images into regions and then learn to predict words using regions. Each region will be described by some set of features. In machine translation, a lexicon links discrete objects (words in one language) to discrete objects (words in the other language). However, the features naturally associated with image regions do not occupy a discrete space. The simplest solution to this problem is to use k-means to vector quantize the image region representation. We refer to the label associated with a region by this process as a "blob."

In the current work, we use all keywords associated with each image. If we need to refer to the abstract model of a word (resp. blob) — rather than an instance — and the context doesn't make the reference obvious, we will use the term "word token" (resp. blob token). The problem is to use the training data set to construct a probability table linking blob tokens with word tokens. This table is the conditional probability of a word token given a blob token.

The difficulty in learning this table is that the data set does not provide explicit correspondence — we don't know which region is the `train`. This suggests the following iterative strategy: firstly, use an estimate of the probability table to predict correspondences; now use the correspondences to refine the estimate of the probability table. This, in essence, is what EM does.

We can then annotate the images by first classifying the segments to find the corresponding blobs, and then finding the corresponding word for each blob by choosing the word with the highest probability.

N : Number of images.
M_T : Number of words. L_T : Number of blobs.
M_n : Number of words in the n-th image. L_n : Number of blobs in the n-th image.
w_n : words in the n-th image, b_n : blobs in the n-th image,
$w_n = (w_{n1}, \ldots, w_{nj}, \ldots, w_{nM_n})$ $b_n = (b_{n1}, \ldots, b_{ni}, \ldots, b_{nL_n})$
$w^\star$: A particular word. $b^\star$: A particular blob.

a_n : Assignment $a_n = \{a_{n1}, \ldots, a_{nM_n}\}$, such that $a_{nj} = i$ if b_{ni} translates to w_{nj}.
$t(w|b)$: Probability of obtaining instance of word w given instance of blob b;
$p(a_{nj} = i)$: Assignment probabilities;
θ : Set of model parameters $\theta = (p(a_{nj} = i), t(w_{nj}|b_{ni}))$.
$p(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})})$: Indicators;

Fig. 2. Notation

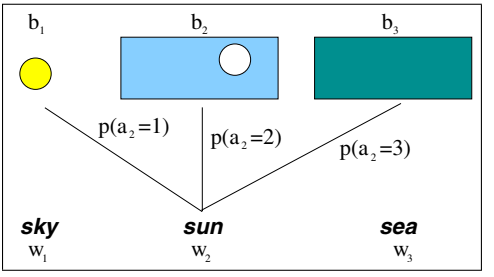


Fig. 3. Example : Each word is predicted with some probability by each blob, meaning that we have a mixture model for each word. The association probabilities provide the correspondences (assignments) between each word and the various image segments. Assume that these assignments are known; then computing the mixture model is a matter of counting. Similarly, assume that the association probabilities are known; then the correspondences can be predicted. This means that EM is an appropriate estimation algorithm.

2.1 EM Algorithm for Finding the Correspondence between Blobs and Words

We use the notation of figure 2. When translating blobs to words, we need to estimate the probability $p(a_{nj} = i)$ that in image n, a particular blob b_i is associated with a specific word w_j. We do this for each image as shown in Figure 3.

We use model 2 of Brown *et.al.* [3], which requires that we sum over all the possible assignments of words to blobs.

$$p(w|b) = \prod_{n=1}^{N} \prod_{j=1}^{M_n} \sum_{i=1}^{L_n} p(a_{nj} = i) t(w = w_{nj}|b = b_{ni}) \tag{1}$$

Maximising this likelihood is difficult because of the sum inside the products; the sum represents marginalisation over all possible correspondences. The

problem can be treated as a missing data problem, where the missing data is the correspondence. This leads to the EM formulation. (From now on, we write $t(w = w_{nj}|b = b_{ni})$ as $t(w_{nj}|b_{ni})$.)

Initialise

E step

1. For each $n = 1, \ldots, N$, $j = 1, \ldots, M_n$ and $i = 1, \ldots, L_n$, compute
$$\widetilde{p}(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})}) = p(a_{nj} = i)t(w_{nj}|b_{ni}) \quad (2)$$
2. Normalise $\widetilde{p}(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})})$ for each image n and word j
$$p(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})}) = \frac{\widetilde{p}(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})})}{\sum_{i=1}^{L_n} p(a_{nj} = i)t(w_{nj}|b_{ni})} \quad (3)$$

M step

1. Compute the mixing probabilities for each j and image of the same size (*e.g.* $L(n) = l$ and $M(n) = m$)
$$p(a_{nj} = i) = \frac{1}{N_{l,m}} \sum_{n:L(n)=l,M(n)=m}^{N} p(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})}) \quad (4)$$
where $N_{l,m}$ is the number of images of the same length.
2. For each different pair $(b^\star, w^\star)$ appearing together in at least one of the images, compute
$$\widetilde{t}(w_{nj} = w^\star|b_{ni} = b^\star) = \sum_{n=1}^{N}\sum_{j=1}^{M_n}\sum_{i=1}^{L_n} p(a_{nj} = i|w_{nj}, b_{ni}, \theta^{(\text{old})})\delta_{(w^\star,b^\star)}(w_{nj}, b_{ni}) \quad (5)$$
where $\delta_{(w^\star,b^\star)}(w_{nj}, b_{ni})$ is 1 if $b^\star$ and $w^\star$ appear in image and 0 otherwise.
3. Normalise $\widetilde{t}(w_{nj} = w^\star|b_{ni} = b^\star)$ to obtain $t(w_{nj} = w^\star|b_{ni} = b^\star)$.

Fig. 4. EM (Expectation Maximization) algorithm

2.2 Maximum Likelihood Estimation with EM

We want to find the maximum likelihood parameters

$$\theta^{\text{ML}} = \arg\max_{\theta} \; p(w|b, \theta) = \arg\max_{\theta} \; \sum_{a} p(a, w|b, \theta). \quad (6)$$

We can carry out this optimisation using an EM algorithm, which iterates between the following two steps.

1. E step: Compute the expected value of the complete log-likelihood function with respect to the distribution of the assignment variables $Q^{\text{ML}} = \mathbf{E}_{p(a|w,b,\theta^{(\text{old})})}[\log p(a, w|b, \theta)]$, where $\theta^{(\text{old})}$ refers to the value of the parameters at the previous time step.

2. M step: Find the new maximum $\theta^{(\text{new})} = \arg\max_{\theta} Q^{\text{ML}}$.

In our case, the Q^{ML} function is given by

$$Q^{\text{ML}} = \sum_{n=1}^{N} \sum_{j=1}^{M_n} \sum_{i=1}^{L_n} p(a_{nj} = i | w_{nj}, b_{ni}, \theta^{(\text{old})}) \log \left[p(a_{nj} = i) t(w_{nj} | b_{ni}) \right]. \quad (7)$$

We need to maximise Q^{ML} subject to the constraints $\sum_i p(a_{nj} = i) = 1$ for all words j in all images n with equal number of words and blobs, and $\sum_{w^\star} t(w^\star | b^\star) = 1$ for any word $w^\star$ and each blob $b^\star$. This can be accomplished by introducing the Lagrangian

$$\mathcal{L} = Q^{\text{ML}} + \sum_{\alpha_{n,l,m}} \alpha_{n,l,m} \left(1 - \sum_{i=1}^{L_n} p(a_{nj} = i) \right) + \sum_{b^\star} \beta_{b^\star} \left(1 - \sum_{w^\star} t(w^\star | b^\star) \right) \quad (8)$$

and, computing derivatives with respect to the multipliers (α, β) and the parameters $(p(a_{nj} = i), t(w^\star | b^\star))$. Note that there is one $\alpha_{n,l,m}$ multiplier for each image n with $L(n) = l$ blobs and $M(n) = m$ words. That is, we need to take into account all possible different lengths for normalisation purposes. The end result is the three equations that form the core of the EM algorithm shown in Figure 4.

3 Applying and Refining the Lexicon

After obtaining the probability table, we can annotate image regions in any test image. We do this by assigning words to some or all regions. We first determine the blob corresponding to each region by vector quantisation. We now choose the word with the highest probability given the blob and annotate the region with this word. There are several important variants available.

3.1 Controlling the Vocabulary by Refusing to Predict

The process of learning the table prunes the vocabulary to some extent, because some words may not be the word predicted with highest probability for any blob. However, even for words that remain in the vocabulary, we don't expect all predictions to be good. In particular, some blobs may not predict any word with high probability, perhaps because they are too small to have a distinct identity. It is natural to establish a threshold and require that

$$p(word|blob) > threshold$$

before predicting the word. This is equivalent to assigning a null word to any blob whose best predicted word lies below this threshold. The threshold itself can be chosen using performance measures on the training data, as in section 4. This process of refusing to predict prunes the vocabulary further, because some

words may never be predicted with sufficient probability. In turn, this suggests that once a threshold has been determined, a new lexicon should be fitted using only the reduced vocabulary. In practice, this is advantageous (section 4), probably because reassigning probability "stolen" by words that cannot be predicted improves correspondence estimates and so the quality of the lexicon.

3.2 Clustering Indistinguishable Words

Generally, we do not expect to obtain datasets with a vocabulary that is totally suitable for our purposes. Some words may be visually indistinguishable, like `cat` and `tiger`, or `train` and `locomotive`. (some examples are shown in figure 11). Other words may be visually distinguishable *in principle*, but not using our features, for example `eagle` and `jet`, both of which occur as large dark regions of roughly the same shape in aerial views. Finally, some words may never appear apart sufficiently often to allow the correspondence to be disentangled in detail. This can occur because one word is a modifier — for example, in our data set, `polar` reliably predicts `bear` — or because of some relation between the concepts — for example, in our data set, either `mare` or `foals` almost quite reliably predicts `horses` — but in either case, there is no prospect of learning the correspondence properly. There are some methods for learning to form compounds like `polar bear` [11], but we have not yet experimented with them.

All this means that there are distinctions between words we should not attempt to draw based on the particular blob data used. This suggests clustering the words which are very similar. Each word is replaced with its cluster label; prediction performance should (and does, section 4) improve.

In order to cluster the words, we obtain a similarity matrix giving similarity scores for words. To compare two words, we use the symmetrised Kullback-Leibler (KL) divergence between the conditional probability of blobs, given the words. This implies that two words will be similar if they generate similar image blobs at similar frequencies. We then apply normalised cuts on the similarity matrix to obtain the clusters [13]. At each stage, we set the number of clusters to 75% of the current vocabulary.

4 Experimental Results

We train using 4500 Corel images. There are 371 words in total in the vocabulary and each image has 4-5 keywords. Images are segmented using Normalized Cuts [13]. Only regions larger than a threshold are used, and there are typically 5-10 regions for each image. Regions are then clustered into 500 blobs using k-means. We use 33 features for each region (including region color and standard deviation, region average orientation energy (12 filters), region size, location, convexity, first moment, and ratio of region area to boundary length squared). We emphasize that we chose a set of features and stuck with it through the experimental procedure, as we wish to study mechanisms of recognition rather than specific feature sets.

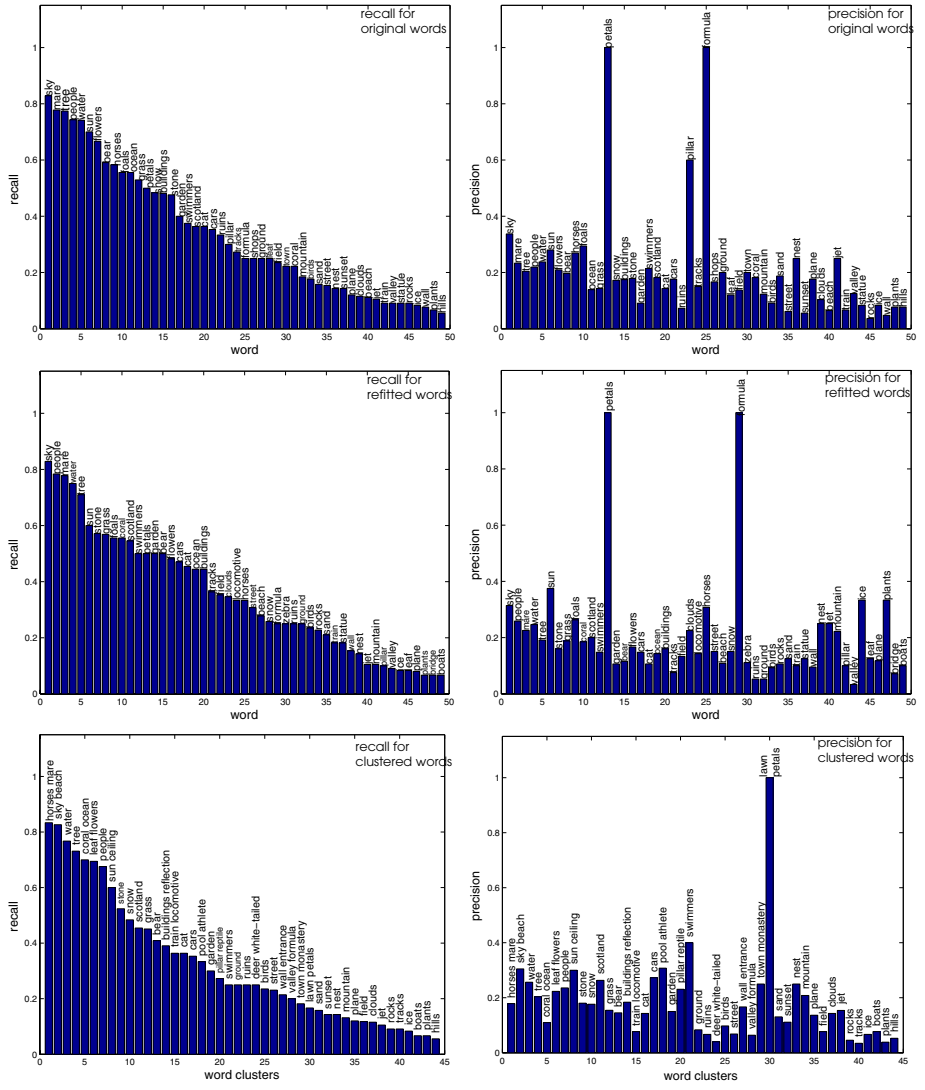


Fig. 5. For 500 test images, **left:** recall values in sorted order, **right:** corresponding precision values. **top:** original words, **middle:** refitted data, **bottom:** word clusters. The axes are same in all of the figures. We have some very good words with high recall, and some words with low recall. However, the precision values are not so much different from each other. Although they don't have high recall values, some words have very high precision values, which means that they are not predicted frequently, but when we do predict them we can predict them correctly. When we restrict ourselves only to the reduced vocabulary and run the EM algorithm again, the number of words that we can predict well doesn't change much, but the values increase slightly. Most of the clusters group indistinguishable words into one word, so clustering slightly increases the recall for some clusters (like `horse-mare`, `coral-ocean`)

4.1 Evaluating Annotation

Annotation is relatively easy to evaluate, because the images come from an annotated set. We use 500 images from a held-out test set to evaluate annotation performance. A variety of metrics are possible; the receiver operating curve is not particularly helpful, because there are so many words. Instead, we evaluate the performance of a putative retrieval system using automatic annotation. The class confusion matrix is also not helpful in our case, because the number of classes is 371, and we have a very sparse matrix.

Evaluation method: Each image in the test set is automatically annotated, by taking every region larger than the threshold, quantizing the region to a blob, and using the lexicon to determine the most likely word given that blob; if the probability of the word given the blob is greater than the relevant threshold, then the image is annotated with that word. We now consider retrieving an image from the test set using keywords from the vocabulary and the automatically established annotations. We score relevance by looking at the actual annotations, and plot recall and precision.

Base results: Only 80 words from the 371 word vocabulary can be predicted (others do not have the maximum value of the probability for any blob). We set the minimum probability threshold to zero, so that every blob predicts a word. As figure 5 shows, we have some words with very high recall values, and we have some words with low recall. The precision values shown in the figure don't vary much on the whole, though some words have very high precision. For these words, the recall is not high, suggesting that we can also predict some low frequency words very well. Table 1 shows recall and precision values for some good words for which recall is higher than 0.4 and precision is higher than 0.15.

The effect of retraining: Since we can predict only 80 words, we can reduce our vocabulary only to those words, and run EM algorithm again. As figure 5 shows, the results for the refitted words are very similar to the original ones. However, we can predict some words with higher recall and higher precision. Table 2 shows the recall and precision values for the selected good words after retraining. The number of good words are more than the original ones (compare with table 1), since the words have higher probabilities.

The effect of the null probability: We compare the recall and precision values for test and training data on some chosen words. As can be seen in figure 6, the results are very similar for both test and training data. We also experiment with the effect of null threshold by changing it between 0 and 0.4. By increasing the null threshold the recall decreases. The increase in the precision values shows that our correct prediction rate is increasing. When we increase the null threshold enough, some words cannot be predicted at all, since their highest prediction rate is lower than the null threshold. Therefore, both recall and precision values become 0 after some threshold. Table 1 and table 2 shows

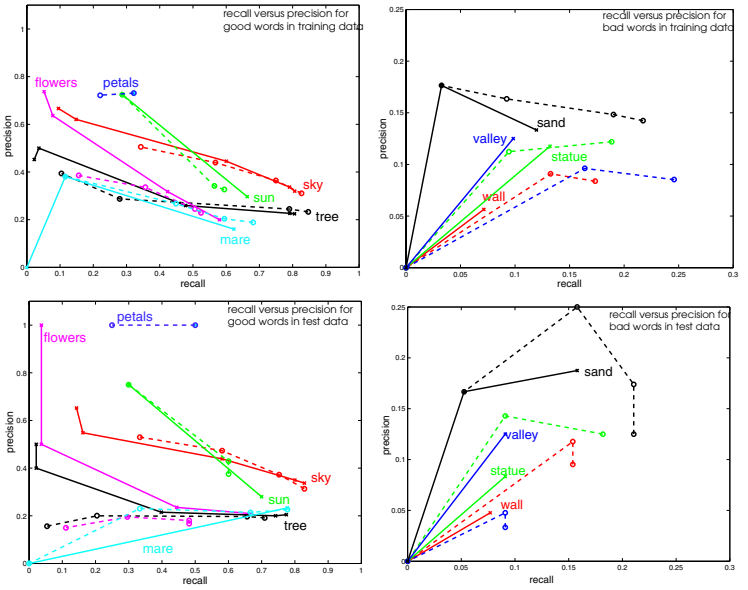


Fig. 6. Recall versus precision for selected words with increasing null threshold values (0-0.4) : On the **left** some good words with high recall and precision, and on the **right** some bad words with low recall and precision are shown. The top line shows the results for training and bottom line shows the results for test. Solid lines show the initial results using all of the original words in training, dashed lines show the results after training on reduced vocabulary. The axes for good words are different than the axes for bad words. The results are very similar both for training and test. Recall values decrease by increasing null threshold, but usually precision increase since the correct prediction rate increase. After a threshold value, all precision and recall may go to 0 since we cannot predict the words anymore.

that, with the increasing null threshold values, the number of words decreases but we have more reliable words. Since null word prediction decreases the word predictions, recall decreases. The increase in the precision shows that null word prediction increases the quality of the prediction.

The effect of word clustering: We also compute recall and precision after clustering the words. As figure 5 shows, recall values of the clusters are higher than recall values of the single words. Table 3 shows that we have some very nice clusters which have strong semantic or visual relations like `kit-horses-mare-foals`, `leaf-flowers-plants-vegetables` or `pool-athlete-vines-swimmers` and the results are better when we cluster the words (compare with table 1).

4.2 Correspondence

Evaluation Method: Because the data set contains no correspondence information, it is hard to check correspondence canonically or for large volumes of data;

Table 1. Some good words with their recall and precision values for increasing null threshold. Words are selected as good if their recall values are greater than 0.4 , and precision values are greater than 0.15. The null threshold changes between 0 and 0.4. With the increasing threshold the number of good words decreases, since we can predict fewer words. While the recall is decreasing precision is increasing, since we predict the remaining words more accurately.

word	**th = 0**	**th = 0.1**	**th = 0.2**	**th = 0.3**	**th = 0.4**
	rec - prec	**rec - prec**	**rec - prec**	**rec - prec**	**rec - prec**
petals	0.50 - 1.00	0.50 - 1.00	0.50 - 1.00	0.50 - 1.00	0.50 - 1.00
sky	0.83 - 0.34	0.80 - 0.35	0.58 - 0.44		
flowers	0.67 - 0.21	0.67 - 0.21	0.44 - 0.24		
horses	0.58 - 0.27	0.58 - 0.27	0.50 - 0.26		
foals	0.56 - 0.29	0.56 - 0.29	0.56 - 0.29		
mare	0.78 - 0.23	0.78 - 0.23			
tree	0.77 - 0.20	0.74 - 0.20			
people	0.74 - 0.22	0.74 - 0.22			
water	0.74 - 0.24	0.74 - 0.24			
sun	0.70 - 0.28	0.70 - 0.28			
bear	0.59 - 0.20	0.55 - 0.20			
stone	0.48 - 0.18	0.48 - 0.18			
buildings	0.48 - 0.17	0.48 - 0.17			
snow	0.48 - 0.17	0.48 - 0.19			

Table 2. Some good words with their recall and precision values for increasing null threshold after reducing the vocabulary only to the predicted words and running the EM algorithm again. Words are selected as good if their recall values are greater than 0.4, and precision values are greater than 0.15. The null threshold changes between 0 and 0.4. When we compare with the original results (table 1), it can be observed that words remain longer, which means that they have higher prediction probabilities. We have more good words and they have higher recall and precision values.

word	**th = 0**	**th = 0.1**	**th = 0.2**	**th = 0.3**	**th = 0.4**
	rec - prec	**rec - prec**	**rec - prec**	**rec - prec**	
petals	0.50 - 1.00	0.50 - 1.00	0.50 - 1.00	0.50 - 1.00	
sky	0.83 - 0.31	0.83 - 0.31	0.75 - 0.37	0.58 - 0.47	
people	0.78 - 0.26	0.78 - 0.26	0.68 - 0.27	0.51 - 0.31	
water	0.75 - 0.25	0.75 - 0.25	0.72 - 0.26	0.44 - 0.27	
mare	0.78 - 0.23	0.78 - 0.23	0.67 - 0.21		
tree	0.71 - 0.19	0.71 - 0.19	0.66 - 0.20		
sun	0.60 - 0.38	0.60 - 0.38	0.60 - 0.43		
grass	0.57 - 0.19	0.57 - 0.19	0.49 - 0.22		
stone	0.57 - 0.16	0.57 - 0.16	0.52 - 0.23		
foals	0.56 - 0.26	0.56 - 0.26	0.56 - 0.26		
coral	0.56 - 0.19	0.56 - 0.19	0.56 - 0.19		
scotland	0.55 - 0.20	0.55 - 0.20	0.45 - 0.19		
flowers	0.48 - 0.17	0.48 - 0.17	0.48 - 0.18		
buildings	0.44 - 0.16	0.44 - 0.16			

Table 3. Some good clusters, where the recall values are greater than 0.4, and precision values are greater than 0.15 when null threshold is 0. Cluster numbers shows how many times we cluster the words and run EM algorithm again. Most of the clusters appear to represent real semantic and visual clusters (e.g.`kit-horses-mare-foals`, `leaf-flowers-plants-vegetables`, `pool-athlete-vines-swimmers`). The recall and precision values are higher than those for single words (compare with table 1).

1st clusters	r	p	2nd clusters	r	p	3rd clusters	r	p
horses mare	0.83	0.18	kit horses mare foals	0.77	0.16	kit horses mare foals	0.77	0.27
leaf flowers	0.69	0.22	leaf flowers plants vegetables	0.63	0.25	leaf flowers plants vegetables	0.60	0.19
plane	0.12	0.14	jet plane arctic	0.46	0.18	jet plane arctic prop flight penguin dunes	0.43	0.17
pool athlete	0.33	0.31	pool athlete vines	0.17	0.50	pool athlete vines swimmers	0.75	0.27
sun ceiling	0.60	0.30	sun ceiling	0.70	0.30	sun ceiling cave store	0.62	0.35
sky beach	0.83	0.30	sky beach cathedral	0.82	0.31	sky,beach cathedral clouds mural arch waterfalls	0.87	0.36
water	0.77	0.26	water	0.72	0.25	water waves	0.70	0.26
tree	0.73	0.20	tree	0.76	0.20	tree	0.58	0.20
people	0.68	0.24	people	0.62	0.26	people	0.54	0.25

instead, each test image must be viewed by hand to tell whether an annotation of a region is correct. Inevitably, this test is somewhat subjective. Furthermore, it isn't practically possible to count false negatives.

Base Results: We worked with a set of 100 test images for checking the correspondence results. The prediction rate is computed by counting the average number of times that the blob predicts the word correctly. For some good words (e.g: `ocean`) we have up to 70% correct prediction as shown in figure 7; this means that, on this test set, when the word `ocean` is predicted, 70% of the time it will be predicted on an ocean region. This is unquestionably object recognition.

It is more difficult to assess the rate at which regions are missed. If one is willing to assume that missing annotations (for example, the ocean appears in the picture, but the word `ocean` does not appear in the annotation) are unbiased, then one can estimate the rate of false negatives from the annotation performance data. In particular, words with a high recall rate in that data are likely to have a low false negative rate.

Some examples are shown in figures 8 and 9. We can predict some words like `sky`, `tree`, `grass` always correctly in most of the images. We can predict the words with high recall correctly, but we cannot predict some words which have very low recall.

The effect of the null prediction: Figure 10 shows the effect of assigning null words. We can still predict the well behaved words with higher probabilities. In

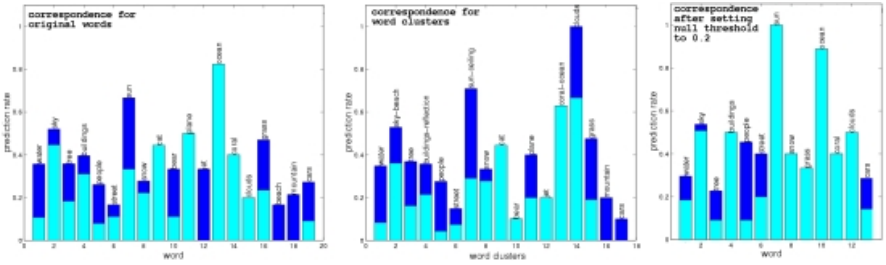


Fig. 7. Correspondence results for 100 test images. **Left**: results for original data, **middle:** after first clustering words, **right:** after assigning null threshold to 0.2. The light bar shows the average number of times that a blob predicts the word correctly in the right place. The dark bar shows the total number of times that a blob predicts the word which is in the image. Good performance corresponds to a large dark bar with a large light bar, meaning the word is almost always predicted and almost always in the right place. For word clusters, for example if we predict `train-locomotive` and either `train` or `locomotive` is keyword, we count that as a correct prediction. We can predict most of the words in the correct place, and the prediction rate is high.

Fig. 8. Some examples of the labelling results. The words overlaid on the images are the words predicted with top probability for corresponding blob. We are very successful in predicting words like `sky`, `tree` and `grass` which have high recall. Sometimes, the words are correct but not in the right place like `tree` and `buildings` in the center image.

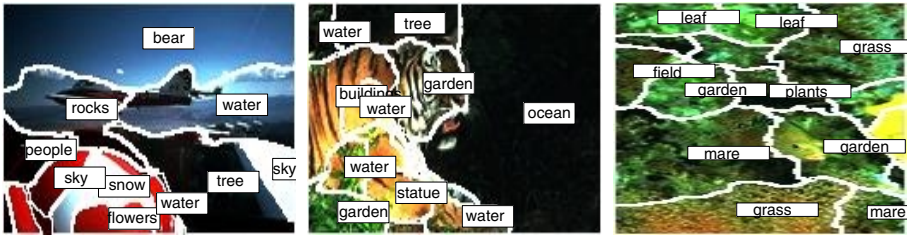


Fig. 9. Some test results which are not satisfactory. Words that are wrongly predicted are the ones with very low recall values. The problem mostly seen in the third image is since green blobs coocur mostly with `grass`, `plants` or `leaf` rather than the under water plants.

figure 7 we show that null prediction generally increases the prediction rate for the words that we can predict.

Fig. 10. Result of assigning null. Some low probability words are assigned to null, but the high probability words remain same. This increases the correct prediction rate for the good words, however we may still have wrong predictions as in the last figure. The confusion between `grass` and `foals` in the second figure is an example of correspondence problem. Since `foals` almost always occur with `grass` in the data, if there is nothing to tell the difference we cannot know which is which.

The effect of word clustering: In figure 11 we show the effect of clustering words. As can be seen generally similar words are grouped into one (e.g. `train-locomotive, horse-mare`). Figure 7 shows that the prediction rate genarally increases when we cluster the words.

5 Discussion

This method is attractive, because it allows us to attack a variety of otherwise inaccessible problems in object recognition. It is wholly agnostic with respect to features; one could use this method for any set of features, and even for feature sets that vary with object definition. It may be possible to select features by some method that attempts to include features that improve recognition performance. There is the usual difficulty with lexicon learning algorithms that a bias in correspondence can lead to problems; for example, in a data set consisting of parliamentry proceedings we expect the English word `house` to translate to the French word `chambre`. We expect — but have not so far found — similar occasional strange behaviour for our problem. We have not yet explored the many interesting further ramifications of our analogy with translation.

- **Automated discovery of non-compositional compounds.** A greedy algorithm for determining that some elements of a lexicon should be grouped might deal with compound words (as in [11]), and might be used to discover that some image regions should be grouped together before translating them.
- **Exploiting shape.** Typically, a set of regions should map to a single word, because their compound has distinctive structure as a shape. We should like to learn a grouping process at the same time as the lexicon is constructed.

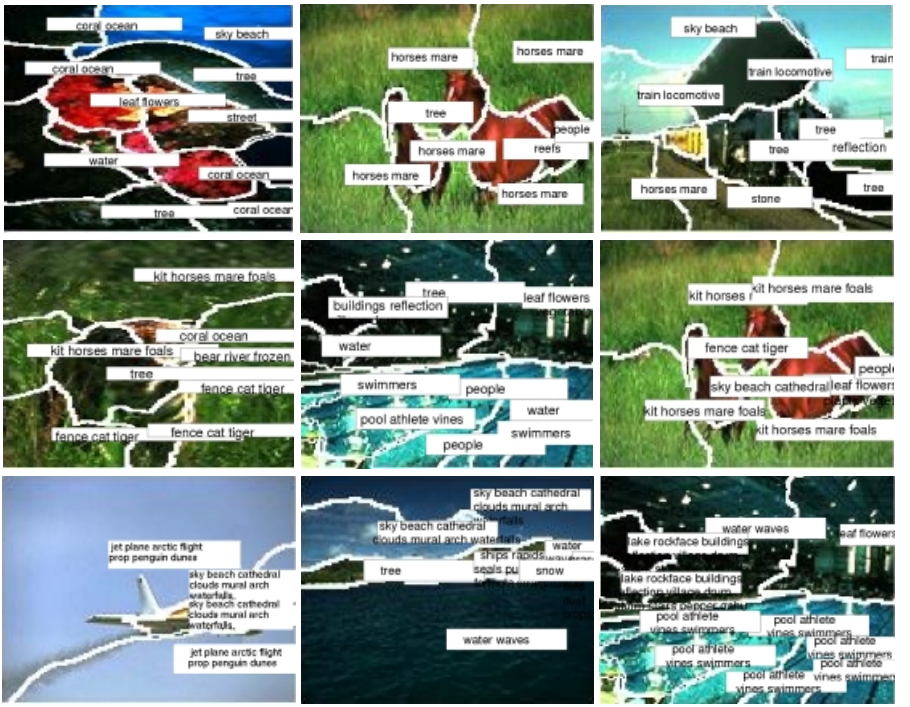


Fig. 11. Result of clustering words after the first, second, and third iterations of clustering. Clustering increases the prediction since indistinguishable words are grouped into one (e.g. `locomotive-train`, `horses-mare`).

– **Joint learning of blob descriptions and the lexicon.** We are currently studying methods that cluster regions (rather than quantizing their representation) to ensure that region clusters are improved by word information.

Acknowledgements. This project is part of the Digital Libraries Initiative sponsored by NSF and many others. Kobus Barnard also receives funding from NSERC (Canada), and Pinar Duygulu is funded by TUBITAK (Turkey). We are grateful to Jitendra Malik and Doron Tal for normalized cuts software, and Robert Wilensky for helpful conversations.

References

1. K. Barnard, P. Duygulu and D. A. Forsyth. Clustering art. In *IEEE Conf. on Computer Vision and Pattern Recognition,* II: 434-441, 2001.
2. K. Barnard and D. A. Forsyth. Learning the semantics of words and pictures. In *Int. Conf. on Computer Vision* pages 408-15, 2001.
3. P. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics,* 32(2):263-311, 1993.

4. D.A. Forsyth and J. Ponce. *Computer Vision: a modern approach.* Prentice-Hall 2001. in preparation.
5. D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice-Hall, 2000.
6. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* MIT Press, 1999.
7. M. Markkula and E. Sormunen. End-user searching challenges indexing practices in the digital newspaper photo archive. *Information retrieval,* 1:259-285, 2000.
8. Y. Mori, H. Takahashi, R. Oka Image-to-word transformation based on dividing and vector quantizing images with words In *First International Workshop on Multimedia Intelligent Storage and Retrieval Management* (MISRM'99), 1999
9. O. Maron. *Learning from Ambiguity.* PhD thesis, MIT, 1998.
10. O. Maron and A. L. Ratan. Multiple-Instance Learning for Natural Scene Classification, In *The Fifteenth International Conference on Machine Learning*, 1998
11. I. Dan Melamed. *Empirical Methods for Exploiting Parallel Texts.* MIT Press, 2001.
12. S. Ornager. View a picture, theoretical image analysis and empirical user studies on indexing and retrieval. *Swedis Library Research,* 2-3:31-41, 1996.
13. J. Shi and J. Malik. Normalised cuts and image segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition,* pages 731-737, 1997.