Snaphappi automation

Alexey Badalov 2012-06-10

# Architecture

I assume that all major decisions for rating photos are made by experts; the algorithms, which I will call the AI, provide hints or defaults for the experts. Once a user uploads a set of images, it is sent for automatic processing, which extracts GIST descriptors, computes hints, and then stores them in the database. An expert may choose to wait for automatic processing to complete or not, to follow hints or ignore them.

Note that AI-made decisions should always be stored separately from human-made decisions and should not be exposed directly to the user. For instance, they might be stored in separate database table columns. Even if you want to accept an AI decision without verifying it, it is better to make a copy. This is important for two reasons: first, AI decisions may change over time, as the system evolves; second, we want to know when expert decisions are the same or different from the AI ones in order to improve the latter.

Training is an important step. Many algorithms are controlled by parameters that have no intuitive or generally correct values. By comparing expert and AI decisions, we can evaluate algorithm performance and adjust parameters to bring AI performance closer to the human expert level. Because AI training can be computationally expensive, it should be done on machines other than the ones serving users or scheduled for times of low traffic. Training would be controlled by scripts that would take sample data and expert decision records, perform trial runs of algorithms with different choices of parameters, and then update these parameters for production.

The AI should not access the database directly. Rather, the backend should provide an API to allow updating hints, as well as getting sample data sets of images and expert decisions. Ideally, the AI should also have an API, so that all communication occurs through formalized channels.

The following page shows a sample sequence diagram for the task of ranking images.



Figure . Sequence diagram for ranking images.

# Use Cases

## Grouping photos

The goal is to group multiple shots of the same scene, so that the best shot could be chosen. We can assume that all images within a group have similar time stamps, but images with similar time stamps do not necessarily belong to the group.

A convenient way of representing photo groups is to order them by time and mark the first photo in each group. This is preferable to the more general unordered group representation, because it allows us to compare expert and AI decisions. An expert could group photos by viewing them ordered by time and marking those photos that do not begin a new group. The AI could assign groups by comparing GIST descriptor differences between consecutive images against a threshold parameter. Mistakes by AI could be detected when it misses a group separator set by an expert or adds a separator where an expert does not.

## Rating photos

Photos are rated on a six-point scale. An image that is blurry, severely over- or underexposed receives -1. An image lacking a clear subject, having poor composition, or minor technical problems is rated 1-3. An image that could make a magazine cover is rated 5.

There exist both algorithmic and machine learning methods for detecting technically poor images. For instance, [[1](#Pot09)] uses machine learning to confidently detect photos with exposure defects, such as over- and underexposed, backlit, blurred photos as well as images affected by strong JPEG artifacts. AI could help detect images with 1-2 ratings, and possibly higher. Mistakes by AI could be seen as differences from expert ratings.

## Wedding pictures

With a wedding event, we are dealing with several streams of photos. Many photos are taken of the same events. There might be 3-4 thousand images in an event. Streams might have time stamps based on different time zones.

We could try to determine time zones algorithmically by clustering all photos together, and then calculating the probability of different time offsets for each stream based on which other streams it as been clustered with. The assumption would be that similar photos are likely taken from the same event and at similar times.

## Montage

We want to take photos grouped by time, person, or event and display them all on pages. Each page would have a set number of images, with a large top-rated photo and smaller lower-rated ones.

# Bibliography

x

|  |  |
| --- | --- |
| [1] | Ekaterina Potapova, Marta Egorova, and Ilia Safonov, "Automatic Photo Selection for Media and Entertainment Applications," *GRAPHICON 2009*, 2009. |

x