Image Comparison PoC

Jan 27-2025

CARING CURIOUS CREATIVE CUSTOMER FOCUS COMMITMENT TO RESULTS CONTINUOUS IMPROVEMENT

make my trip \ goibibo \ redBus

Agenda

- Exploring the Image comparison methods considered
- Mapping properties PoC based on Image comparison (Approach, Results, Wayforward)

Image Comparison PoC – update

Context:

- Flywheel currently has the capability to do automated mappings basis lat-long, name and address.
- Properties are shortlisted basis the radius of 3 km, And then basis Fuzzy logic (token sort/set ratio) comparison shall happen for name and address. Final score is generated basis distance, name and address comparison scores.
- Number of properties above score 70 is ~ 50% (About 50/100 scheduled to run have score >= 70)

Labels	% of properties above score 70	Scores	Accuracy basis our experience	Manual effort involved
very_strict	52%	90+	95%+	Low
strict	7%	85-90	80%+	Med
decent	6%	80-85	70%+	High
lenient	36%	70-80	40%+	Very High

Problem Statement:

- To increase the accuracy esp for strict, decent, lenient category
- To increase the # of properties above score 70 from current ~50% to 70%+

Image Comparison PoC – update

Approach:

- To introduce Image comparison as one more parameter with relevant weightage

Models chosen:

- Resnet, Inception, EfficientNet, VGG cv2
- These are all pretrained CNN models which are trained over imagenet and are opensource
- Output of the model is Multidimensional vector capturing info like brightness, color, objects,….
- Cosine similarity comparison shall happen basis this vector to generate the similarity score

Models tried but not chosen:

- Cv2 – Structural model basis luminance, contrast, and structure
- Clip- Needed SSH network permission to access

Image Comparison PoC – update

Pls bear with the image size

Question for the audience: Are these 2 images same?

Let's see what AI thinks…

Image Comparison PoC – update

InceptionV3 model-based output

MagnumPic3.jpg MagnumPic2.jpg

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step
Similarity score between the two images: 0.8359928727149963
```

room1.jpg room2.jpg

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
Similarity score between the two images: 0.8522713780403137
```

Pls bear with the image size

Swim1.jpg Swim2.jpg

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
Similarity score between the two images: 0.7748344540596008
```

Ginger1.jpg Ginger2.jpg

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
Similarity score between the two images: 0.6814899444580078
```

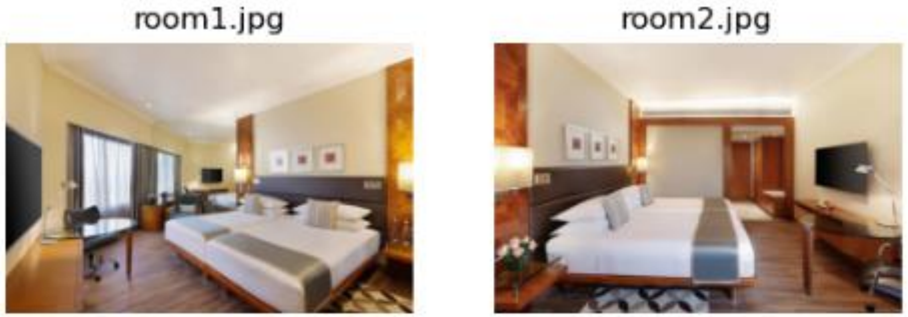
entrance1.jpg entrance2.jpg

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step
Similarity score between the two images: 0.6915736794471741
```

Image Comparison PoC – update

Resnet50 model-based output


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
Similarity score between the two images: 0.6789659261703491
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 121ms/step
Similarity score between the two images: 0.8387798070907593
```

Pls bear with the image size


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step
Similarity score between the two images: 0.9792493581771851
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
Similarity score between the two images: 0.7132307887077332
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
Similarity score between the two images: 0.6488447785377502
```

Image Comparison PoC – update

We can try any 2 images of your choice

Approach to Mapping the properties

Image Comparison PoC – update

Approach:

Identified 110 already mapped set for Bkg and MMT

Sourced the Bkg images link from s3

Programmatically,
1. converted the .png, ifif, jpg into jpeg
2. Downloaded about ~ 2200 images and resized into 2400x1600 px and stored in respective folder

Sourced the MMT images from hotstore Mongodb

Programmatically,
1. converted the .png, ifif, jpg into jpeg
2. Picked images with MMT tags only Facade, Room, Outdoors, Lobby/Common Area (These filters helped reduce images considered by **75%)**
3. Downloaded about ~ 2200 images and resized

Built the Image Comparison algo

SAMPLE	URL	MMT Tags
202109151802278551	https://r2imghtlak.mmtcdn.com/r2-mmt-htl-image/room-imgs/202109151802278551-2312-b715c87a161911eca38a0a58a9feac02.jpg	Room
202109151802278551	https://r2imghtlak.mmtcdn.com/r2-mmt-htl-image/htl-imgs/202109151802278551-eea89232161811eca5570a58a9feac02.jpg	Lobby/Common Area

Note the MMT Tags

Image Comparison PoC – update

Image Comparison algo

1. For each Mmtid, identify the corresponding Bkgid and the next 10 Bkgid and then access Images from the folders.
2. Image Comparison: For each image in the Mmtid set, compare with every image in the corresponding Bkgid set.
3. Use ResNet, Inception, VGG, and EfficientNet models for comparison.
4. Store Results with fields: Mmt_id, Bkg_id, MMT_image_id, BKG_image_id, Resnet score, Inception score, EfficientNet score, VGG score

Resnet score	Inception score	EfficientNet score	VGG score
0.46	0.52	0.42	0.48
0.47	0.58	0.36	0.38
0.57	0.65	0.16	0.47
0.99	0.98	0.99	0.99
0.67	0.58	0.55	0.70
0.57	0.65	0.60	0.58
0.40	0.61	0.30	0.38
0.70	0.71	0.45	0.66
0.68	0.69	0.55	0.67
0.89	0.76	0.81	0.83
0.61	0.61	0.61	0.72
0.53	0.54	0.45	0.52
0.51	0.56	0.33	0.37
0.45	0.61	0.14	0.48

For scores >0.8 for 3 out of 4 models	True Postive	True Negatives	False Positives	False Negatives
Accuracy basis my sampling checks out of 1500 combinations	100%	100%	0%	0%

14 combinations snippet *(For my ref- MMTid: 202402120104099115 comparison with Bkgid: 11465256 (Mapped))*

File name: comparison_results4

Image Comparison PoC – update

Note on Latency:

1. For 3 MMTid x 30 Bkgid ~ 1500 image comparison took about 6 hrs (Need more time to run on HP basic laptop :p)
2. Converting, filtering the MMT tags and downloading for 110 properties took about 2 hrs

Way Forward:

- We can store every vector in vector db (using knn) so that we can retrieve it as and when needed/queried (Atlas db vector search with cosine/Euclidean similarity powered by Mongodb)
- Further filter basis MMT Tags and compare efficiently. Or/And generate tags for BKG images and then filter further.
- Run on powerful processors on aws cloud
- Add the image comparison check wherever the score is less than 95 for prev algo
- Set the threshold such that wherever the score of atleast 3 models greater than 0.8 for any of the comparison, then the property is 100% match OR
- Set the threshold such that wherever the score of atleast 3 models greater than 0.8 for 2 of the comparison, then the property is 100% match

Thanks

Result:

- Wherever the score of atleast 3 models greater than 0.8, then the property is 100% match

Appendix

http://confluence.mmt.com/display/FLYW/Automated+Mappings

Core Data Provider for Products - Holidays package pricing

Flywheel is now powering data to Holidays LOB for pricing of train tickets across Europe for ~670 sectors

Problem Statement:

- Prices of train tickets were manually updated quarterly or bi-yearly with high markups to prevent losses
- This used to often result in overpriced Europe packages
- In peak months due to surge pricing, we used to absorb losses as our tickets were priced low

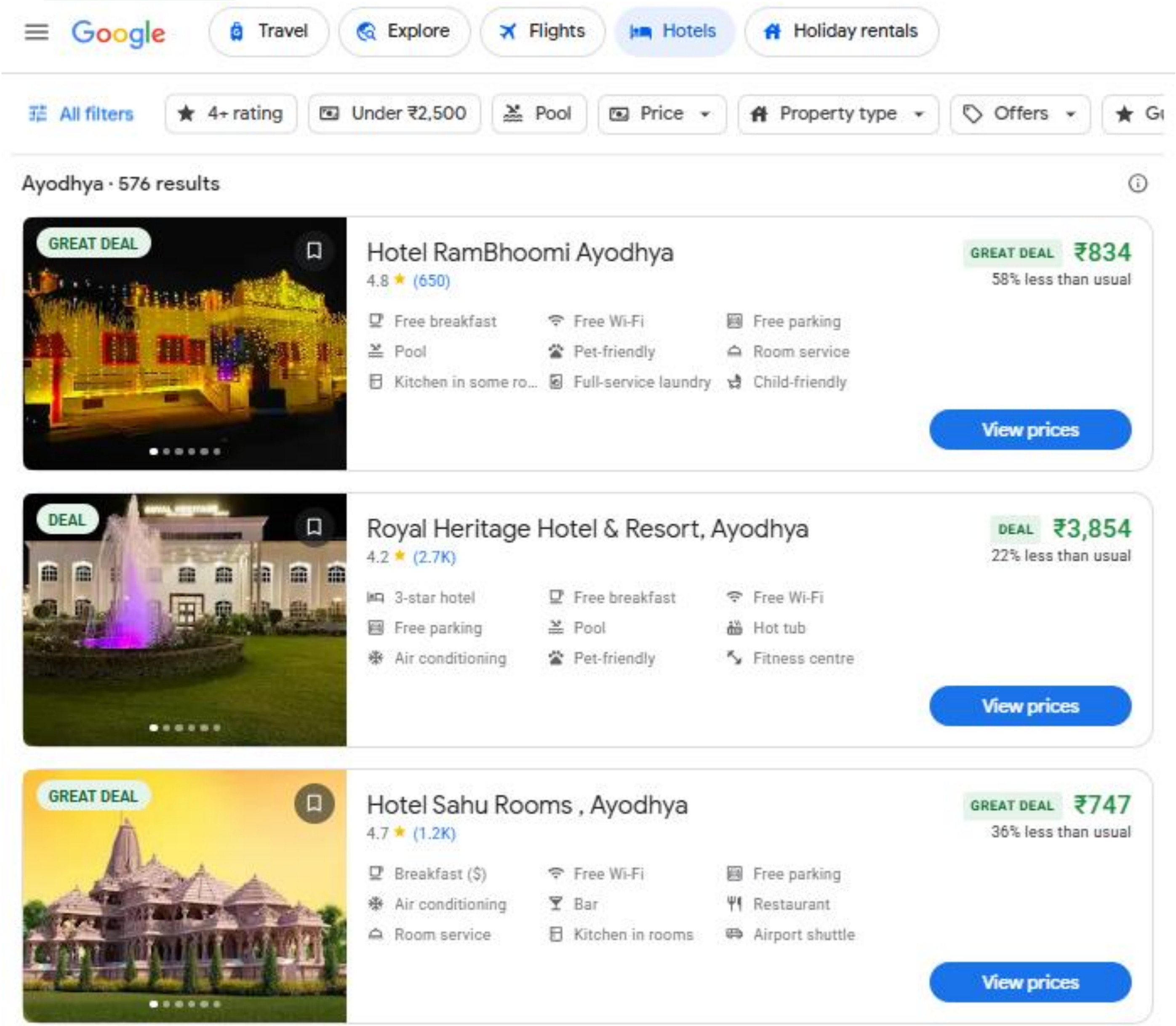
What we did:

Started crawling Raileurope website which has extensive coverage of all sectors with rail pricing

Current situation:

- We are now crawling 670 locations to and fro twice a week
- Specific prices at First/Second class, time, cancellation policies and pax level are being considered for rail prices in packages
- Because of the latest data and automatic integration, package prices are on par with the actuals.

Google Listing crawls to identify the supply contracting opportunities for Tier 2 & 3 cities



Context:

- Potential supply contracting opportunity needed to be identified through Google listing crawls for Tier 2 and Tier 3 cities to start with.

Capability developed:

- Conducted a PoC with the integration of third-party tools and APIs to crawl Google Hotel Listings and assess the scalability of the process.

Result of PoC:

- 6 cities are successfully crawled
- Data captured: hotel name, listing price, ratings, lat/long, address and phone number

Way forward:

- Scaling to 20 cities/week
- Enhancing tech capability to crawl Tier 1 cities