Image Segment Classification Techniques

Duncan Sommer, Silas Pedrosa

***Abstract*— Computer vision research has examined many approaches to train computers to understand the world. An often overlooked source of information about images is the textual captions that accompany them; using this method could improve entity identification. As one part of that goal, this project focuses on identifying entities in images using a manually annotated database without handling the captions.**

I. Introduction

Classifying objects within images using captions requires several tasks to be performed, including image processing, segment classification, natural language processing (NLP), and combining the various results. The raw image data is first split into segments, and then each segment is processed to generate a numerical "feature vector" that describes its characteristics, such as area, color and convexity. These features are used to classify each segment, while NLP techniques find entities and their relations in the caption. Using both parts together offers the potential for better results.

This project attempts to accurately identify individual segments using only the feature information extracted from the images. The dataset used for development and testing is the Segmented and Annotated IAPR TC-12 benchmark (SAIAPR), which is comprised of nearly 20,000 images split into segments, their pre-extracted features, and labels for each segment manually chosen from a list of classes.

II. System Architecture

The overall idea behind a classifier is that it receives some classified samples, is trained with them and becomes able to predict the class of a new, unclassified sample. In more detail, it must receive some samples, each one containing features and its classification as well as the list of all possible classifications. By making use of a statistical classification method it trains on these so called *training samples* (or *training group*) and learns how to predict the classification of any new sample, which is called a *test sample*. Actually, the classifier output for one test sample is a vector containing the probabilities of the sample being each one of the possible classes. Logistic regression with cross validation is used as the statistical classification method, provided through a Python interface to the Liblinear library.

After the manual segmentation of the images, each segment has its features extracted and the manual classification is recorded. This feature vector along with the classification is a sample. These samples are split in three groups: the training group for the local classifier (TG1), the training group for the reranker (TG2) and the test group (TEST). TG1 is used to train the local classifier, which affects the following results as well, making this sample group critically important. TG2 is used as a test group for the local classifier and as the training group for the reranker. TEST is tested with the final classifier, which outputs the final results. These results, however, are not a vector containing the probabilities of the given sample being each one of the possible classes, but just the most probable class and its related likelihood.

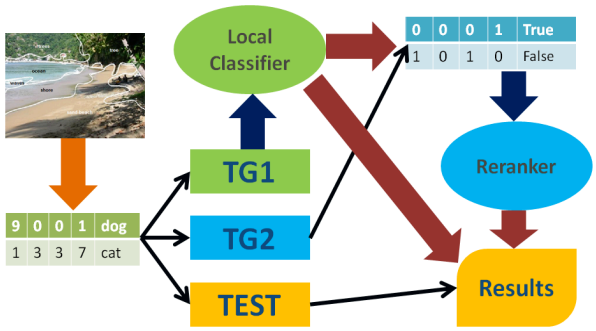

Figure 1: Data Flow
Images have their features extracted, and these samples are split into three groups. The groups are used to train the local classifier, generate reranker samples, and test.

III. Local Classifier

The local classifier outputs the predictions generated by logistic regression with cross validation. It has three parameters that heavily affect the results:

A. Clustering

There are 273 classes used in the SAIAPR annotations, but if the clustering parameter is set, these 273 classes are grouped into 13 classes. This offers greater speed and accuracy, but information is lost when using many fewer classes. The conceptual clustering of the classes was made manually, using only empirical semantic criteria.

B. Sample Limit

This parameter limits the maximum amount of samples used. If it is set, only the first samples from the whole dataset are used. Although initially implemented for development purposes, interesting results were found with this setting during test. In this project, the standard limit used was 5000.

C. Segment Limit

If set, only the images containing fewer segments than this limit will be considered, meaning that the classifier will only deal with simpler images. For testing, a segment limit of 5 was used when limiting the number of segments.

IV. Reranker

After the initial implementation and testing of the local classifier, a secondary algorithm was designed to improve upon the original accuracy. The local version of the algorithm processes just a single segment at a time; it has no knowledge of the other segments in the image. However, there is plenty of information in the relationships between segments - for example, a table is very likely to appear near a chair. The goal of the reranker is to take the entire image into consideration, so that it can choose labels for each segment while knowing what is going on nearby.

The reranker receives the top predictions for each segment from the local classifier, and attempts to reorder them to achieve better results.

This project's implementation applies the same logistic regression model as the local classifier, but with a different data set. While the local classifier determines the likelihood of an object's class from its visual characteristics, the reranker is trained to determine the likelihood of a certain combination of objects. With this in mind, it is necessary to build a dataset usable by such a classifier.

In order to train the reranker classifier, data representing actual combinations of observed objects was needed, as well as some false samples. Predictions were taken from the local classifier's results on the training set to generate samples for this purpose. If every possible unobserved combination of images was included in the false samples, the overwhelming bias for predicting false would make logistic regression unusable. Instead, for each image, the top guess for each segment is chosen, creating a single sample. Then, the segments with the most uncertain predictions are expanded upon to create more samples, which represent the local classifier's best ideas of what constitutes a coherent image. The gold-standard sample for each image is added into the new training set if it was not included in the local classifier's top predictions. With this method, the proportion of positive examples observed in training can be adjusted; in this implementation, 5% positive samples were used.

The new feature set and its ground truth classifications are used to train the reranker algorithm. Once prepared, this model can be used to determine the likelihood that a certain combination of entities in an image will appear together.

V. Final Classifier

The reranker algorithm does not stand alone. Instead, it must be used alongside predictions from the local classifier, and their decisions merged for the best possible results. When making final predictions on the test set, the local classifier is first run on each segment. From these segment classifications, samples for the reranker are generated using the same process as for training, but without knowledge of the true classes. Then, once the reranker calculates the likelihood of each combination, a final decision on each image can be made.

For each option, the following equation gives the final decision value:

$$V = (P_{RR})^W \bullet \prod(P_i).$$

V represents the decision value, P_{RR} is the probability given by the reranker, W is a weight given to the reranker and used to tune its influence, and $\prod(P_i)$ is the product of each segment's probability.

The best guess is the set of labels for an image with the highest probability. The final classifier decides upon an entire image at a time, and picks the best overall assignments for the segments. However, many of these terms are extremely small probabilities, so logarithms are used to prevent underflow:

$$V = W \bullet \log(P_{RR}) + \sum \log(P_i).$$

All samples in the dataset are processed in this way, and the final accuracy is calculated for analysis using the withheld gold-standard classifications.

VI. Results

The accuracy of both classifiers depended heavily on the parameters described for the local classifier. The original classifier quickly achieved the level of accuracy expected, but the reranker required extensive fine-tuning in order to improve upon it. Shown below are the final tests for both algorithms; the instances where the reranker helped the accuracy are highlighted.

Figure 2: Local Classifier Accuracy (%)

	No Segment Limit		Segment Limit = 5	
	All Samples	5000 Samples	All Samples	5000 Samples
Clustered	47.44	46.25	55.90	54.47
Unclustered	24.92	22.13	34.28	35.27

Figure 3: Reranker Accuracy (%)

	No Segment Limit		Segment Limit = 5	
	All Samples	5000 Samples	All Samples	5000 Samples
Clustered	47.46	46.30	55.89	55.26
Unclustered	24.92	22.40	34.27	35.17

The effects of clustering were the most drastic, as using only 13 labels improved the accuracy by more than 20% in most cases. The next most important parameter was the segment limit, and it was observed that using the simpler images also offered an increase of around 10%. Of least importance was the sample limit, but this was still essential for positive reranker results; three out of the four instances where the reranker improved the accuracy had a sample limit imposed.

VII. Conclusion

Computer vision is a very broad field and has many applications. Some methods suit some applications better than others, due to different requirements, such as speed and accuracy. The initial approach taken in this project led to results that, by themselves, couldn't be directly applied to any real world problem. The reranker made improvements in certain cases, but still too few to change the applicability of the system.

These small improvements, however, make us believe that there are several other ways to improve the system. For instance, the optimal number of classes and better clustering were not researched; only the two clustering setups described here were tested. Another thing that could have a great impact is taking another approach for the reranker. For example, a naive classifier would make it possible to test various hypotheses regarding the semantical and statistical dependencies between the segments in the same image.

This project is just a part of a greater goal. When used with the textual caption processing, the accuracy of this system can be increased significantly. In a simplistic approach, the classes recognized in the captions would have their probabilities increased in the predictions from the classifiers. NLP is well studied and these results would be very reliable, making us believe that integrating results from the captions would lead to a powerful improvement.

References

[1] A. Tegen, R. Weegar, L.Hammarlund, M. Oskarsson, F. Jiang, D. Medved, P. Nugues, and K. Åström, "Image segmentation and labeling using free-form semantic annotation."

[2] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification."