Recommending photo filters via image content in a REST API

Joshua Bridge
14032908
joshua.m.bridge@stu.mmu.ac.uk

March 29, 2018

Abstract

To be completed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Digital image processing

Image processing deals with the analysis and manipulation of image data. An example of image processing is the use of digital signal processing. This involves converting analog sensory data from a digital camera sensor into a computer-interpretable format with minimal data loss from external sources such as noise and distortion.

1.1.1 Bitmaps

Before you are able to analyse an image, you must first represent the data in a way that it can be interpreted by a computer, and a human. One basic form of doing this is via a bitmap image. A bitmap - as its name implies - is a simple spacial mapping of values (bits) along a horizontal axis (x) and vertical axis (y). Using a greyscale image as an example, a bitmap representation of this would contain a number of values (or 'pixels'), the number of which is equal to the product of the sizes of the x and y axis. Therefore an image of size 200 x 200 would contain 40,000 pixels. Each of these pixels contains an integer value representing brightness, typically ranging from 0 - 255 (the total value range

of an 8-bit integer), '0' being completely black, '255' being completely white.

A colour image follows a very similar format, except now each pixel contains three brightness values instead of one. Each of these values map to the brightness of the colours (or 'channels') red, green and blue - in that order. Therefore a pixel with values (0,255,0) would be entirely green and a pixel with values (0,0,255) would be entirely blue. It should be noted that when these colours are displayed on a computer screen their colour values are additive (i.e. they can mix together to form a different, brighter colour). A pixel with values (0,255,255) would therefore represent cyan, and finally a pixel with values (255,255,255) would represent white.

1.1.2 Image processing tasks

There are several methods of improving the results of image analysis, one of which is to run an image processing algorithm against it. Generally this is to make the image clearer by sharpening it or removing noise - these are often referred to as low-level processing methods (Sonka et al. 2014). While these methods are often applied to make analysis by a computer a lot easier, they also can be used to increase the 'clarity' or the percieved 'beauty' of an image when viewed by a human.

Sharpening

For an image to be captured, it must first enter through some kind of lens which refracts the incoming light into a 'focal point' onto which some kind of light-sensitive surface is placed such as a digital image sensor. In order for an object to be perfectly 'in focus' it must be at the optimal distance from the lens - an area known as the 'focal plane'. If the subject of a photograph is not near enough to the focal plane (either in front of it or behind it) then the subject will appear to blur.

1.2 Computer vision

Computer vision involves modelling the human vision system in such a way that a computer can interpret abstract visual data.

Chapter 2

Literature Review

In order to understand the background of the technology involved in this project, it is necessary to complete a significant review into the current and past literature. This will help form a basis of knowledge from which the future development and analysis of the proposed system will utilise and build upon. This research will be vital in order to make use of the most optimal technology for any given problem.

The following review will be an investigation into the current knowledge of the major components of the proposed system, which are as follows:

- The recommendation system.
- The image-filter generation system.

2.1 Recommendation systems

As defined in Ricci et al. (2011), recommending content is a problem which involves attempting to predict what a user may desire at any given time when using a system. In the 'internet era' applications of this are far-reaching, such as recommending products on Amazon.com Linden et al. (2003). Recommendation systems are a popular topic due to the "abundance of practical applications that help users to deal with information overload

and provide personalized recommendations, content, and services to them" (Adomavicius & Tuzhilin 2005).

Jannach et al. (2010) describe three different methods for giving recommendations: Collaborative, Collaborative, Content-based, and Hybrid.

2.1.1 Collaborative recommendation

When using an online streaming platform that serves video content such as Netflix or YouTube, a user will visit their site looking for something to watch. The problem faced by these sites is trying to find content that the user hasn't seen before, but will align with the user's interests enough to make them want to watch it (Davidson et al. 2010), (Gomez-Uribe & Hunt 2016). The information needed to carry out such a prediction can include the semantic value of the entire users history of interaction with the system (often referred to as *metadata* (Duval et al. 2002)), from which preference can be extrapolated. If there is not enough metadata about the user from which to extrapolate preference with reasonable certainty, then preference can be inferred from other users of the system - especially users with a similar predicted preference.

2.1.2 Content-based recommendation

There can sometimes be barriers to using collaborative recommendation, such as when the 'cold-start problem' is present. This problem describes when a user has just signed up to a website, it will not yet have enough information / user history to recommend them content (Schein et al. 2002). In this situation content-based recommendation could be used more effectively to build up more reliable recommendations (Lops et al. 2011). There are different kinds of content-based recommendation, as it can be applied to many different kinds of content. For example a system may work differently if the content is user-generated as opposed to internally or computer generated, where the information

would likely be provided already. In a system with user-generated content, the information about that content would have to be inferred, or provided by the user.

Feature extraction for content analysis

When content is user-generated and without a definition provided alongside it, its definition must be derived. For textual data this can be done by analysing the words and extrapolating a commonality between keywords to find a prevailing subject (Sanderson & Croft 1999). This type of process is often referred to as 'Feature extraction' (Guyon & Elisseeff 2006). When analysing raw data such as an image file or an audio file, this becomes a much more difficult task as you must first find a way of deciphering the content before deriving the subject or meaning.

Deep learning has recently become a common way of deciphering this content, including for image, audio, and video content types (Coates et al. 2011), (Ciregan et al. 2012), (Lee et al. 2009), (Mobahi et al. 2009). Spotify has implemented this kind of deep learning to determine genre types from raw audio data - see section 2.1.3. This can be applied to image data also, for example in the context of a search algorithm, it would bring the most relevant images by content to the top of the results (Yee et al. 2003). This works similarly to how a recommender system will select the most relevant items for the user to see.

Google's ImageNet has proven to be a very reliable system for image classification (Krizhevsky et al. 2012). As it has a very high accuracy on most image types, it would be reasonably effective for providing a basis of feature extraction for use in a recommender system. In the system proposed in this report, scene classification along with prevailing colours in the image would be an ideal input into the recommendation system for determining optimal image filters.

2.1.3 Hybrid recommendation

Generally, hybrid recommendation applies both of the previous methods together in order to draw up a more comprehensive list of recommendations, in order to avoid the individual weaknesses of both types.

Spotify - Discover Weekly

Discover Weekly is a playlist made every week for every user of Spotify. Instead of getting humans to curate a playlist for each and every user, an algorithm is used to try and find a set of songs which it predicts the user will like, but hasn't listened to before. As described by Popper (2015) it does this by mixing collaborative and content-based recommendation. The first is done by analysing all playlists on spotify and determining their similarity to playlists made by the user. The next step is looking for songs in those playlists that the user in question hasn't heard yet.

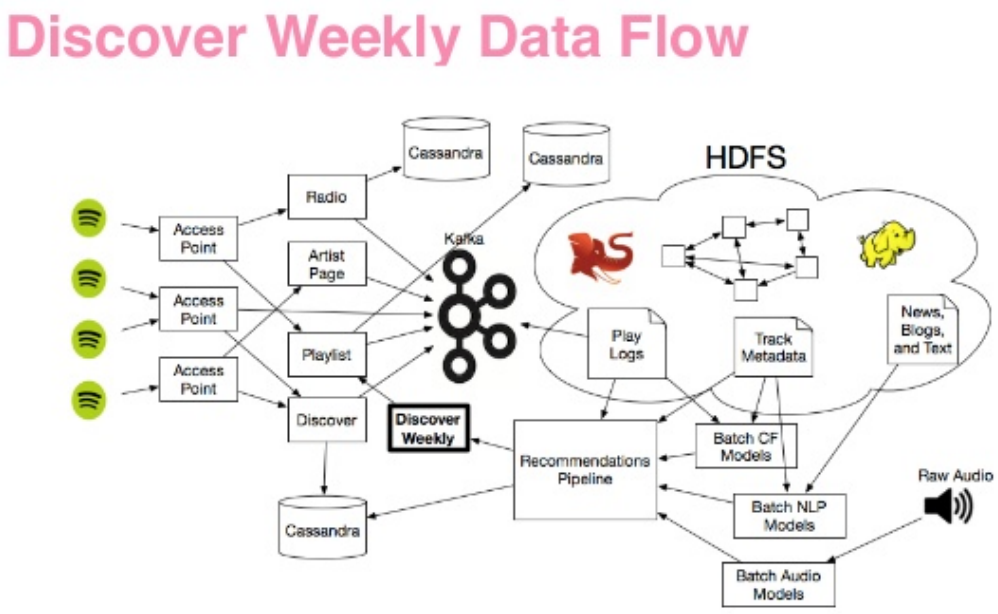

Figure 2.1: System component architecture of Spotify's 'Discover Weekly' (Johnson 2015)

The second, more complicated part of this process is the content-based recommendation. Spotify's algorithms actually analyse the music data in several different ways to determine a users 'taste'. The first is by defining the genres a user listens to, which it does by scanning music blogs for what genre a certain piece of music is described as being. Using this as a search, spotify can then look for other music in that genre to recommend. The arguably more complicated part of this process is the auditory analysis of raw music data. As shown in figure 2.1 the raw audio is converted into 'batch audio models' where they can be used as a basis for recommendations.

2.2 Image-filter generation

There is little to no research into the area of developing new photo filters, likely due to the fact that filters can be very subjective in terms of what effect they have on an image. Furthermore the relation of that effect to how many people choose that filter for a certain type of photograph would require a large data-set. However it is possible to determine the most common types of photographs posted online, which will guide the process of creating filter templates.

2.2.1 Instagram

In order to create image filters that users would want to apply to their photographs, an assessment must first be made about that the type of photos people tend to post online. 'Instagram' is a popular online photo-sharing platform that attracts over 700 million users (Instagram 2017). An investigation by Hu et al. (2014) revealed that the types of photos posted on Instagram could be roughly categorised into eight types: "self-portraits, friends, activities, captioned photos (pictures with embedded text), food, gadgets, fashion, and pets". The study clarifies that 'self-portraits' (or 'selfies') made up 24.2% of the images posted, and 'friends' made up 22.4%, totalling 46.6% of the images analysed. From this

it can be hypothesized that creating filters which are complimentary for skin-tones would be a contributing factor to success when developing image-filters.

Hu et al. (2014) however did not study a wide proportion of the Instagram user-base, in fact they only closely studied 50 users, out of an initial 95,343 'unique seed users', from a user-base of more than 150 million when the study was taken place. Furthermore the study only analyses pictures posted online, to Instagram, with public profiles, and from users which had "at least 30 friends, 30 followers, and had posted at least 60 photos". The system proposed in this report would not be limited in terms of where the resulting image would be posted, and has no requirement that the image be posted to an online website. Therefore the findings in the study should only be used as a loose guideline for popular image categories when defining image-filter styles.

2.2.2 Personalisation of image enhancement

There has been some work done on the mapping of image enhancement styles to personal taste, although the field appears to be still in early stages. However, two studies in particular have heavily contributed to the development in this field. Kang et al. (2010) found that when given a set of images, participants would deviate into different styles when asked to choose between two options of enhancement (e.g. lighter or darker). Furthermore, many participants preferred their own enhancement when compared to an averaged enhancement, suggesting that personalised enhancements would be a valuable tool.

Using the work done by Kang et al. (2010) as a basis, Caicedo et al. (2011) developed a method of personalised image enhancement that focused on putting users into groups (or 'clusters') that had similar tastes within image enhancement. Therefore when attempting to apply personalised enhancement on a per-user basis, instead of determining a specific users preference from scratch, the user's preference would only have to be mapped to a pre-defined group. Using this method the authors suggest that personalisation of image enhancement could be scale-able to large amounts of users.

2.3 Conclusion

Posting images online is fast becoming one of the biggest trends in recent history, as evidenced by Instagram's user-base expanding to over 700 million users (Instagram 2017). While not much public research seems to have been carried out into the reasons why users pick certain styles of image-filters before posting, it certainly appears viable to aid them in making that process easier. The findings of this review suggest that feature extraction could be a useful tool in developing recommendations for filters. This would be carried out in the form of image classification, most commonly to determine whether a person is present in a photograph which would thereby influence the algorithm's decision of what filters to recommend to the user for a certain image.

The findings of this report also suggest (mostly from Kang et al. (2010) and Caicedo et al. (2011)) that the field of personalised image enhancements (or filters) is a relatively new field of study, therefore there is a lot of room for collecting new findings by extending upon previous works in a new implementation.

Chapter 3

Design

3.1 Requirements

The application proposed in this report would be best utilised on multiple platforms & devices, as the service it would offer is platform-independent (i.e. all you need is an image file). For this reason the primary focus will be making an online web-app, which connects to a seperate REST (Fielding & Taylor 2000) API which performs the main functions of the application, without the clutter of the front-end mixing in with it. This would also allow other apps and services to utilise this API and integrate it with other apps, such as on social media apps when sharing an image.

When designing a user-facing website and an API, there are certain considerations that have to be met, such as data security and ease of use. The following features define what the final product should achieve:

- A web-based user interface.
- A user-history (metadata) tracking mechanism.
- A RESTful API which can be utilised by any application.

Following these features, these are some of the things that should be taken into consideration when implementing these features:

- The application & API should be able to handle a high-volume amount of users.
- It should not store any more information about users than it has to.
- It should make this data available at a users request.
- It should maintain a reasonable level of security to prevent user-data from being stolen.
- It should not store any user images after the user has finished using the application.

3.2 Front-end

3.2.1 React

3.3 API

The API is to be written in Python (`https://www.python.org`) due to its high suitability for dealing with image data, web-access and recommender algorithms. The application is able to be completely self-contained as it can perform all of the functions required in a single project.

3.3.1 Web Framework

A simple framework for handling HTTP access to the API endpoints is critical when creating an application which is meant to be able to handle high-volume traffic. Flask (`http://flask.pocoo.org`) performs exactly this purpose as it is a very lightweight framework which provides very easy methods for transferring data in different formats such as JSON.

Other frameworks could be considered for this purpose, however few match Flask in terms of low complexity and ease-of-use. Django (`https://www.djangoproject.com`) is another very common web framework for Python which comes with many useful features such as a built-in user system. Many of the features, however, are not required within this project and would only increase the complexity of the code in the application. When applying the LSD methodology (Poppendieck & Poppendieck 2003) this would come under the heading of eliminating waste - for example spending less time learning a complex framework and more time actually developing core features & meeting requirements. Therefore Flask is a good foundation upon which to build a RESTful API that is easy to use & easy to develop.

3.3.2 Task Queueing

3.3.3 Database Storage

A database is a crucial component within this application, where there are multiple requirements which make use of one. The choice of database technology falls into a few categories such as SQL/NoSQL (or Relational/Non-Relational). When considering the type of data which needed to be stored by the application it was decided that a NoSQL database would be best for multiple reasons.

Performance

NoSQL databases are often easier to host than SQL databases due to their ability to scale horizontally (Cattell 2011) rather than just vertically (i.e. adding more servers to handle load rather than upgrading from one machine to another higher power machine). This is applicable in the context of this project as a requirement is that the application should be able to scale.

Data Structures

3.3.4 NumPy

3.3.5 SciKit

KMeans

3.3.6 Pillow

3.3.7 Recommendation

3.4 Deployment

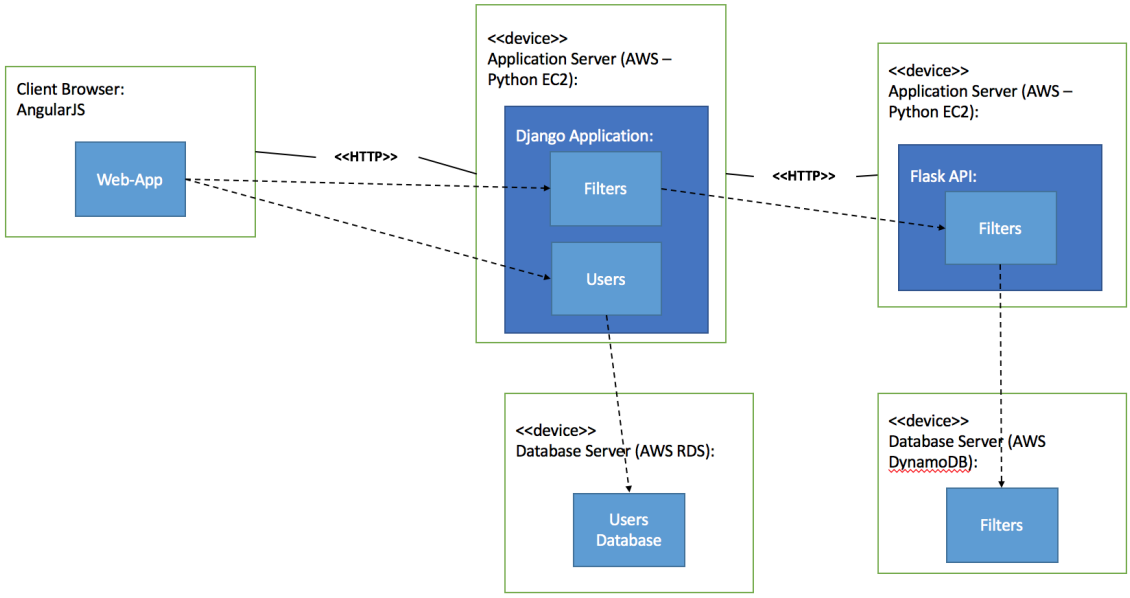

Figure 3.1: Component deployment diagram

Fig. 3.1 defines the component layout for the entire workflow of the application proposed in this report.

3.4.1 API - AWS

3.4.2 Front-end - Github Pages

Chapter 4

Implementation

In this chapter a detailed description of the product implementation will be laid out, including explanations for each choice of technology and how they enabled the product to achieve its goal. The product can essentially be split up into two major components - a RESTful API & a web-based front-end. Using an API is an important factor in the implementation as it defines how the flow of the client application will work. As a result, technologies must be chosen which easily make use of this type of application-flow such as a framework which integrates well with AJAX calls - a common way of connecting to APIs. The front-end development is a crucial part of the implementation as it acts as a testing interface for the API. If the API is not structured well then creating a front-end around it is much more difficult, meaning any future developers wishing to use it might be dissuaded by its complexity and end up choosing a different service. Developing both at the same time makes it much easier to see how clients will access the service, therefore making it much easier to remove any complexity it may contain while the service is still being built.

Chapter 5

Evaluation

Chapter 6

Conclusion

Bibliography

Adomavicius, G. & Tuzhilin, A. (2005), 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE transactions on knowledge and data engineering* **17**(6), 734–749.

Caicedo, J. C., Kapoor, A. & Kang, S. B. (2011), Collaborative personalization of image enhancement, *in* 'Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on', IEEE, pp. 249–256.

Cattell, R. (2011), 'Scalable sql and nosql data stores', *Acm Sigmod Record* **39**(4), 12–27.

Ciregan, D., Meier, U. & Schmidhuber, J. (2012), Multi-column deep neural networks for image classification, *in* 'Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on', IEEE, pp. 3642–3649.

Coates, A., Ng, A. & Lee, H. (2011), An analysis of single-layer networks in unsupervised feature learning, *in* 'Proceedings of the fourteenth international conference on artificial intelligence and statistics', pp. 215–223.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. et al. (2010), The youtube video recommendation system, *in* 'Proceedings of the fourth ACM conference on Recommender systems', ACM, pp. 293–296.

Duval, E., Hodgins, W., Sutton, S. & Weibel, S. L. (2002), 'Metadata principles and practicalities', *D-lib Magazine* **8**(4), 1082–9873.

Fielding, R. T. & Taylor, R. N. (2000), *Architectural styles and the design of network-based software architectures*, University of California, Irvine Doctoral dissertation.

Gomez-Uribe, C. A. & Hunt, N. (2016), 'The netflix recommender system: Algorithms, business value, and innovation', *ACM Transactions on Management Information Systems (TMIS)* **6**(4), 13.

Guyon, I. & Elisseeff, A. (2006), 'An introduction to feature extraction', *Feature extraction* pp. 1–25.

Hu, Y., Manikonda, L., Kambhampati, S. et al. (2014), What we instagram: A first analysis of instagram photo content and user types., *in* 'Icwsm'.

Instagram (2017), '700 million'.
URL: *https://instagram-press.com/blog/2017/04/26/700-million/*

Jannach, D., Zanker, M., Felfernig, A. & Friedrich, G. (2010), *Recommender systems: an introduction*, Cambridge University Press.

Johnson, C. (2015), 'From idea to execution: Spotify's discover weekly'.
URL: *https://www.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly/*

Kang, S. B., Kapoor, A. & Lischinski, D. (2010), Personalization of image enhancement, *in* 'Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on', IEEE, pp. 1799–1806.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* 'Advances in neural information processing systems', pp. 1097–1105.

Lee, H., Pham, P., Largman, Y. & Ng, A. Y. (2009), Unsupervised feature learning for audio classification using convolutional deep belief networks, *in* 'Advances in neural information processing systems', pp. 1096–1104.

Linden, G., Smith, B. & York, J. (2003), 'Amazon. com recommendations: Item-to-item collaborative filtering', *IEEE Internet computing* **7**(1), 76–80.

Lops, P., De Gemmis, M. & Semeraro, G. (2011), Content-based recommender systems: State of the art and trends, *in* 'Recommender systems handbook', Springer, pp. 73–105.

Mobahi, H., Collobert, R. & Weston, J. (2009), Deep learning from temporal coherence in video, *in* 'Proceedings of the 26th Annual International Conference on Machine Learning', ACM, pp. 737–744.

Poppendieck, M. & Poppendieck, T. (2003), *Lean software development: an agile toolkit*, Addison-Wesley.

Popper, B. (2015), 'Tastemaker: How spotify's discover weekly cracked human curation at internet scale'.
URL: *https://www.theverge.com/2015/9/30/9416579/spotify-discover-weekly-online-music-curation-interview*

Ricci, F., Rokach, L. & Shapira, B. (2011), Introduction to recommender systems handbook, *in* 'Recommender systems handbook', Springer, pp. 1–35.

Sanderson, M. & Croft, B. (1999), Deriving concept hierarchies from text, *in* 'Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 206–213.

Schein, A. I., Popescul, A., Ungar, L. H. & Pennock, D. M. (2002), Methods and metrics for cold-start recommendations, *in* 'Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval', ACM, pp. 253–260.

Sonka, M., Hlavac, V. & Boyle, R. (2014), *Image processing, analysis, and machine vision*, Cengage Learning.

Yee, K.-P., Swearingen, K., Li, K. & Hearst, M. (2003), Faceted metadata for image search and browsing, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 401–408.