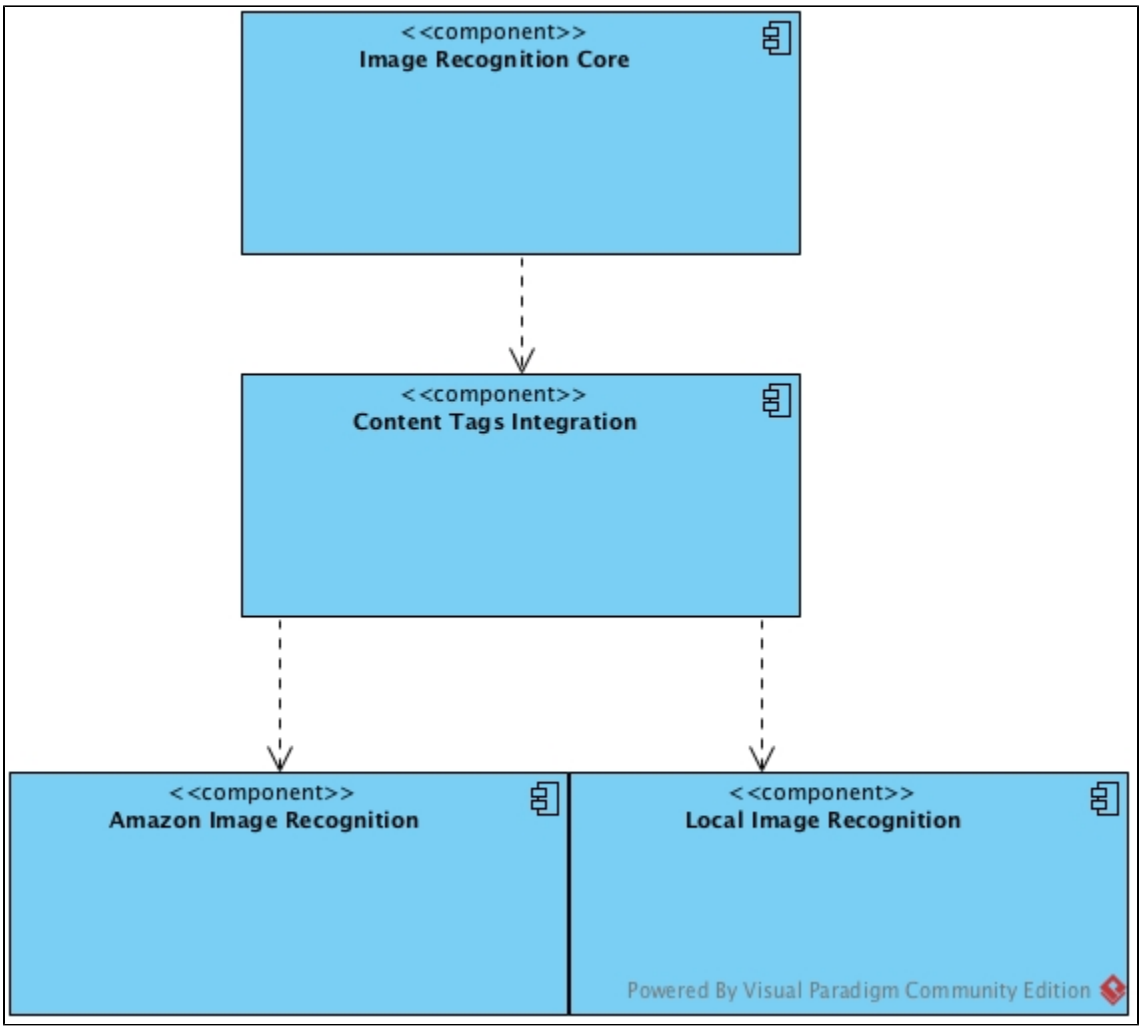
Image recognition

Introduction

We want our customers to search for images within Magnolia, this can be done either with Periscope or any other search method. However, searching for images is not possible unless you know the exact name of the image.

This is not the case most of time since you have to manually recognise (by human) your images before you upload them and tag them accordingly (tagging via file name or asset name). That being said, with the rise of Machine learning the particular problem to recognise images manually is not the case anymore. Nowadays this can be done by machine which are quite powerful as well as external services like Amazon Rekognition or Google Vision API. There is also a possibility to do this with a local pre-trained neural network and hence one doesn't need to send their images to external services but rather have their own service locally.

Figure below shows the relationship between the modules which are described in this paper.



Related sub modules

Image Recognition Core

This is the core module of whole image recognition services. Essentially it brings ImageRecognitionService interface which is currently implemented in *LocalImageRecognitionService* and *AmazonImageRecognition* service. If one wants to implement another image recognition service or integrate with another third party service, this is the interface to implement.

```
 * Takes image bytes as parameter and returns a {@link Collection collection}
 * of {@link ImageLabel Image label}s as output.
 *
 * <p>
 *     Returns empty collection for the cases below:
 *     <li>Upon exception</li>
 *     <li>Image couldn't be recognised</li>
 * </p>
 */
Collection<ImageLabel> recognise(byte[] imageBytes);
```

In addition, it also brings *ImageRecognitionModule* which is a Magnolia module class.

This module queries the given workspace(DAM) in the module startup phase in order to tag all untagged images. To do so, it simply delegates to *ImageRecognitionService* and uses *TagManager* to tag the given labels from *ImageRecognitionService.*

Content Tags Integration

In general, this module provides following functionality to tag images:

- Provides necessary decorations to enable content-tags module for assets app
- Has dedicated action to delegate to *ImageRecognitionService* to recognise newly uploaded images or it could be used as a separate action to trigger *ImageRecognitionService* on demand.

Image Recognition Service Implementations

Local Image Recognition

This module brings a pre-trained model which we use to recognise images locally. It's limited 1000 classified tags and that is the upper limit initial implementation. See future work/ideas section to understand why/how it can be improved further.

Essentially, this module brings *LocalImageRecognitionService* s TODO:hereee

Amazon Image Recognition

This module provides integration to Amazon Rekognition via *AmazonImageRecognitionService.*

Required modules

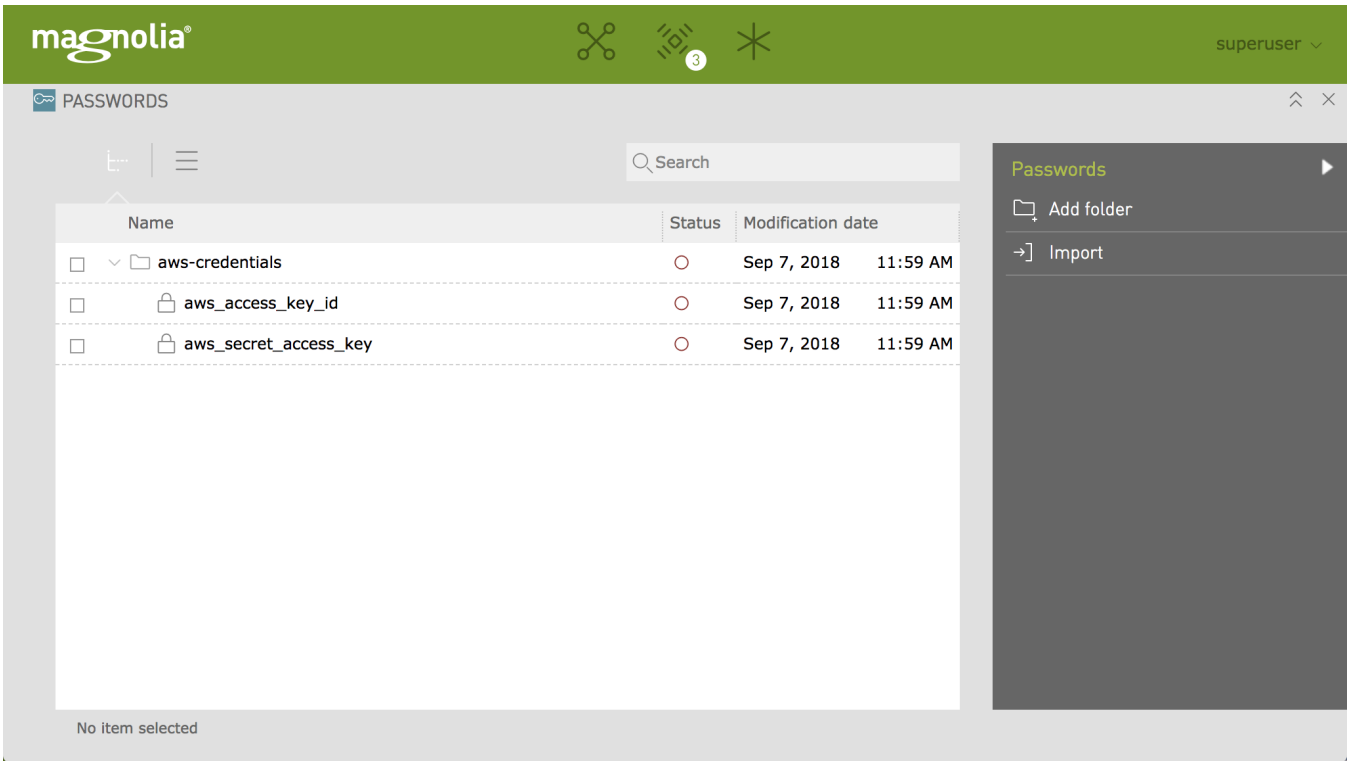
- amazon-image-recognition – https://nexus.magnolia-cms.com/service/local/repositories/magnolia.enterprise.snapshots/content/info/magnolia/ai/image/amazon-image-recognition/1.0-SNAPSHOT/amazon-image-recognition-1.0-20180918.001831-42.jar
- aws-java-sdk-core – http://central.maven.org/maven2/com/amazonaws/aws-java-sdk-core/1.11.299/aws-java-sdk-core-1.11.299.jar
- aws-java-sdk-rekognition – http://central.maven.org/maven2/com/amazonaws/aws-java-sdk-rekognition/1.11.299/aws-java-sdk-rekognition-1.11.299.jar

In addition, Amazon Rekognition currently supports only three different image formats: png, jpg, and jpeg.

AWS Credentials

In order to use the service, one has to provide aws_access_key_id and aws_secret_access_key. We have integrated our Password module to store those keys.

One simply has to put the keys under folder 'aws-credentials' and create those two keys via the Password app.



Grant AWS account permission to use Rekognition

TODO: Describe how to grant permissions in IAM console.

Service Configuration

Image below show the default configuration which is bootstrapped to JCR when the module is installed. Once bootstrapped this configuration can be changed via Configuration app under amazon-image-recognition/config or alternatively one can change it before it's being bootstrapped.

```
config:
  regionName: your_aws_region_name
  maxLabels: 10
  minConfidence: 50
  supportedFormats:
    png: png
    jpg: jpg
    jpeg: jpeg
```

- *regionName:* Has to be set into a real region name such as `eu-west-1` and also note that the particular region should support Amazon Rekognition, currently only couple of regions are supporting the service
- *maxLabels*: How many maximum labels will be returned by the service
- *minConfidence*: Service gives us confidence rates between 0 and 100, this number here determines what would be minimum acceptable confidence rate
- *supportedFormats*: Service currently supports couple of formats however we made this configurable in case it changes in the future

AWS regions where Rekognition is available

> *Q: In which AWS regions is Amazon Rekognition available?*

> *Amazon Rekognition Image is currently available in the US East (Northern Virginia), US West (Oregon), US East (Ohio) , EU (Ireland), Asia Pacific (Tokyo), Asia Pacific (Sydney), Asia Pacific (Mumbai), Asia Pacific (Seoul), and AWS GovCloud (US) regions. Amazon Rekognition Video is available in US East (Northern Virginia), US West (Oregon), US East (Ohio) , EU (Ireland), Asia Pacific (Tokyo), Asia Pacific (Sydney), Asia Pacific (Mumbai), Asia Pacific (Seoul), and AWS GovCloud (US) regions. Amazon Rekognition Video real-time streaming is only available in US East (Northern Virginia), US West (Oregon), EU (Ireland), Asia Pacific (Tokyo), Asia Pacific (Mumbai), and Asia Pacific (Seoul) regions.*

Source: https://aws.amazon.com/rekognition/faqs/

Future Ideas/Work

Open issues

Key	Summary
IMGREC-99	Image recognition function doesn't work on mgnl demo (6.2.2)
IMGREC-98	Minor problems when applying decorations in dx-core logs
IMGREC-89	Use correct field type for content tags integration
IMGREC-88	JPG image is not recognised by local or amazon
IMGREC-85	Report image recognition usage metrics
IMGREC-83	Disable the Run recognition action if noop is set for image recognisers
IMGREC-66	Convert Semaphore based Producer/Consumer logic to Queues
IMGREC-65	Observe and recognise all configured workspaces
IMGREC-49	ImgRec Later
IMGREC-38	Newly uploaded images in Asset App show tags automatically when present
IMGREC-35	Local ImgRec. Include hypernyms when tagging
IMGREC-19	Re-run image recognition on demand
IMGREC-16	Publish new image tags to public, if applicable

13 issues

Concurrency in ImageRecognitionModule or Service

IMGREC-5 - Getting issue details... STATUS

Currently we rely on a main application thread to recognise images, this is plain wrong for two reasons:

- Will take forever when on *ImageRecognitionModule* startup if there are plenty of images
- This execution blocks the main application thread and hence has to wait no matter what in the startup even it's for 100 untagged images

Essentially the problem is not the fact that it takes time but rather it blocks the main thread. Hence we should ideally have a queue based (or similar) execution mechanism in order to do this job in concurrent manner in different threds.

That way we will be able to not block the main thread and also benefit from concurreny since we are not blocked by the service especially if we are using an external service to do the job.

Improve Local Image Recognition

MGNLPER-17 - Getting issue details... STATUS

Use labels that represent objects in typical marketing images

Local image recognition is limited to the ImageNet 1000 synsets (synonym sets). Labels in this collection do not represent typical marketing imagery. Animals ("African elephant", "hyena", "weasel") account one third of the labels while common marketing subjects such "computer", "person" or "shoe" are missing. This means that a neural network pre-trained on Imagenet 1000 classifications does not recognize common marketing subjects.

Examples of common marketing images and tags from AWS:

		
pancake, toast, French-toast, bread, food	human, clothing, people, person, boot	human, reading, people, pe

Actions to take:

- Use a set of labels that better represent subjects in typical marketing images.
  - Core Wordnet is a list of 5000 core word senses in WordNet.
  - Contains most frequently used word senses followed by some manual filtering and adjustment.
  - http://wordnetcode.princeton.edu/standoff-files/core-wordnet.txt
  - Excerpt:

> *n [shoe%1:06:00::] [shoe] footwear*
> *n [shoe%1:06:01::] [horseshoe] shoe for horses*
> *n [shop%1:06:00::] [shop] store*
> *n [shop%1:06:01::] [shop] workshop*
> *n [shopping%1:04:00::] [shopping]*
> *n [short%1:06:01::] [short circuit] short*
> *n [shortage%1:26:00::] [shortage] dearth, famine*
> *n [shorts%1:06:00::] [shorts] short pants*
> *...*

- Use nouns only. Remove verbs, adjectives and adverbs from Core Wordnet 5000.
- Retrieve WordNet IDs for the synsets. http://imagenet.stanford.edu/download-API
- Generate the labels JSON file. https://git.magnolia-cms.com/projects/ENTERPRISE/repos/image-recognition/browse/local-image-recognition/src/main/resources/info.magnolia.ai.image.local/imageNetLabels.json
- Re-train the neural network on the new set of labels.

This way we should have a local solution which performs better/more accurately when it comes to recognizing marketing images.

Further ideas

We could have dedicated network for customers and train the network on their manually assigned tags.