A Noise-Resistant Long-Term Learning Approach for Content-Based Image Retrieval

Jacob Linenthal
August 10, 2007

Funded by the NSF
What is Content Based Image Retrieval?

- The goal of a CBIR system is to return images which are related to an image supplied by the user.
- Suppose the query image is
What is CBIR? (cont)

• Then the following images are could be returned, based solely on the similarity of the low-level feature vector.
Relevance Feedback

• The user classifies the returned images as either relevant or irrelevant.
Relevance Feedback

- The feedback is then used to help classify other images in the database.
- More on this later...
Why is it hard?

- Semantic Gap – Low-level features of the image do not capture the high-level semantic content, while users are typically looking for semantically similar images. For example, these both represent images of a desert, but their low level features are quite different.
Why is it hard? (cont.)

- The problem of noise: users may not always classify images correctly. If there is too much erroneous feedback, noise can wreak havoc on the further classification of images.
- Causes:
  - Subjectivity – the same users will not necessarily give the same feedback
  - Laziness – the user might be flippant about the feedback
  - Maliciousness – the user may want to deliberately affect the effectiveness of the system.
Proposed Approach

- We combine a low-level learning scheme along with a high-level, long-term learning scheme.
- Intuitively, the high-level approach learns the semantic connections between images from the user's feedback over many query sessions. We experimented with a number of different ways to compute the high-level similarity scores.
- The low-level learning only happens over one query session (i.e. no information is saved)
Low-level Approach

- Image can be represented as a vector in $\mathbb{R}^n$
- In the first iteration, the euclidean distance is used to find the images closest to the query.
- In future iterations, a Support Vector Machine (SVM) is used to find a separating hyperplane between the positive and negative feedback.
- Other images in the database can then be classified as either positive or negative using the SVM.
High-Level Learning

- We get semantic information about the images in the database from the user's feedback.
- Suppose two images are returned in the same query session.
  - If both are classified as relevant to the query image, then they are clearly related.
  - If one is classified as positive and one as negative, this counts as evidence that they are not related.
  - If both are negative, we gain almost no information.
High-Level Learning (cont.)

- We define the “semantic similarity” of two images by the ratio of how many times they were returned and both classified as positive to how many times they were returned and at least one was classified as positive.
- In this way, we can calculate the high-level similarity of the query image and the rest of the database and return the best images to the user.
Refined High-Level Similarity

- However, we don't have to consider only the distance to the query image.
- After the user has provided some positive and negative feedback, we can calculate the semantic similarity with the *feedback set* instead of just the query image.
- Why do this?
  - May not have information about direct connection of images and the query image, but there may be a connection with the feedback set.
Semantic Transitivity

- Using the feedback set to improve the semantic similarity calculation rests on the idea of semantic transitivity.
- Intuitively, this says that if image a is related to image b, and image b is related to image c, then it is likely that image a and image c are also related.
Method 1

• Take the best semantic match in the set and count this as the semantic similarity

• Problem – Noise!
  - Semantic transitivity will cause many incorrect images to be classified as close to the positive feedback set.
Method 2

- Instead of taking the best similarity value, we take the mean of the similarities.
  - Accounts for noise – one misclassified image can't ruin everything now.
- This works well, but we're not using all the information we can...
Method 3

- Although the semantic similarity values range between zero and one, identical values do not indicate an identical correlation.
  - If two images have only been classified together once, then we don't have much confidence in it's similarity value.
  - On the other hand, if two images have repeatedly been classified together as relevant, then we can be confident that the two are related.
- Hence, we weight the mean by how many times they have been classified together
Method 4

- One further source of information is the idea that some images in the positive feedback set may be more representative of the desired semantic concept than others.
- If an image in the database has a connection to a strongly representative image, then this should count more than if it has a connection to a weakly representative image.
- We also weight the mean by the strength of the images representation.
Other Improvements

- Automatically suggesting feedback: if an image is sufficiently close to the positive feedback set, it is automatically added.
- The similarity to the negative feedback set can also be computed using similar methods. We'd like images to be very similar to the positive feedback set, but not similar to the negative feedback set.
Results

- We tested on a 6,000 image Corel image database, with 5% simulated noise, training on 10% of the database.
- The following shows the viability of the high-level approach.
Results (cont.)

- The following shows the degradation of the results using Method 1 (red) versus Method 2 (green).
The following shows the final results of our approach, using the different methods of computing semantic similarity. The red line represents Method 2, the green, Method 3, and the blue, Method 4.
Questions?

No monkeys here.