Ingesting 35M images with Python
In the cloud.

Àlex Vinyals
Software Engineer @ Hotels Data
Unify all the data

Challenges of a metasearch
W Barcelona 20 Apr 2016 - 21 Apr 2016 1 guest, 1 room

18 results near W Barcelona, sorted by Most popular

Selected hotel

W Barcelona ★★★★★
[see large map]

Single room
Room only • Non refundable

$488 $356 [View deal]

Nearby hotels

Eurostars Grand Marina Hotel ★★★★★

0.53 mi. to W Barcelona (map)

Single room
Room only • Non refundable

$2,146 $195 [View deal]

Hotel 54 Barceloneta ★★

0.55 mi. to W Barcelona (map)

Single room
Room only • Non refundable

$2,146 $210 [View deal]
### Selected hotel

<table>
<thead>
<tr>
<th>Hotel Name</th>
<th>Rating</th>
<th>Price</th>
<th>Deal Price</th>
<th>View Deal</th>
</tr>
</thead>
<tbody>
<tr>
<td>W Barcelona</td>
<td>★★★★★</td>
<td>$488</td>
<td>$356</td>
<td>View deal</td>
</tr>
</tbody>
</table>

### Nearby hotels

<table>
<thead>
<tr>
<th>Hotel Name</th>
<th>Price</th>
<th>Deal Price</th>
<th>View Deal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eurostars Grand Marina Hotel GI</td>
<td>$2,146</td>
<td>$195</td>
<td>View deal</td>
</tr>
<tr>
<td>Hotesar</td>
<td>$2,146</td>
<td>$198</td>
<td>View deal</td>
</tr>
<tr>
<td>Getaroom</td>
<td>$2,146</td>
<td>$210</td>
<td>View deal</td>
</tr>
</tbody>
</table>

### Filters

- **Total Price**
  - $0 - $55: 1
  - $10 - $165: 1
  - $165 - $250: 1
  - $250+: 1
- **Stars**
  - ★★★★★: 2
  - ★★★★: 4
  - ★★★: 4
  - ★☆☆☆☆: 5
- **District**
  - Ciutat Vella: 15
- **Meal Plan**
  - Room only: 4
Partner A
Hotel ID 123
Name Euskalduna Center
Street address Avenida Abandoibarra 3
Coordinates 1.23, 2.43

Partner B
Hotel ID $abc
Name Euskalduna Conference Center
Street address Av. Abandoibarra 3
Coordinates 1.23754, 2.43123

Partner C
Hotel ID bilbao-hot1
Name Euskalduna CC
Street address Avda. Abandoibarra3, 48009
Coordinates 1.238, 2.431

Skyscanner
Hotel ID 123456
Name Euskalduna Conference Center
Street address Av. Abandoibarra 3
Coordinates 1.23754, 2.43123
<table>
<thead>
<tr>
<th>Partner A</th>
<th>Hotel ID</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Euskalduna Center</td>
<td></td>
</tr>
<tr>
<td>Street address</td>
<td>Avenida Abandoibarra 3</td>
<td></td>
</tr>
<tr>
<td>Coordinates</td>
<td>1.23, 2.43</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partner B</th>
<th>Hotel ID</th>
<th>$abc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Euskalduna Conference Center</td>
<td></td>
</tr>
<tr>
<td>Street address</td>
<td>Av. Abandoibarra 3</td>
<td></td>
</tr>
<tr>
<td>Coordinates</td>
<td>1.23754, 2.43123</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partner C</th>
<th>Hotel ID</th>
<th>bilbao-hot1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Euskalduna CC</td>
<td></td>
</tr>
<tr>
<td>Street address</td>
<td>Avda. Abandoibarra 3, 48009</td>
<td></td>
</tr>
<tr>
<td>Coordinates</td>
<td>1.238, 2.431</td>
<td></td>
</tr>
</tbody>
</table>

---

**Skyscanner**

<table>
<thead>
<tr>
<th>Hotel ID</th>
<th>123456</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Euskalduna Conference Center</td>
</tr>
<tr>
<td>Street address</td>
<td>Av. Abandoibarra 3</td>
</tr>
<tr>
<td>Coordinates</td>
<td>1.23754, 2.43123</td>
</tr>
</tbody>
</table>

---

**Data Release**
So what about the images?
Partner A
Hotel ID
123

Partner B
Hotel ID
$abc

Partner C
Hotel ID
bilbao-hot1

Magic Happens

Skyscanner
Hotel ID
123456
With more than 200 partners
800,000 hotels reach production
Images to process = $K \times M \times N \sim 35M$ images

$K =$ number of partners
$M =$ avg number of hotels per partner
$N =$ avg number of images per partner hotel
Resizing is a thing
And we have 14 different configurations

```
[sizes]
628x470, scaledownfit
314x235, scaledownfit
2048x1536, scaledownfit
1024x768, scaledownfit
200x200, scalefill, crop
100x100, scalefill, crop
280x185, 185x280, scaledownfill, crop, contrast 1.05
102x182, scaledownfill, crop, contrast 1.05
102x78, scaledownfill, crop, contrast 1.1
50x58, scaledownfill, crop, contrast 1.1
480x319, 240x319, scaledownfill, crop, contrast 1.05
45x45, scaledownfill, crop, contrast 1.1
627x470, scalefill, crop
313x235, scalefill, crop
```
Tale of an image processing pipeline
Tech Stack
Riding on AWS
Tech Stack

Riding on AWS

SQS
Simple Queue Service
Tech Stack
Riding on AWS

SQS
Simple Queue Service
*with DjangoRestFramework
*without Django ORM

Libraries
Libraries
Libraries

Kombu
Messaging / queues / amqp

*with DjangoRestFramework
*without Django ORM
Libraries

Kombu
Messaging / queues / amqp

Boto
Amazon stuff
Libraries
Python 2.7

Kombu
Messaging / queues / amqp

Pillow
Image Processing

SQLAlchemy

Boto
Amazon stuff
Tale of an image processing pipeline
Tale of an image processing pipeline

Triggered by the Data Release

Asynchronous (Always Running)

- Triggering → Downloading → Fingerprinting

- Deduplicating → Prioritising → Generating
Triggering → Downloading → Fingerprinting

Triggering

deduplication → prioritising → generating
<table>
<thead>
<tr>
<th>id</th>
<th>url</th>
<th>partner_id</th>
<th>acc_id</th>
<th>status</th>
<th>en</th>
</tr>
</thead>
<tbody>
<tr>
<td>23452927</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452917</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452848</td>
<td><strong><a href="http://img1.hotels/CAM">http://img1.hotels/CAM</a></strong></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452841</td>
<td><strong><a href="http://img1.hotels/CAM">http://img1.hotels/CAM</a></strong></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452846</td>
<td><strong><a href="http://img1.hotels/CAM">http://img1.hotels/CAM</a></strong></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452844</td>
<td><strong><a href="http://img1.hotels/CAM">http://img1.hotels/CAM</a></strong></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452851</td>
<td><strong><a href="http://img1.hotels/CAM">http://img1.hotels/CAM</a></strong></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452796</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452797</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452794</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452788</td>
<td><strong><a href="http://www.hotels.com/ima">http://www.hotels.com/ima</a></strong></td>
<td>18</td>
<td>931146</td>
<td>filtered</td>
<td>t</td>
</tr>
<tr>
<td>23452914</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452798</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452802</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452922</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452919</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452795</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452924</td>
<td><strong><a href="http://hotels.com/h">http://hotels.com/h</a></strong></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452799</td>
<td><strong><a href="http://d1palerts.cloudflare">http://d1palerts.cloudflare</a></strong></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>id</td>
<td>url</td>
<td>partner_id</td>
<td>acc_id</td>
<td>status</td>
<td>en</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>----</td>
</tr>
<tr>
<td>23452927</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452917</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452848</td>
<td><a href="http://hotels.CAM">http://hotels.CAM</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452841</td>
<td><a href="http://hotels.CAM">http://hotels.CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452846</td>
<td><a href="http://hotels.CAM">http://hotels.CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452844</td>
<td><a href="http://hotels.CAM">http://hotels.CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452851</td>
<td><a href="http://hotels.CAM">http://hotels.CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452796</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452797</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452794</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452788</td>
<td><a href="http://www.1pas.com/ima">http://www.1pas.com/ima</a></td>
<td>18</td>
<td>931146</td>
<td>filtered</td>
<td>t</td>
</tr>
<tr>
<td>23452914</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452798</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452802</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452922</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452919</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452795</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452924</td>
<td><a href="http://hotels.com/h">http://hotels.com/h</a></td>
<td>4</td>
<td>3969975</td>
<td>available</td>
<td>t</td>
</tr>
<tr>
<td>23452799</td>
<td><a href="http://d1pa.cloudfro">http://d1pa.cloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>id</td>
<td>url</td>
<td>partner_id</td>
<td>acc_id</td>
<td>status</td>
<td>en</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------</td>
<td>----</td>
</tr>
<tr>
<td>23452927</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452917</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452848</td>
<td><a href="http://img.hotel/CAM">http://img.hotel/CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452841</td>
<td><a href="http://img.hotel/CAM">http://img.hotel/CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452846</td>
<td><a href="http://img.hotel/CAM">http://img.hotel/CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452844</td>
<td><a href="http://img.hotel/CAM">http://img.hotel/CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452851</td>
<td><a href="http://img.hotel/CAM">http://img.hotel/CAM</a></td>
<td>5</td>
<td>THN3J4</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452796</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452797</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452794</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452788</td>
<td><a href="http://www.hotel.com/ima">http://www.hotel.com/ima</a></td>
<td>18</td>
<td>931146</td>
<td>filtered</td>
<td>t</td>
</tr>
<tr>
<td>23452914</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452798</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452802</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452922</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452919</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452795</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
<tr>
<td>23452924</td>
<td><a href="http://hotels.com/hotele">http://hotels.com/hotele</a></td>
<td></td>
<td>4</td>
<td>3969975</td>
<td>t</td>
</tr>
<tr>
<td>23452799</td>
<td><a href="http://d1pacloudfro">http://d1pacloudfro</a></td>
<td>9</td>
<td>252966</td>
<td>production</td>
<td>t</td>
</tr>
</tbody>
</table>
Downloading

Deduplicating → Prioritising → Generating

Triggering → Downloading → Fingerprinting
import io
import boto
import requests
from PIL import Image

s3 = boto.connect_s3()
bucket = s3.get_bucket('available-images')

@reliable_callback()
def downloader_callback(queued_image):
    """ Overly simplified downloading callback without error handling logic ""
    response = requests.get(queued_image.url)
blob = response.content
key = bucket.new_key(queued_image.basename)
key.set_contents_from_string(blob)

    image = Image.open(io.BytesIO(blob))
    if should_filter(image):
        return

    fingerprinting_producer.publish(queued_image)

def should_filter(image):
    height, width = image.size

    short_size = min(width, height)
    if short_size < minimum_short:
        return True

    long_size = max(width, height)
    if long_size < minimum_long:
        return True

    total_pixels = width * height
    if total_pixels > max_pixels:
        return True

    return False
import io
import boto
import requests
from PIL import Image

s3 = boto.connect_s3()
bucket = s3.get_bucket('available-images')

@reliable_callback()
def downloader_callback(queued_image):
    """ Overly simplified downloading callback without error handling logic """
    response = requests.get(queued_image.url)
    blob = response.content
    key = bucket.new_key(queued_image.basename)
    key.set_contents_from_string(blob)
    image = Image.open(io.BytesIO(blob))
    if should_filter(image):
        return
    fingerprinting_producer.publish(queued_image)

def should_filter(image):
    height, width = image.size
    short_size = min(width, height)
    if short_size < minimum_short:
        return True
    long_size = max(width, height)
    if long_size < minimum_long:
        return True
    total_pixels = width * height
    if total_pixels > max_pixels:
        return True
    return False
import io
import boto
import requests
from PIL import Image

s3 = boto.connect_s3()
bucket = s3.get_bucket('available-images')

@reliable_callback()
def downloader_callback(queued_image):
    """ Overly simplified downloading callback without error handling logic """
    response = requests.get(queued_image.url)
    blob = response.content
    key = bucket.new_key(queued_image.basename)
    key.set_contents_from_string(blob)
    image = Image.open(io.BytesIO(blob))
    if should_filter(image):
        return
    fingerprinting_producer.publish(queued_image)

def should_filter(image):
    height, width = image.size
    short_size = min(width, height)
    if short_size < minimum_short:
        return True

    long_size = max(width, height)
    if long_size < minimum_long:
        return True

    total_pixels = width * height
    if total_pixels > max_pixels:
        return True
    return False
import functools
import warnings
from PIL import Image

def reliable_callback():
    def decorator(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            warnings.simplefilter('error', Image.DecompressionBombWarning)
            try:
                return func(*args, **kwargs)
            except BaseException:
                logger.error("Critical worker error", exc_info=True)
            return wrapper

    return decorator
import functools
import warnings
from PIL import Image

def reliable_callback():
    def decorator(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            warnings.simplefilter('error', Image.DecompressionBombWarning)
            try:
                return func(*args, **kwargs)
            except BaseException:
                logger.error("Critical worker error", exc_info=True)
            return wrapper
    return decorator
import functools
import warnings
from PIL import Image

def reliable_callback():
    def decorator(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            warnings.simplefilter('error', Image.DecompressionBombWarning)
            try:
                return func(*args, **kwargs)
            except BaseException:
                logger.error("Critical worker error", exc_info=True)
            return wrapper
    return decorator
import functools
import warnings
from PIL import Image

def reliable_callback():
    def decorator(func):
        @functools.wraps(func)
        def wrapper(*args, **kwargs):
            warnings.simplefilter('error', Image.DecompressionBombWarning)
            try:
                return func(*args, **kwargs)
            except BaseException:
                logger.error("Critical worker error", exc_info=True)
            return wrapper
    return decorator
from kombu import Connection, Consumer, Exchange, Queue, eventloop

class KombuConsumer(common.BaseConsumer):
    # ... bla bla

    def callback(self, body, message):
        self.handler(body)
        message.ack()

    def listen(self):
        with Connection(self.backend.broker, transport_options={'region': self.backend.region}) as connection:
            with Consumer(connection, self.queue, callbacks=[self.callback], accept=[self.backend.serializer]):
                for _ in eventloop(connection):
                    pass

# What a simplified worker looks like
# Broker URI stored on Backend object, looks like:
# sqs://{s3_key}:{s3_secret}@
consumer = KombuConsumer(backend, handler=downloader.downloader_callback)
consumer.listen()
from kombu import Connection, Consumer, Exchange, Queue, eventloop

class KombuConsumer(common.BaseConsumer):
    # ... bla bla

    def callback(self, body, message):
        self.handler(body)
        message.ack()

    def listen(self):
        with Connection(self.backend.broker, transport_options={'region': self.backend.region}) as connection:
            with Consumer(connection, self.queue, callbacks=[self.callback], accept=[self.backend.serializer]):
                for _ in eventloop(connection):
                    pass

# What a simplified worker looks like
# Broker URI stored on Backend object, looks like:
# sqs://{s3_key}:{s3_secret}@
consumer = KombuConsumer(backend, handler=downloader.downloader_callback)
consumer.listen()
from kombu import Connection, Consumer, Exchange, Queue, eventloop

class KombuConsumer(common.BaseConsumer):
    # ... bla bla

    def callback(self, body, message):
        self.handler(body)
        message.ack()

    def listen(self):
        with Connection(self.backend.broker, transport_options={'region': self.backend.region}) as connection:
            with Consumer(connection, self.queue, callbacks=[self.callback], accept=[self.backend.serializer]):
                for _ in eventloop(connection):
                    pass

# What a simplified worker looks like
# Broker URI stored on Backend object, looks like:
#   sqs://{s3_key}:{s3_secret}@
c
consumer = KombuConsumer(backend, handler=downloader.downloader_callback)
cconsumer.listen()
from kombu import Connection, Consumer, Exchange, Queue, eventloop

class KombuConsumer(common.BaseConsumer):
    # ... bla bla

    def callback(self, body, message):
        self.handler(body)
        message.ack()

    def listen(self):
        with Connection(self.backend.broker, transport_options={'region': self.backend.region}) as connection:
            with Consumer(connection, self.queue, callbacks=[self.callback], accept=[self.backend.serializer]):
                for _ in eventloop(connection):
                    pass

# What a simplified worker looks like
# Broker URI stored on Backend object, looks like:
# sqs://s3_key:s3_secret@
consumer = KombuConsumer(backend, handler=downloader.downloader_callback)
consumer.listen()
Fingerprinting

Deduplicating → Prioritising → Generating

Fingerprinting
Are those images the same?
Are those images the same?
Are those images the same?
Yes, they are
import imagehash

def fingerprint_callback(queued_image):
    blob = download_image_blob(queued_image.basename)
    image = Image.open(BytesIO(blob))
    result = cropped_hash(image, imagehash.phash)
    store_hashes(queued_image.image_id, result)
import imagehash

def fingerprint_callback(queued_image):
    blob = download_image_blob(queued_image.basename)
    image = Image.open(BytesIO(blob))
    result = cropped_hash(image, imagehash.dhash)
    store_hashes(queued_image.image_id, result)
```python
def cropped_hash(image, algorithm, steps=range(0, 51, 10)):
    result = []
    w, h = image.size
    # We want to cut by steps % of the image (default 0%, 10%...50%), which
    # means we need to cut half of that from each side:
    # | N%/2 |                   | N%/2 |
    # +------------------------------------------+
    # | : |                   | :   |
    # +-----+-------------------------------------+-
    # | | N%/2 |                   | |
    # | | : |                   | |
    # | | : |                   | |
    # | | : |                   | |
    # | | : |                   | |
    # +-----+-------------------------------------+-
    for x in steps:
        x_band = x * w / 200
    for y in steps:
        y_band = y * h / 200
        with image.crop((x_band, y_band, w-x_band, h-y_band)) as sub_image:
            sub_hash = algorithm(sub_image)
            result.append(hash_to_int(sub_hash))
```

def cropped_hash(image, algorithm, steps=range(0, 51, 10)):
    result = []
    w, h = image.size

    # We want to cut by steps % of the image (default 0%, 10%...50%), which
    # means we need to cut half of that from each side:
    #          | N%/2 |
    # +------|------+
    # |      :          :      | N%/2 |
    # +--------------------------+-
    # | : | : | N%/2 |
    # + - + - + + + + + + + + + +--
    # | | | | |
    # | | | |
    # + - - + + + + + + + + + +--
    # | : | : | N%/2 |
    # +--------------------------+-

    for x in steps:
        x_band = x * w / 200
        for y in steps:
            y_band = y * h / 200
            with image.crop((x_band, y_band, w - x_band, h - y_band)) as sub_image:
                sub_hash = algorithm(sub_image)
                result.append(hash_to_int(sub_hash))
Deduplication Time

Triggering → Downloading → Fingerprinting

Deduplicating → Prioritising → Generating
Data Release

CSV with 1M groups of hotel IDs

API

Deduplicators

Queue

RDS/postgres

Prioritizers Queue

If needed

Deduplicator

Group Payloads

CSV with 1M

groups of hotel IDs
## What is a “group”?

<table>
<thead>
<tr>
<th>id</th>
<th>dr</th>
<th>group_id</th>
<th>status</th>
<th></th>
<th>elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>10220644</td>
<td>70</td>
<td>2053940</td>
<td>done</td>
<td>17,454995,4,501446,13,1003798</td>
<td></td>
</tr>
<tr>
<td>10220643</td>
<td>70</td>
<td>2054825</td>
<td>done</td>
<td>34,102912,4,363502,28,37152,46</td>
<td></td>
</tr>
<tr>
<td>10220642</td>
<td>70</td>
<td>2054040</td>
<td>done</td>
<td>32,E7D860D5–0576–4BBD–AA26–418</td>
<td></td>
</tr>
<tr>
<td>10220641</td>
<td>70</td>
<td>2054394</td>
<td>done</td>
<td>129,243137,131,41113,5,PHAIGV,</td>
<td></td>
</tr>
<tr>
<td>10220640</td>
<td>70</td>
<td>2054826</td>
<td>done</td>
<td>33,NOGTIEHTBUMBAVEEBVCV,4,3153</td>
<td></td>
</tr>
</tbody>
</table>
What is a “group”?

<table>
<thead>
<tr>
<th>id</th>
<th>dr</th>
<th>group_id</th>
<th>status</th>
<th>elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>10220644</td>
<td>70</td>
<td>2053940</td>
<td>done</td>
<td>17,454995,4,501446,13,1003798</td>
</tr>
<tr>
<td>10220643</td>
<td>70</td>
<td>2054825</td>
<td>done</td>
<td>34,102912,4,363502,28,37152,46</td>
</tr>
<tr>
<td>10220642</td>
<td>70</td>
<td>2054040</td>
<td>done</td>
<td>32,E7D860D5–0576–4BBD–AA26–418</td>
</tr>
<tr>
<td>10220641</td>
<td>70</td>
<td>2054394</td>
<td>done</td>
<td>129,243137,131,41113,5,PHAIGV,</td>
</tr>
<tr>
<td>10220640</td>
<td>70</td>
<td>2054826</td>
<td>done</td>
<td>33,NOGTIEHTBUMBAVEEBVCV,4,3153</td>
</tr>
</tbody>
</table>
“If needed”
Hotel Group 123

[((partner_id₁, accomodation_id₁), …, (partner_idₙ, accomodation_idₙ)) ]
Hotel Group 123
[(partner_id_1, accomodation_id_1), …, (partner_id_n, accomodation_id_n) ]
Hotel Group 123
[(partner_id₁, accomodation_id₁), …, (partner_idₙ, accomodation_idₙ)]
Hotel Group 123
[(partner_id_1, accomodation_id_1), ..., (partner_id_n, accomodation_id_n)]

Hotel Group 123 has two image groups:
[456, 203]
def deduplicate_group(all_hotel_images):
    all_images = set(all_hotel_images)
    groups = []
    while all_images:
        seed_image = all_images.pop()
        group = {seed_image}
        new_additions = {seed_image}
        while new_additions:
            me = new_additions.pop()
            for other in all_images:
                if is_same_picture(me.hashes, other.hashes):
                    group.add(other)
                    new_additions.add(other)
            all_images = all_images - group
        groups.append(group)
    return groups
def deduplicate_group(all_hotel_images):
    all_images = set(all_hotel_images)
    groups = []
    while all_images:
        seed_image = all_images.pop()
        group = {seed_image}
        new_additions = {seed_image}
        while new_additions:
            me = new_additions.pop()
            for other in all_images:
                if is_same_picture(me.hashes, other.hashes):
                    group.add(other)
                    new_additions.add(other)
            all_images = all_images - group
        groups.append(group)
    return groups
def deduplicate_group(all_hotel_images):
    all_images = set(all_hotel_images)
    groups = []
    while all_images:
        seed_image = all_images.pop()
        group = {seed_image}
        new_additions = {seed_image}
        while new_additions:
            me = new_additions.pop()
            for other in all_images:
                if is_same_picture(me hashes, other hashes):
                    group.add(other)
                    new_additions.add(other)
            all_images = all_images - group
        groups.append(group)
    return groups
def deduplicate_group(all_hotel_images):
    all_images = set(all_hotel_images)
    groups = []
    while all_images:
        seed_image = all_images.pop()
        group = {seed_image}
        new_additions = {seed_image}
        while new_additions:
            me = new_additions.pop()
            for other in all_images:
                if is_same_picture(me.hashes, other.hashes):
                    group.add(other)
                    new_additions.add(other)
            all_images = all_images - group
        groups.append(group)
    return groups
def deduplicate_group(all_hotel_images):
    all_images = set(all_hotel_images)
    groups = []
    while all_images:
        seed_image = all_images.pop()
        group = {seed_image}
        new_additions = {seed_image}
        while new_additions:
            me = new_additions.pop()
            for other in all_images:
                if is_same_picture(me.hashes, other.hashes):
                    group.add(other)
                    new_additions.add(other)
        all_images = all_images - group
    groups.append(group)
    return groups
def is_same_picture(left_hashes, right_hashes):
    for left_hash in left_hashes:
        for right_hash in right_hashes:
            ham_dist = hamdist(left_hash, right_hash)
            if ham_dist < threshold:
                return True
    return False
def is_same_picture(left_hashes, right_hashes):
    for left_hash in left_hashes:
        for right_hash in right_hashes:
            ham_dist = hamdist(left_hash, right_hash)
            if ham_dist < threshold:
                return True
    return False
How do you tune this step?

Guarantees are needed
You build a corpus.
<table>
<thead>
<tr>
<th>Measure</th>
<th>Run #1</th>
<th>Run #2</th>
<th>Run #3</th>
<th>Run #4</th>
<th>Run #5</th>
<th>Run #6</th>
<th>Run #44</th>
<th>Run #157</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General measures</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homogeneity</td>
<td>83.10%</td>
<td>63.53%</td>
<td>33.76%</td>
<td>99.59%</td>
<td>98.01%</td>
<td>92.20%</td>
<td>99.50%</td>
<td>99.50%</td>
</tr>
<tr>
<td>Completeness</td>
<td>99.02%</td>
<td>96.97%</td>
<td>97.74%</td>
<td>99.11%</td>
<td>99.15%</td>
<td>99.27%</td>
<td>99.13%</td>
<td>99.13%</td>
</tr>
<tr>
<td>V-measure</td>
<td>90.37%</td>
<td>76.77%</td>
<td>50.18%</td>
<td>99.35%</td>
<td>98.58%</td>
<td>95.60%</td>
<td>99.31%</td>
<td>99.31%</td>
</tr>
<tr>
<td>A-score</td>
<td>55.52%</td>
<td>27.61%</td>
<td>10.79%</td>
<td>95.77%</td>
<td>91.06%</td>
<td>72.16%</td>
<td>95.86%</td>
<td>95.86%</td>
</tr>
<tr>
<td><strong>Group measures</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Groups complete</td>
<td>88.92%</td>
<td>79.20%</td>
<td>86.03%</td>
<td>91.14%</td>
<td>91.35%</td>
<td>93.12%</td>
<td>91.29%</td>
<td>91.29%</td>
</tr>
<tr>
<td>Groups pure</td>
<td>83.81%</td>
<td>97.55%</td>
<td>98.32%</td>
<td>98.23%</td>
<td>93.88%</td>
<td>90.42%</td>
<td>97.82%</td>
<td>97.82%</td>
</tr>
<tr>
<td><strong>Image release</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image loss</td>
<td>29.46%</td>
<td>34.47%</td>
<td>64.13%</td>
<td>-1.80%</td>
<td>5.01%</td>
<td>18.44%</td>
<td>-1.20%</td>
<td>-1.20%</td>
</tr>
<tr>
<td>Duplicate probability</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
<td>100.00%</td>
</tr>
<tr>
<td>Average duplicates</td>
<td>4.14%</td>
<td>10.15%</td>
<td>5.13%</td>
<td>4.46%</td>
<td>3.87%</td>
<td>2.81%</td>
<td>4.26%</td>
<td>4.26%</td>
</tr>
</tbody>
</table>
Prioritisation

Prioritising → Deduplicating → Generating
Hotel Group 123
[ImageGroup\textsubscript{406}, ImageGroup\textsubscript{203}]
Hotel Group 123
[ImageGroup_{406}, ImageGroup_{203}]
Hotel Group 123
[ImageGroup_{406}, ImageGroup_{203}]
Hotel Group 123
[ImageGroup\textsubscript{406}, ImageGroup\textsubscript{203}]

“Best image”

“Best Image”

“Best order”

reaches production
What could go wrong?
What could go wrong?

<table>
<thead>
<tr>
<th>Platform</th>
<th>Room Type</th>
<th>Price</th>
<th>Deal Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Booking.com</td>
<td>Double room</td>
<td>$90</td>
<td>View deal</td>
</tr>
<tr>
<td>Expedia</td>
<td>Single room</td>
<td>$92</td>
<td>View deal</td>
</tr>
<tr>
<td>Hotels.com</td>
<td>Single room</td>
<td>$92</td>
<td>View deal</td>
</tr>
</tbody>
</table>

Good
Reviews: 431

84
What could go wrong?

- Detect features, prioritise based on that.
- Tools to manually fix data.
from PIL import ImageEnhance

def scalefit(image):
    sz = image.size
    bs = best_size(image)

    # A bit of math:
    # - if we scale to fit width, the scaling factor is: scale_width = bs[0]/sz[0]
    # - if we scale to fit height, the scaling factor is: scale_height = bs[1]/sz[1]
    # We want to scale to the smaller of them (so that the image fits in both), so we scale to width if:
    # scale_width < scale_height ⇒ bs[0]/sz[0] < bs[1]/sz[1] ⇒ bs[0]*sz[1] < bs[1]*sz[0]
    # Having it in this form means no floats are needed; all in integers.
    if bs[0] * sz[1] < bs[1] * sz[0]:
        # Scale to width
        w = bs[0]
        h = sz[1] * bs[0] / sz[0]
    else:
        # Scale to height
        w = sz[0] * bs[1] / sz[1]
        h = bs[1]

    return image.resize((w, h), "bilinear")

def contrast(image, value):
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(float(value))
from PIL import ImageEnhance

def scalefit(image):
    sz = image.size
    bs = best_size(image)

    # A bit of math:
    # - if we scale to fit width, the scaling factor is: scale_width = bs[0]/sz[0]
    # - if we scale to fit height, the scaling factor is: scale_height = bs[1]/sz[1]
    # We want to scale to the smaller of them (so that the image fits in both), so we scale to width if:
    # scale_width < scale_height ⇒ bs[0]/sz[0] < bs[1]/sz[1] ⇒ bs[0]*sz[1] < bs[1]*sz[0]
    # Having it in this form means no floats are needed; all in integers.
    if bs[0] * sz[1] < bs[1] * sz[0]:
        # Scale to width
        w = bs[0]
        h = sz[1] * bs[0] / sz[0]
    else:
        # Scale to height
        w = sz[0] * bs[1] / sz[1]
        h = bs[1]

    return image.resize((w, h), "bilinear")

def contrast(image, value):
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(float(value))
from PIL import ImageEnhance

def scalefit(image):
    sz = image.size
    bs = best_size(image)

    # A bit of math:
    # - if we scale to fit width, the scaling factor is: scale_width = bs[0]/sz[0]
    # - if we scale to fit height, the scaling factor is: scale_height = bs[1]/sz[1]
    # We want to scale to the smaller of them (so that the image fits in both), so we scale to width if:
    # scale_width < scale_height => bs[0]/sz[0] < bs[1]/sz[1] => bs[0]*sz[1] < bs[1]*sz[0]
    # Having it in this form means no floats are needed; all in integers.
    if bs[0] * sz[1] < bs[1] * sz[0]:
        # Scale to width
        w = bs[0]
        h = sz[1] * bs[0] / sz[0]
    else:
        # Scale to height
        w = sz[0] * bs[1] / sz[1]
        h = bs[1]

    return image.resize((w, h), "bilinear")

def contrast(image, value):
    enhancer = ImageEnhance.Contrast(image)
    return enhancer.enhance(float(value))
And that’s it for the pipeline
<table>
<thead>
<tr>
<th>status</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>production</td>
<td>7260896</td>
</tr>
<tr>
<td>downloaded</td>
<td>48729</td>
</tr>
<tr>
<td>filtered</td>
<td>3560235</td>
</tr>
<tr>
<td>error_download</td>
<td>1332522</td>
</tr>
<tr>
<td>available</td>
<td>16665606</td>
</tr>
<tr>
<td>not_found</td>
<td>2248861</td>
</tr>
<tr>
<td>new</td>
<td>10387</td>
</tr>
</tbody>
</table>

(7 rows)
<table>
<thead>
<tr>
<th>status</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>production</td>
<td><strong>7260896</strong></td>
</tr>
<tr>
<td>downloaded</td>
<td>48729</td>
</tr>
<tr>
<td>filtered</td>
<td>3560235</td>
</tr>
<tr>
<td>error_download</td>
<td>1332522</td>
</tr>
<tr>
<td>available</td>
<td>16665606</td>
</tr>
<tr>
<td>not_found</td>
<td>2248861</td>
</tr>
<tr>
<td>new</td>
<td>10387</td>
</tr>
</tbody>
</table>

(7 rows)
Thanks for listening

Any questions?