REPORT

Development of content-aware social graphs

- project deliverable 4.8

Authors:	Panagiotis Galopoulos, Chryssanthi Iakovidou, Symeon Papadopoulos, Yiannis Kompatsiaris, Kristina Kapanova, Barbara Guidi, Andrea Michienzi, Laura Ricci
Confidentiality:	Public

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825585.

HELIOS Commercial Exploitation Requirements Gathering	
project name HELIOS	**Grant agreement #** 825585
Authors Panagiotis Galopoulos (CERTH), Chryssanthi Iakovidou (CERTH), Symeon Papadopoulos (CERTH), Yiannis Kompatsiaris (CERTH), Kristina Kapanova (TCD), Barbara Guidi (UNIPI), Andrea Michienzi (UNIPI), Laura Ricci (UNIPI)	**Pages** 75
Reviewers Vanessa Clemente Nuñez (WLI), Javier Marín Morales (UPV)	
Keywords user profiling, multimedia analysis, image classification, content-based matching	**Deliverable identification code** D4.8
Summary In this deliverable we explore the development of content-based features to enrich the HELIOS Heterogeneous Social Graph (HSG), and in that way provide support for content-aware user profiling and matching services. To this end, we aggregate the image collections of the HELIOS users and extract semantic information in three complementary ways; the first focuses on analyzing the user interests, the second constructs a user representation in a self-supervised way and the third relies on processing pieces of text extracted from the images. We employ state of the art CNN models to process the user images for the tasks of classification, construction of user embeddings with Deep Metric Learning and text extraction with an LSTM-based Optical Character Recognition engine. Special care is taken to design models that are suitable for mobile deployment and respect the privacy of the users performing all required calculations on the device. The developed user profiling and matching services are able to operate on different spatio-temporal contexts using an attention mechanism. In that way, our framework supports the extraction of conditional content aware profiles that capture multiple aspects of the user interests.	
Confidentiality	Public
25/06/2020 Written by Panagiotis Galopoulos	
Project Coordinator's Contact address Ville Ollikainen, ville.ollikainen@vtt.fi, +358 400 841116	
Distribution HELIOS project partners, subcontractors and the Project Officer	

Table of Contents

List of figures

List of tables

List of Acronyms

Term	Description
API	Application Programming Interface
AUC	Area Under Curve
CEN	Contextual Ego Network
CNN	Convolutional Neural Network
COCO	Common Objects in COntext
DCG	Discounted Cumulative Gain
DML	Deep Metric Learning
DNN	Deep Neural Network
GLP	Group constrained Label Propagation
HSG	Heterogeneous Social Graph
IDCG	Ideal Discounted Cumulative Gain
LDA	Latent Dirichlet Allocation
LP	Label Propagation
LSTM	Long Short Term Memory
LP	Label Propagation
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
OCR	Optical Character Recognition
OOV	Out Of Vocabulary
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine
TF-ITF	Term Frequency–Inverse Term Frequency
UML	Unified Modeling Language
WIDER	Web Image Dataset for Event Recognition
YFCC	Yahoo Flickr Creative Commons

1 Introduction

The goal of this deliverable is the aggregation of multimedia content produced by the HELIOS users and its semantic spatio-temporal analysis with the goal to identify matching users in terms of interests and enrich the Heterogeneous Social Graph (HSG), described in the D4.2.

Each user leverages the Contextual Ego Network (CEN), as the local view of a node in the HSG, in order to store information concerning their contexts and relationships with others. The current CEN is able to store information concerning the relationships, such as the strength of a tie. We are interested in analyzing HELIOS user nodes in order to enrich the CEN with personal interest information extracted from the consumed and exchanged content. In this deliverable, we present methods for creating a user profile primarily based on personal photo collections stored on device. Special care has been taken so that all computations run locally on the device, thus eliminating the need for the users' data to ever be transferred to an external server in accordance with the HELIOS strict privacy policies and decentralized nature described in deliverables D3.1 and D3.2. The outcome of this deliverable is an extension module to the HELIOS ecosystem that aims to enable content awareness to other HELIOS modules and services by embedding the constructed user profiles in the Contextual Ego Network. In this deliverable we will, also, discuss one such service, namely user matching, but the profiles could also be leveraged by the Social Graph Mining Module for interaction and friend recommendations, a topic relevant to Task 4.3 and further discussed in D4.3.

This deliverable expands on the HELIOS sources of user information beyond the social graph's structure and the social activity patterns to the users' multimedia content. A significant amount of work has been done in the research community involving the topics of user profiling and matching through visual content which we will review on Section 2 and in the following sections we discuss the less explored area of designing such methods for on-device execution, presenting two different avenues of approaching the user profiling task, one through interest concepts and one through the notion of user similarity, while also touching upon methods based on analyzing the textual content extracted from images.

Another topic of interest to HELIOS is that of contexts and mainly spatio-temporal contexts, as described in D4.1. The HELIOS core operates on multiple contexts and the users maintain different ego networks for each one. In this way the user's social network becomes a multidimensional entity to better reflect the real life social network of the user. Content awareness could be multidimensional too, matching the various contexts the user already has, this is further discussed in Section 6.1, where we build a user profile conditioned on spatio-temporal restrictions.

After this introduction, in Section 2 we discuss the main research challenges that we faced, provide an overview of the literature for different profiling schemes and finally present some preliminaries on the evaluation metrics used later in the deliverable as well as two brief overviews, one on the subject of Deep Metric Learning and one on the subject of executing deep learning models on mobile devices.

In Section 3 we present our proposed method for user profiling through predefined interest categories. We start with a discussion of how we collected the data from the Pinterest platform and move on to analyze the hierarchical classifier that we developed. Then we evaluate our model on a separate, manually labeled dataset of Pinterest users and finally we conclude the section with a discussion of user matching techniques.

An alternative method of user profiling that does not depend on predefined categories but rather learns a user representation through data is presented in Section 4. We use a subset of the YFCC100m [1] dataset and annotate it with user similarity data based on externally provided autotags[1]. The model is based on Deep Metric Learning and learns to represent similar users closely while pulling dissimilar ones further apart. We conclude by presenting some implementation details.

The focus of Section 5 is the textual content present in images for the purpose of constructing the user profiles. We first extract the text from the images using the open source Tesseract OCR engine. Then the extracted text is pre-processed - stop words are removed, the text is lemmatized and stemmed. Finally, bag-of-words and LDA approaches are used to infer user profiles and interest.

In Section 6 we introduce conditional content aware profiling and matching that can operate on different spatio-temporal contexts, described in D4.1. We also evaluate user matching using a dataset of Pinterest users annotated with follower-followee relationships.

In Section 7 we discuss an integration of profiling in social graph mining, in which the structure of the social graph capturing user actions can help improve their profiles by modeling that users engaging in social actions influence each other towards similar interests. In that section we also analyse how interest- and preference- related information can be combined to match users based on both their content and social behavior.

Then, in Section 8 we discuss some practical aspects about the mobile deployment of the developed models as well as some privacy considerations.

Finally, in Section 9 we conclude the deliverable with some final remarks and briefly discuss some opportunities for future work.

[1] Autotags are categories generated automatically for each image, following a deep learning scheme. Those tags serve as descriptors of the visual concepts that are present in each photo. Autotags along with the related exploitation strategy, are presented in more depth in Section 4.2.

2 Research Challenges and Related Work

In this section we provide research challenges and related work towards profiling approaches. These can be broadly categorized in two classes, the ones that depend on some predefined categories and through those construct the user profiles and the ones that can adapt the user representations usually with some form of unsupervised or self-supervised method. In the following we discuss the literature involving both as well as some hybrid approaches.

2.1 Research Challenges

Defining the Scope and Form of User Profiles

Developing a content-aware social graph requires reasoning about the high level semantic meaning of the users' multimedia content. Understanding a user's images at the level of recognizing simple objects and scenes is not enough to provide an assessment of the user's profile. However, these are the tasks that the research community has mostly focused on, providing large datasets and optimized CNN architectures, and as such no off-the-shelf solutions are suitable for eliciting interest user profiles from visual content.

To create content aware profiles, we firstly need to define the nature of a user's content. In general, it would be a vector that somehow captures the semantic content of the user's images relevant to some concepts. These concepts can be predefined as we do in Section 3 through the notion of interest categories or can be left to be discovered by the model, allowing it to freely mold the user representation space. The latter approach is the subject of Section 4, where we construct a similarity measure from their image auto tags and employ deep metric learning (DML) to learn the user representations that best reflect the given similarities.

Obtaining Proper Datasets

While appropriately defining the nature of user profiles is essential, there is a need for proper datasets to train new deep learning models (Convolutional Neural Networks - CNNs in our case). In fact, this turned out to be a big challenge and we inevitably had to compromise as our need for large, high quality datasets of user images annotated with relevant information for the task of semantic profiling could not be accommodated with today's publicly available resources.

These circumstances led us to create our own dataset[2] for the training and evaluation purposes of this deliverable. The dataset can be thought to contain four parts; the first one was created from Pinterest based on the interest categories defined in Section 3.2 and aims to facilitate the training of interest classifiers. The second consists of 12 randomly selected, hand-labeled Pinterest users and serves as a test for the model developed on the previous part. The third one is based on a subset of the YFCC100m dataset [1] where the images were grouped by user and each user was labeled with a feature vector calculated from the auto tags of their images, as described in Section 4.2. Last but not least, for the purpose of evaluating the quality of matching as well as a comparison for the different profiling techniques, we collected the images of 422 Pinterest users annotated with follower-followee relationships.

[2] The dataset is published in Zenodo under the name PIPD2020 - Pinterest Interests Profiling Dataset 2020. https://zenodo.org/record/3895162#.XvMmTpZRUrg

On-device Execution

Another issue that significantly complicates matters is the requirement of the HELIOS app to be executed exclusively on the users' devices and without communication with any external servers. Even though computer vision has one of the most active research communities and is evolving at a staggering rate, unfortunately, the mobile and embedded device deep learning ecosystem has yet to enter a mature state. As matters stand now, we have to be considerate of the devices' limited memory and computational resources as well as be aware that deep learning frameworks have only a subset of their operations supported for mobile inference. For our model to be successful, it is imperative to be designed from the start around these restrictions. This is the reason that throughout the deliverable we have opted for simple but efficient models with straightforward execution order as well as computationally efficient versions of performant but otherwise bulky CNNs. On the positive side, things are rapidly progressing with more operations becoming supported and many challenges being alleviated.

2.2 Profiling with Predefined Categories

Using predefined categories to construct user profiles is the most straightforward way that has the benefits of simplicity and interpretability. The user profiles in this case hold a specific meaning that is humanly intuitive and can be used to develop insights to how the algorithm works which could be beneficial not only for the developers but also for the end users. App users tend to distrust obscure features that they have no intuition about how they work. An interpretable model can help demystify the provided services and alleviate adoption problems. There are many possible ways to choose these fixed categories and we will discuss some of them in the next subsections.

Object and Scene Categories

One of the simplest predefined categories are those that are used for object and scene recognition tasks. Object recognition is a well-defined and heavily researched task with large publicly available datasets[3] as well as pretrained models[4] and the same is also true for scene recognition[5]. It is, therefore, reasonable to take advantage of these sophisticated and well performant models to capture a user's profile based on the detected objects and recognized scenes in their photo collection.

Indeed, a recent paper [2] combined object and scene recognition models to summarize a user's images with a merged counter of the corresponding categories detected in their images as well as their videos. This counter served as the user's profile. The paper is also concerned about mobile execution and the models that they used were chosen according to their trade-off of efficiency and performance. The authors, however, go even a step further and propose a hybrid application that would conditionally send some images to be processed by more accurate and power hungry server models. Although this is an interesting idea, it is not applicable in the context of HELIOS because of the strict data privacy policy demanding on-device execution.

Another paper [3] also used object detection to produce user profiles, but interestingly it did so only as an intermediate step. The actual profiles were also based on predefined categories, but this time the authors chose to use the 34 different domains encapsulated by the BabelNet[6] synset. After crawling images of Flickr users along with their tags, the authors fed these tags to

[3] Common Objects in COntext (COCO) http://cocodataset.org
[4] https://github.com/tensorflow/models/tree/master/research/object_detection
[5] http://places2.csail.mit.edu/
[6] https://babelnet.org/

BabelNet in order to annotate these images according to the output category. From these annotations the ground truth user profile was created for each user. The image based user profile was then created by running the object detection model on the user's images and then matching the objects with the appropriate BabelNet categories.

An important issue of the object detection oriented strategies is whether identifying common objects is enough to provide an adequate semantic description of an image, the aggregation of which should translate to a user profile that captures the users' characteristics. Unfortunately, the works mentioned previously do not provide strong enough evidence that this is the case and this matches our intuition as well; we believe that even though object recognition is nowadays streamlined and easy to apply, it would be more meaningful to try and find more meaningful descriptions that represent higher-level semantic information.

Event Categories

Profiling users based on the detected events in their images was among the first fields that we explored. Given the lack of a formal event definition, we resort to a definition by example; by events we refer to happenings like a wedding or a funeral, a concert, a sports competition, a family reunion or a picnic. These carry higher semantic meaning than any single object or scene detected and have the potential to be more relevant to the construction of user profiles. In addition, some research concerning the event recognition problem has been made, but none to the best of our knowledge used such events to construct user profiles and thus seemed an interesting research opportunity.

Reviewing the event recognition literature, we found that one of the most commonly used datasets is WIDER[7], which consists of approximately 60,000 images spanning 61 different event categories and was produced by Xiong et. al. [4]. In their paper, the authors proposed the combination of object, human and face detection techniques in a custom deep architecture, but the results were not that encouraging as their best model achieved only 42.4% classification accuracy.

However, several papers have been published since then, proposing new techniques and improving the model's accuracy. In 2017, Wang et. al. [5] proposed the use of two networks running in parallel, one initialized from an object detection task and one from scene recognition and experimented with various configurations for computing the loss function. However, for inference their input pipeline involved 54 crops of a single image which would entail 108 forward passes, which is computationally intensive and not suitable in the context of the HELIOS mobile app.

Also in 2017, Ahmad et. al. [6] proposed a technique based on detecting saliency regions of images. Their training pipeline consisted of first extracting candidate regions from an image, selecting the most salient ones by crowdsourcing, extracting features with a CNN and creating a bag of features whose label would be the same as the original image's label. They would then proceed to do multiple instance learning. The obvious drawback is the need for crowdsourcing during training which seems to be an integral part of the process. However, the reported 55% accuracy on the WIDER dataset seems quite a step up from the original WIDER paper.

In 2019, Lakhdar et. al. [7] took a quite different approach by using probabilistic topic models for the event recognition task. Probabilistic topic models are frequently used in the field of Natural

[7] Web Image Dataset for Event Recognition (WIDER) http://yjxiong.me/event_recog/WIDER/

Language Processing (NLP) and have been developed quite extensively. Perhaps the simplest probabilistic topic model is Latent Dirichlet Allocation introduced by Blei et. al. [8]. LDA hypothesizes the existence of a generative process from which all the documents are produced. This generative process involves first, for each document the selection of a distribution of possible topics and then the selection of each word in the document according to the per-topic word distributions. This process can be encoded concisely with a probabilistic graph model. They report a classification accuracy score of 58.1% on the WIDER dataset marking the state of the art value. However, inference with probabilistic topic models is not straightforward and does not correspond to a simple forward pass, but rather to an iterative optimization procedure. This made us wary about its potential in an environment where execution is time constrained and computational resources are limited. This concludes our brief review of the available event recognition models, but there is yet another issue that needs to further discussed, that is how suitable event categories are to describe user profiles.

To make matters more concrete we present in Table 1 the 61 WIDER event categories. We have highlighted some of the categories in red, as we feel these do not contribute much to describing a person and they do not fit well in the context of user profiling. This fact along with the lack of fit of the most successful event recognition models with the HELIOS requirements and some not so promising initial experiments, led us to further investigate strategies for user profiling and in particular interest-based profiling, which is the subject of the next section.

Table 1. WIDER dataset categories, the highlighted categories are emphasized because they do not fit well in the context of user profiling.

Parade	Gymnastics	Baseball	Soldier Firing	Worker Laborer	Celebration Or Party
Handshaking	Swimming	Ceremony	Soldier Patrol	Award Ceremony	Sports Coach Trainer
Football	Greeting	Concerts	Soldier Drilling	Stock Market	Parachutist Paratrooper
Riot	Meeting	Couple	Tennis	Basketball	Students Schoolkids
Dancing	Group	Soccer	Sports Fan	Demonstration	People Driving Car
Car Accident	Interview	Festival	Car Racing	Family Group	People Marching
Funeral	Traffic	Picnic	Surgeons	Waiter Waitress	Election Campain
Cheering	Running	Shoppers	Spa	Ice Skating	Matador Bullfighter
Voter	Angler	Hockey	Row Boat	Street Battle	Press Conference
Dresses	Rescue	Raid	Aerobics	Balloonist	Photographers
Jockey					

Interest Categories

Compared to the previously discussed object, scene and event categories, interests is the concept that is semantically closest to the users and thus a promising alternative. In fact this is the method we opted out for and the specifics of our interest-based model will be fleshed out in Section 3.3. In this section we are going to present the similar work of Quanzeng et al. [9].

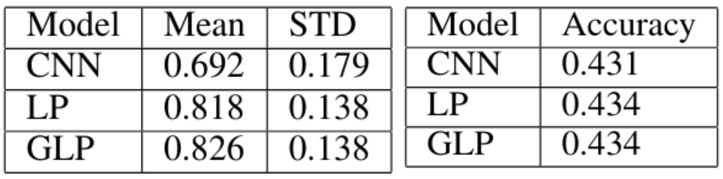
Quanzeng et al. trained a model for user interests profiling based on data scraped from Pinterest. The data consisted of approximately 1.6 million images from 748 users, labeled with one of the 32 interest categories as defined by Pinterest as well as an extra “other” category. The categories appear on Table 2.

Table 2. 32 Pinterest categories.

Animals and pets	Films, Music & Books	Home decor	Quotes
Architecture	Food and drink	Humor	Science and nature
Art	Gardening	Illustrations and posters	Sports
Cars and motorcycles	Geek	Kids and parenting	Tattoos
Celebrities	Hair and beauty	Men’s fashion	Technology
Design	Health and fitness	Outdoors	Travel
DIY and crafts	History	Photography	Weddings
Education	Holidays and events	Products	Women’s fashion

Pinterest’s model is presented in Section 3.2 but summarizing briefly, Pinterest organizes images, called pins, in pinboards. The labels of the images in the dataset were inherited from the user assigned label of the pinboard they belonged to. Unfortunately, however, nowadays, this information seems to not be available and this methodology cannot be applied anymore.

For their model, they initially trained a CNN on the classification task achieving 43% accuracy. However, motivated by the fact that their final goal is to predict a user-level distribution of interests instead of a per image prediction, they used a second label propagation step. The label propagation algorithm was run on the graph induced by the similarity of the extracted images’ features and guided by the similarity of the different interest categories. Although the algorithm did not help improve significantly the classification accuracy, it did increase the mean Normalized Discounted Cumulative Gain (NDCG) from 0.69 to 0.83, as Figure 1 shows. NDCG is a metric for the evaluation of ranked retrieval tasks, that is in our case the ranking of user interests, that we will describe in Section 2.3.



Model	Mean	STD
CNN	0.692	0.179
LP	0.818	0.138
GLP	0.826	0.138

Model	Accuracy
CNN	0.431
LP	0.434
GLP	0.434

Figure 1. Evaluation of the baseline CNN model along with the proposed Label Propagation (LP) and Group constrained Label Propagation (GLP) models from [9]. The table on the left shows the mean and standard deviation of the NDCG scores for the test set users, while the table on the right, the classification accuracy.

Some of our issues against this work is that we believe that including all the Pinterest categories is not appropriate as some of them, marked with red in Table 2, make only sense in the context of Pinterest and are not useful for the task of user profiling. For example, although illustrations and posters are pleasant and tempting for people to click, they do not define a meaningful interest category and including it can potentially do more harm than good. The same applies to other categories such as products, while some others are too broad to be included in the set such as education and celebrities or inappropriate for semantic image analysis, for example quotes. It is also the case that the classification accuracy is rather low and the fact that it is not improved through the label propagation algorithm is worrisome, especially considering the complications of it running on mobile devices.

Profiling with Machine Discovered Representations

While profiling with predefined categories has the benefits of being more straightforward and interpretable, the semantics that it can capture are inevitably limited by the fixed nature of the categories. Although the initial categories we define make intuitive sense and seem relevant to the task, it is very likely that there are even more discriminating dimensions that would lead to more nuanced user profiles. Finding those dimensions is not an easy task, but we will present our proposal in Section 4, while in the remainder of this section we will review some of the relevant literature.

The HKUST-NIE Social Media Lab[8] of the Hong Kong university of science and technology has been actively researching the topic of connection discovery in social networks at least since 2015. Connection discovery is meant in a similar way to what we refer to as user matching, but in most cases in order to find matches or connections a user representation is first constructed, which can serve the purpose of a user profile, and a measure of similarity is then calculated, typically being cosine similarity.

Cheung et. al. [10] proposed a Bag of Features Tagging (BoFT) method that would generate user representations based on tagging the users' images with labels produced from the BoFT approach. Specifically, their pipeline included first a feature extraction step using traditional computer vision method, namely a Harris Affine detector, a Maximally Stable Extremal Regions detector and a Kadir Brady saliency detector, followed by the creation of a visual codebook. The visual codewords were then clustered and the image labels were calculated from the cluster distribution of its extracted codewords. Although the employed feature extraction techniques are nowadays much less popular, the idea of creating representations based on an unsupervised clustering method is interesting and we will see it again later on. Finally, the connections between the users were calculated with the cosine similarity of the user profiles, but a probabilistic method for user recommendation was also explored.

A variation of the above method was proposed in [11]. The clustering and tagging of the images remained the same but instead of computing codewords they clustered directly the extracted features, but again did not use any of the powerful CNN extractors but rather more traditional algorithms, namely Scale Invariant Feature Transform (SIFT), GIST and Color Transformation. This time no mention to the probabilistic recommendation scheme was made but rather only cosine similarity was used.

A more novel paper was written around the same time in Li et. al. [11] where they proposed the use of topic modeling to discover user connections. They trained their model on 200k images crawled from Flickr and annotated with follower-followee relationships according to the following generative procedure; first, for each user the topic distribution is sampled and according to those for each image a topic assignment is sampled. Then an image is sampled from a Gaussian distribution whose parameters depend on the assigned topics and last, for a pair of users a binary link indicator is sampled. There are many ways to estimate the unknown parameters of a probabilistic topic model, all of which require the appropriate formulation of the

[8] http://smedia.ust.hk/

probabilistic equations. Although an interesting approach, for reasons analysed previously in this section, probabilistic topic models are not the best fit for HELIOS requirements and unfortunately, this method does not produce the intermediate user profiles we are interested in.

The last and most recent paper [12] is the most similar to our proposed model in Section 4. Utilizing an impressive amount of social media data collected over the years, this time the feature extraction process was driven by Deep Metric Learning (DML) techniques. A brief overview of DML is presented in Section 2.4, but the main idea is that a CNN is trained with a loss that motivates related users to become closer in the feature space while dissimilar users are driven apart. After this model has been trained the approach the authors follow is very similar to their previous works. Unfortunately, however, our attempts to contact them were fruitless and their website[9] that claims to hold data form millions of images seems to not be maintained as of the time of this writing.

User Profiling through Post Textual Analysis and Topic Detection

Using text analysis to detect user profiles has been an active area of research in the past few years. To represent user interests, previous work either used bag-of-words, Topic Models or bag-of-concepts approaches [13]. However, it is difficult to directly estimate the interests of social media users from social media data because their posts do not contain any category information. To mitigate the problem, a variety of approaches have been proposed, including mapping the content of texts in social media into categories of a news corpus. This is advantageous since news media contain additional category information which is being organised by experts into predefined categories. This is effective when categorisation of posts can be identified by distinguishable keywords. However, data sparseness caused by short posts, abbreviations, and infrequently used topical terms in social media still needs to be addressed to accurately capture users' interests. But for terms that occur infrequently, one cannot utilize categorisation information. To mitigate this, external knowledge bases can be used, contributing to an increase in accuracy for estimating users' interests. Such approaches are unsupervised. A supervised approach combining convolutional neural networks and recurrent neural networks to classify text messages of social media into predefined categories, in which data manually labeled by annotators are utilized in a training process has shown some promising results, but it requires manual annotation, as well as significant compute resources [14]. User interests across multiple online social networks (Twitter and Pinterest) was studied in Ottoni et al [15], where in Pinterest user interests are identified by the images a user pins and repins, while in Twitter, user interest topics are identified from tweets of the users. While most of the existing works have focused on extracting explicit interests, there are certain works that focus on inferring implicit interests of the users.

Since users in social networks can freely publish posts without any restriction, their posts are usually unstructured and include a nearly unlimited set of terms. Thus, a user profile can be modeled as a bag-of-words feature vector generated from the user's posts with term frequency–inverse term frequency (TF-IDF). Combination of TF-IDF with bag-of-words approach improves categorisation of posts and finding user interests, but unfortunately, the distances between posts about the same topic might be large if the posts contain different words. Therefore Topic modeling approaches such as Latent Dirichlet Allocation (LDA) have been used for user profiling.

[9] http://smedia.ust.hk/connections/index.html

Topic modeling based on social media analytics facilitates understanding the reactions and conversations between people in online communities. Topic models are a prominent way to find hidden structures in massive information streams. Generally combining TF-IDF, LDA, with shallow word-embedding learning techniques can improve user profiling through textual analysis.

2.3 Evaluation Metrics

A work cannot be complete without proper evaluation of its results, as it is the evaluation process that will highlight the strong points of a model and at the same time reveal its weaknesses. It is thus important to select the proper evaluation metrics that will provide us with valuable insights about the model behavior. Different tasks require different metrics and so we will describe each metric in the context of the task that it was used for.

The user profiles that we will later create are based on the classification of the users' images in interest categories. The classification task is one of the most well studied ones and for our purposes the accuracy metric is well suited to assess the model's success. The accuracy of a model is defined as the percentage of the correctly classified images and typically comes in two flavours, top-1 accuracy and top-5 accuracy. Top-1 accuracy refers to the percentage of the images for which our model assigned the most probability to the correct class, whereas the top-5 accuracy is a more relaxed measure that counts an image as correctly classified if its true label is according to our model within the 5 most probable categories.

While the evaluation of the classification task is a straightforward process, the situation is quite different for the evaluation of the created user profiles. The user profiles that we will construct will at their core be probability distributions over interest categories and as such a naive metric could be based on distance of the probability distributions. The problem of calculating the distance between two probability distributions is a fundamental one and a lot of metrics have been proposed, the most common one being the Kullback-Leibler (KL) divergence. However, we could argue that a metric like this would be too punishing for our model. The user profiles we have at our disposal are mere approximations that we have devised and absolute numbers by themselves do not carry that much meaning; saying for instance that one's interests are 60 percent sports will always be somewhat arbitrary no matter how it is calculated. What is more important and we can measure with greater confidence is the relative ordering of one's interests; claiming that someone is 60 percent into sports and 25 into fashion does seem questionable in an absolute sense but the statement that this person is more interested in sports than fashion can be more confidently supported. Therefore, we argue that the ranking of the users' interests is what we should be mostly concerned with and not give too much of a meaning to the exact shape of the probability distribution. Therefore, our internal evaluation process could first focus on the question of whether we identified correctly the most prominent interest of the user, then on whether we also managed to identify the second one and so on.

Given that for our evaluation purposes we consider ranking as the most important aspect, we turn to the information retrieval literature. Generally, information retrieval is the activity of obtaining information, usually pertaining to a specific query, from a collection of resources. The results of the information retrieval process can be considered either ranked or unranked based on whether their order matters. In our case the query would be a user's interests, the information pool would be the available interests categories and we would consider our output ranked as it is critical for our application to retrieve the user's interests in order of preference.

Fortunately, a lot of evaluation methods have been proposed for this information retrieval task and we will briefly review some of them. Fundamental metrics for the unranked retrieval task are the True Positive, True Negative, False Positive and False Negative Rates, which are defined as

$$
\begin{aligned}
TPR &= \frac{TP}{TP+FN},\\
TNR &= \frac{TN}{TN+FP},\\
FPR &= \frac{FP}{FP+TP},\\
FNR &= \frac{FN}{FN+TP},
\end{aligned}
$$

where FN is the number of relevant but not retrieved entities, TP is the number of relevant entities that have been retrieved, FP is the number of non-relevant retrieved entities and TN is the non-relevant entities that have not been retrieved. These quantities are visually demonstrated in Figure 2.

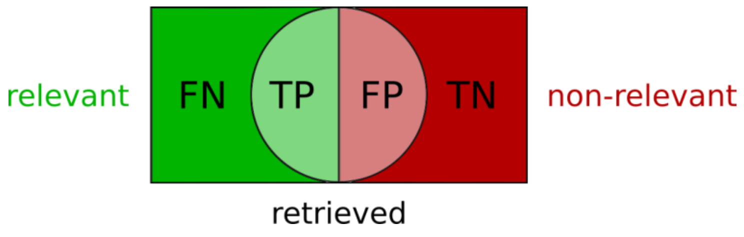


Figure 2. The relation between False Negatives, True Positives, False Positives and True Negatives. The green square corresponds to the relevant documents, the circle to the retrieved and the red square to the non-relevant.

TPR is also called sensitivity or Recall (R) and is commonly used in combination with Precision (P) which is defined as

$$P = \frac{TP}{TP+FP}$$

Precision describes the percentage of the retrieved documents that are relevant, whereas recall refers to the percentage of the relevant documents that are retrieved. Since it is trivial to maximize each of these two metrics on their own, for example by retrieving not and all entities respectively, only their joint maximization can indicate a successful model. This can be achieved with, the F-measure, which performs a harmonic averaging between the “degree of soundness” (precision) and the “degree of completeness” (recall),

$$F_\beta = (1+\beta^2)\frac{P * R}{\beta^2 * P + R}$$

A common shortcoming of precision, recall and the F-measure, are set based measures and thus suitable only for evaluating unranked retrieval. To introduce the concept of ranking we could plot the precision and recall values relative to each rank position forming the precision-recall curve. Precision or recall at rank k means that we calculate the corresponding measure considering only the set of the k first results. Having drawn the precision-recall curve, we can measure the area under its curve through a measure called average Precision[10] (AP) as

$$AP = \int_0^1 P(R)dR.$$

The higher (i.e. closer to 1) the average precision is, the better our model's response to the query. Averaging over a set of queries Q lets us define the mean average precision (MAP) as

$$MAP(Q) = \frac{1}{|Q|}\sum_{i=1}^{|Q|} AP(Q),$$

which is a measure of the overall performance of our model at the ranked retrieval task.

Another commonly used measure for the evaluation of ranked retrieval stems from the Receiver-Operating-Characteristic (ROC) curve. The ROC curve is shaped by plotting TPR against FPR. The area under the ROC curve (AUC) is also a measure of the performance of the model.

Discounted Cumulative Gain (DCG)[11] is yet another measure of ranking quality, which uses a graded relevance scale of the results. The idea is that one rewards retrieved results according to their relevance to the query, but also discounts the reward based on how far down the retrieved list they are. So, naturally, the first step is to define the relevance of each result, which generally is dependent on the problem at hand. In the user profiling case, for example, we will consider as a measure of relevance the probability of the interest category in the ground truth profile, that is

$$relevance(i) \ = p_{gt}(i)$$

where p_{gt} is the ground truth user profile. Next the discounting formula needs to be also defined; a reasonable choice is

$$\mathrm{discount}(i) = \frac{1}{\log_2 \max(i,2)},\ i = 1, \ldots, k\,,$$

where k is the number of included results. Having defined the relevance and the discount of the results, the DCG is given by

$$\mathrm{dcg}_k = \sum_{i=1}^{k} \frac{\mathrm{relevance}(i)}{\mathrm{discount}(i)}\,.$$

Nevertheless, the length of the results for each query may vary and to make the measure comparable across different queries it is commonly normalized. The Normalized Discounted

10 https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Average_precision
11 https://en.wikipedia.org/wiki/Discounted_cumulative_gain

Cumulative Gain (NDCG) is computed by dividing the DCG by the Ideal DCG (IDCG) which is equal to the DCG corresponding to the ideal response,

$$\mathrm{ndcg}_k = \frac{\mathrm{dcg}_k}{\mathrm{idcg}_k} .$$

2.4 Deep Metric Learning

Problem Setting

Metric Learning is the research field that focuses on learning a distance function to measure the similarity between data samples. Early works attempt to learn a Mahalanobis distance metric through a convex optimization problem [16]. However, recent works combine Metric with Deep Learning for the learning of a distance function through the training of a Deep Neural Network (DNN). The main objective of such works is to approximate an embedding function that maps samples in a feature space where relevant samples are closer than irrelevant ones. Relevance between samples may be determined based on any arbitrary semantic relation, e.g., two images of the same concept. A DNN can approximate such an embedding function through a training scheme that penalizes the violation of the samples' ordering. More precisely, to formulate this process, a similarity between two arbitrary samples $x, y \in \Re^n$ has to be defined. The most prevalent measure used to quantify similarity between samples is Euclidean distance in the embedding space, which is defined as:

$$D(x,y) = \left|\left|f_\theta(x) - f_\theta(y)\right|\right|_2$$

where $f_\theta : \Re^n \to \Re^d$ is the embedding function that maps a sample $x \in \Re^n$ to a point in a feature space $\Re^d$ and is modeled through a DNN with θ parameters, and $D(\cdot,\cdot)$ is the Euclidean distance in this space. Any deep learning architecture can be selected for the implementation of the DNN, which is usually adapted based on the underlying problem. The ultimate goal is the training of a DNN network that generates embedding representations which result in small distances between relevant samples and large distances between irrelevant ones.

Training setup and loss function
Several DML setups have been proposed in the literature for the training of the DNN. The main differences between the training schemes concern the employed loss function, which dictates the number of data samples used for the loss calculation. A common practice in the DML applications is the use of Siamese Networks (SNs) combined with the contrastive loss functions [17]. SNs consist of two identical networks that share weights. The contrastive loss functions utilize two data samples and their pairwise label to calculate the loss. It minimizes the distance between relevant sample pairs and maximizes the distance between irrelevant ones. Another popular setup employs Triplet Networks (TNs), composed of three networks that share weights, similar to the SN. They are usually combined with the triplet loss function [18] that considers the relative distance between data points in order to compute the loss; or the angular loss [19] that takes into account the angles of the triplet triangles formed based on the data samples. Other recent works employ more than three samples to calculate the loss. Quadruplet [20] and N-pair loss [21] extents TNs by applying the core idea of relative distances to more than three samples at a time. Finally, lifted structural loss [22] exploit the pairwise distances between all samples in a training batch, which are commonly used for neural network training.

In this section, we discuss in detail the training setup based on the triplet loss, since this is the one employed for image-based user profiling. We selected this training scheme because it is the most widely-used and has been successfully applied to a wide range of multimedia problems.

For the calculation of triplet loss function, three data samples are involved. More precisely, given a sample x along with a relevant sample x^+ and an irrelevant sample x^-, we want to learn an embedding space where the distance between x and x^+ is smaller than the one between x and x^-, i.e., $D(x,x^+) < D(x,x^-)$. To this end, the training collection may be organized in the form of triplets $T = \{x, x^+, x^-\}$, where the x, x^+, x^- are considered the *anchor*, the *positive* (relevant), and the *negative* (irrelevant) samples. A triplet expresses a relative similarity order among three samples, i.e., x is more similar to x^+ in contrast to x^-. The DNN is trained in a way to respect this ordering based on the following *triplet loss* function [18]:

$$L_{tr} = max(0, D(x,x^+) - D(x,x^-) + m)$$

where $m > 0$ is a margin parameter to ensure a sufficiently large difference between the anchor-positive and anchor-negative distances. Since triplet loss function is a hinge loss, if the network correctly assigns distances between samples within margin m, then it will not be penalised. Otherwise, the loss measures the degree of violation of the desired distance between the video pairs.

Figure 3 illustrates some visual examples of training samples in the feature space during the training of the DML network. The blue, green and red colour circles correspond to the anchor, positive and negative samples. Initially, some negatives are closer to the anchor than the positives considering a margin, which is penalized by the triplet loss function. Hence, gradients are generated that "pull" the positives closer to the anchor, while "push" the negatives away until they are beyond the safety margin. The loss is zero when there are no negative samples closer to the anchor than the positives considering the safety margin.

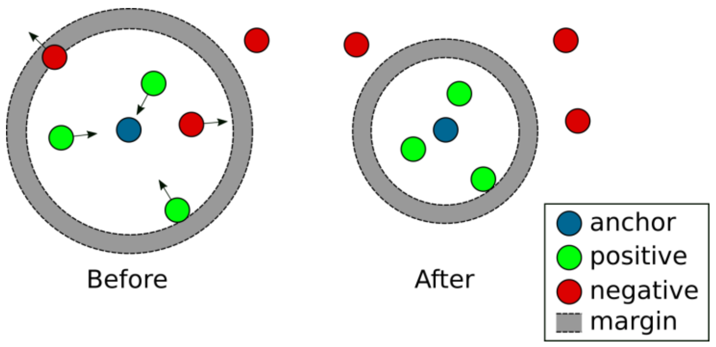


Figure 3. Illustration of samples in the feature space during training of the DML network. The triplet loss function generates gradients that "pull" positives closer to the anchor and "push" the negatives away until they are beyond the safety margin

Sampling Strategy

A critical step of the DML process is the organization of the data samples in the form required by the loss function and the composition of a representative training set. For instance, in the case of the triplet loss function, data samples have to be organized in the form of triplets. The total number of triplets that can be generated equals the total number of 3-combinations over the size N of the corpus, i.e., $\frac{N\cdot(N-1)\cdot(N-2)}{6}$. Usually, only a tiny portion of pairs of samples in a training corpus can be considered as anchor-positives, which significantly narrow down the total amount of possible combinations. Therefore, the research community has invested considerable effort into sampling strategies of negative (irrelevant) samples, which increases the performance of the networks and reduces the required training time. Early works [17] relied on the random sampling of training samples without specific criteria. However, random selection was empirically found to be inefficient due to a large number of training samples that generate zero loss. In the case of the triplet loss, triplets with negatives whose distance to the anchor is greater than the anchor-positive distance plus the margin would not generate any loss to update the network. Such examples are considered *easy negatives*. To overcome this issue, the authors in [23] focused on the mining of *hard negative* examples, which are negative samples that are closer to the anchors than the positives in the feature space, and thus would generate loss during network training. This practice led to significant performance increase and reduction of the total time required for network training. Additionally, in [18], the authors proposed a *semi-hard negative* mining scheme. Along with the hard negatives, they considered all negative samples that are further away from the anchor than the positive example, but still closer than the anchor-positive distance plus the margin. This approach offers a softer transition between positive and negative samples. The three categories of negatives are displayed in Figure 3. Negative samples inside the gray ring can be viewed as the *hard negatives*. Negatives that are further than positive samples in comparison to the anchor, but still on the gray ring, are considered *semi-hard negatives*. Finally, all negative samples outside the gray ring are the *easy negatives*.

2.5 Mobile Device Execution of Deep Learning Models

Research Approaches

Deep learning models have nowadays become very widespread and a lot of consumer applications are being developed. The integration of these applications with mobile devices, commonly smartphones and tablets, is significant to application developers as it directly affects the user experience. However, deep learning models are computationally heavy and not designed to run on restricted resources, which is a major hindrance for fully utilizing what deep learning has to offer. To overcome these difficulties academia and industry have joined forces with the former focusing on making the deep models more efficient, and the latter producing highly optimized libraries for mobile execution and at the same constantly improving the hardware of the devices. We will discuss both of these approaches, focusing first on the academic front.

Broadly speaking, most research efforts at improving the efficiency of deep models can be split in three main categories: a) modifying the architecture of the models to reduce the model's size, the computations needed for inference and the model's memory footprint, b) quantizing the weights as well as the activations of the models and c) pruning models to remove unnecessary branches. None of these categories achieve a lossless compression of the model, but rather attempt to achieve a good trade-off between the model's accuracy and the resources it demands.

a) Modifying the model's architecture
Starting with modifications to the model architecture, one of the most highly cited papers in the deep learning literature describing the now most commonly used mobile network was written by researchers at Google in 2017 [24]. Their approach was simple, but very effective: they proposed to split the convolutional layers into two pieces coining the term depthwise separable convolutions. Their novelty was that instead of using multiple filters at each convolutional layer, they would use at the first step a single kernel and convolve it depthwise with the layers input and at the second step they would perform a 1×1convolution. In this way, the computations are reduced by a factor of approximately D^2, where D is the size of the convolutional kernels, meaning that 3×3 depthwise separable convolutions use between 8 to 9 times less computations than standard convolutions. This is impressive, especially considering the fact that the model's accuracy is not significantly impacted, as shown in Table 3.

Table 3. MobileNet comparison to popular models, source: [24].

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

In a follow-up work [25] Sandler et. al. further improved the MobileNet architecture introducing residual connections and linear bottlenecks. This was the architecture that we chose to use in this deliverable to construct deep networks suitable for mobile deployment.

However, as discussed earlier, modifying the model's architecture is not the only way we can achieve significant savings on the resources consumed.

b) Quantizing model weights and activations
Quantization is perhaps the easiest and most effective way to reduce the model's size and memory footprint. We will only discuss post training quantization but quantization aware training is also possible. A good resource for the subject of model quantization is [26], a whitepaper from Google.

The first flavor of post processing quantization is quantizing the model weights. Typically the floating model is quantized to 8-bit precision, representing a 4-fold reduction over the common 32 bit floating point arithmetic. There are different quantization schemes that can be used, such as symmetric or asymmetric and per-layer or per-channel. Table 4 taken from [26] illustrates some experimental results.

Table 4. Top-1 accuracy on ImageNet for different models and post training weighs-only quantization. The last column refers to the accuracy achieved without any quantization.

Network	Asymmetric, per-layer	Symmetric , per-channel	Asymmetric, per-channel	Floating Point
Mobilenetv1_1_224	0.001	0.591	0.704	0.709
Mobilenetv2_1_224	0.001	0.698	0.698	0.719
NasnetMobile	0.722	0.721	0.74	0.74
Mobilenetv2_1.4_224	0.004	0.74	0.74	0.749
Inceptionv3	0.78	0.78	0.78	0.78
Resnet_v1_50	0.75	0.751	0.752	0.752
Resnet_v2_50	0.75	0.75	0.75	0.756
Resnet_v1_152	0.766	0.763	0.762	0.768
Resnet_v2_152	0.761	0.76	0.77	0.778

We can push the concept of quantization even further by quantizing the outputs of the activations, but for this to work we need to provide our model with some experimental data to measure the range of the activations' outputs. Table 5, again from [26], shows some experimental results.

Table 5. Top-1 accuracy on ImageNet for different models and post training weights and activations quantization. The last column refers to the accuracy achieved without any quantization.

Network	Asymmetric, per-layer	Symmetric , per-channel	Asymmetric, per-channel	Activation Only	Floating Point
Mobilenet-v1_1_224	0.001	0.591	0.703	0.708	0.709
Mobilenet-v2_1_224	0.001	0.698	0.697	0.7	0.719
Nasnet-Mobile	0.722	0.721	0.74	0.74	0.74
Mobilenet-v2_1.4_224	0.004	0.74	0.74	0.742	0.749
Inception-v3	0.78	0.78	0.78	0.78	0.78
Resnet-v1_50	0.75	0.751	0.751	0.751	0.752
Resnet-v2_50	0.75	0.75	0.75	0.75	0.756
Resnet-v1_152	0.766	0.762	0.767	0.761	0.768
Resnet-v2_152	0.761	0.76	0.76	0.76	0.778

c) Pruning the network
Another technique for improving efficiency is model pruning, which is overviewed by Blalock et. al. [27]. The main idea of pruning methods is that there is redundancy in the network's connections and it is possible to sever a portion of them with small impact on the network's performance. While the idea is straightforward, it can be implemented in a lot of different ways and no comparative study was available at the time of writing this report.

Industry-Driven Methods

On the other side of the spectrum, highly optimized mobile frameworks and APIs in combination with powerful mobile GPUs are essential to the success of the mobile deployments of deep learning models. An overview of the current landscape is provided by [28] with the collaboration of the smartphone industry leaders, namely Samsung, Huawei, Qualcomm, MediaTek and Unisoc.

In summary, there has been a lot of effort in providing the right interfaces between the hardware and the mobile operating systems and especially the Android Neural Network API as well as improvements to the ways developers can access those new capabilities. As of the time of this writing, TensorFlowLite[12] (TFLite) is at the leading framework for the development of mobile deep learning applications, while the PyTorch ecosystem recently launched its own support for mobile[13], but is still considered an early experimental release[14].

2.6 Homophily in Social Graphs

An important aspect that often characterizes users of social media is that they tend to form relations with others of similar properties, a behavior known as homophily [29], [30]. Common properties that pertain to forming relations may be gender, ethnicity, age and education, but more recent studies have revealed that users exhibit homophily on grounds of common interests too. In particular, homophily has been corroborated by numerous social media studies show that similar interests frequently pertain to forming relations, such as friendship [31], for example when those interests pertain to moral [32] and political [33] values.

[12] https://www.tensorflow.org/lite
[13] https://pytorch.org/blog/pytorch-1-dot-3-adds-mobile-privacy-quantization-and-named-tensors/
[14] https://pytorch.org/mobile/home/

To relate homophily to content-related interests, in [34] authors propose an analysis of homophily in Last.fm by analysing on-line interaction, shared interests and location. Users connect to online friends but also they reveal their real-life by listing a set of events they co-attended. In [35], authors propose an analysis of ego networks in Twitter to find homophily by considering different user attributes. They show that different types of homophily hold for different types of users. Analysis shows that there is a consistently high homophily. For example, normal users with roughly the same number of followers and followed have location and sentiment homophily. In [36], authors propose an analysis of similarity of social profiles in terms of movie preferences. Results show that homophily is indeed exhibited by users and that it is further correlated with how close users actually are (e.g. how frequently they interact). Another work [37], proposed on Facebook, has demonstrated that with a little amount of information, it is possible to infer user attributes. Authors show that users are often friends with others who share their attributes. Finally, authors of [38] show that homophily can be discovered by evaluating temporal information of social media, in particular of Facebook.

Given the above works and that their results can be applicable to HELIOS applications developing online social networks, in this deliverable we consider the approach of assuming homophily between HELIOS users and improving profile estimations by taking into account that these should also be similar to the profiles of neighbors.

3 Image based Profiling through User Interests

3.1 Motivation and Contribution

As we explained in Section 2.2 one approach to user profiling is through predefined categories and we went on to discuss some of the options. In particular, we introduced the concept of interest categories. These are semantically close to the user characteristics that we intuitively expect to be relevant to the profiles we would want to construct in a social network context. Therefore, we chose to build a model based on those and since there are not any suitable publicly available datasets, we created one from scratch.

The platform we chose for the data collection process was Pinterest given that it is a popular social network where users post content that reflects their interests. The Pinterest API was not sufficient for our data collection purposes, so we had to implement custom crawlers to extract the necessary data. We specifically constructed two datasets, one for training our hierarchical classifier and another one for testing purposes. The first one consists of approximately 60,000 images drawn from 15 coarse interest categories, 4 of which are further divided into subcategories. The latter consists of 12 randomly selected users with hand-labeled user profiles for the purpose of testing our proposed model.

To the best of our knowledge our model is the first one to provide a hierarchical representation of the user profiles being able to provide both coarse and fine descriptions of the users. Care has also been taken to choose categories that are distinct and meaningfully resonate to our goal of user profiling and matching.

The remainder of this section is structured as follows; Section 3.2 describes the dataset we used, Section 3.3 presents our hierarchical classification model and Section 3.4 discusses the results on the test dataset. Finally, Section 3.5 presents some options to use the constructed profiles for user matching.

3.2 Constructing a User Interest Dataset from Pinterest Data

Pinterest Model

For the benefit of the reader who is unfamiliar with Pinterest we will proceed with a description of how Pinterest is organized. Following a bottom up approach, Pinterest's most basic element is called a *pin* and corresponds to a single image. Pins might include comments about the photo and links to the original appearance of the photo. Pins are organized in Pinboards that usually have a specific theme that could be either broad like sports, or specific like a particular basketball team. Pinboards are created by Pinterest users and a collection of Pinboards constitutes the user profile. Each user chooses to create and organize their Pinboard in very different ways which can convey important information about the user's interests and it is this structure that we aim to capture with our model. In general, Pinboards are populated with Pins from all over the Web as it is easy for the users to save any image they encounter and captures their attention as a Pin. Of course, this also includes other users' Pins. In this way Pinterest creates a diverse wealth of images loosely organized in categories according to their Pinboards. We describe the organization as loose because there are not fixed categories as every new Pinboard could possibly correspond to a new category and it is common for Pins to belong to multiple Pinboards. Even though the taxonomy is not perfect, the structure is still rich enough for us to explore and extract useful information for our task.

Pinterest also defines 32 broad interest categories, as shown in Table 2. 32 Pinterest categories., that users can choose to assign to their Pinboard. Earlier work [9] took advantage of this mechanism to classify the Pinboards according to the label assigned to it by the user, but, unfortunately, nowadays it seems that Pinterest hides this information from the public. Ideally, we would like to collect a large number of users and classify their images according to the label of the Pinboard they belong to. We would then split the dataset into training and test sets and assign user profiles according to the distribution of the categories of their images. However, since it is not possible to extract the labels of the users' Pinboards, we have to resort to different methods of constructing the dataset.

Pinterest Classification Dataset

Initially, we identify the most relevant categories for our purpose of profiling user interests and exclude categories such as Products, Illustrations and Posters, Education, Videos that make sense in the virtual world of Pinterest but do not translate well to our needs. The final set of chosen broad categories are shown in the first column of Table 6.

Our goal now is to build an image classifier that can effectively distinguish among these categories, which means that we need to collect images that are representative of each category. Since the labels of users' Pinboards are hidden, to collect these images we take advantage of Pinterest's search service. Pinterest offers its users the ability to retrieve images based on a search query, but also enforces certain limitations as far as the number of images returned and the rate of requests. We were cautious of these limitations to ensure that the collection of images runs smoothly without disrupting the Pinterest service model, as this could potentially lead to temporary or permanent restrictions. Furthermore, we also note that the number of retrieved images can vary unpredictably and depends on the given query. Table 6 shows the search queries that were used to collect the images of the dataset and how many images were collected for each category.

Table 6. The number of images for each category collected by submitting the queries shown to Pinterest.

Category	Search queries	Images
Fashion	"mens fashion", "womens fashion", "fashion hair", "fashion hair men", "fashion bags", "fashion jewellery", "fashion nails", "fashion makeup", "fashion shoes"	8279
Entertainment	"entertainment", "music", "entertainment music", "movies", "cartoon", "anime", "series", "video games", "books"	7897
Art	"art", "doodle art", "art photography", "paintings", "sculptures"	6362
Sports	"basketball", "soccer", "football", "gymnastics", "tennis", "swimming"	5263
Animals	"animals", "pets", "wild animals", "birds", ?"fish"	4649
Vehicles	"cars", "motorcycles", "trucks", "planes", "boats", "ships"	4454
Food-Drinks	"food", "drink", "sweets", "junk food"	4370
Home decor	"home decor", "modern home decor", "cozy home decor"	3892
Tattoos	"tattoos", "tattoos sketches"	3424
Health-Fitness	"health fitness", "yoga", "fitness", "workout"	3238
Architecture	"architecture"	1876

Technology	“technology”, “tech gadgets”	1783
Travel	“travel”	1685
Plants-Flowers	“plants”, “flowers”	1653
Kids-Babies	“kids”, “babies”	1641
Total		61,993

We collected a total of about 62,000 images, but as we can see from Table 6 as well as Figure 4, the number of collected images varies a lot per category.

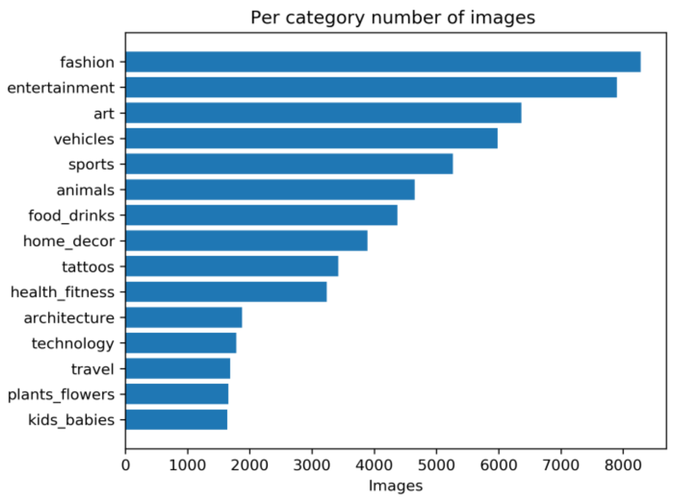


Figure 4. The distribution of the images in the defined categories. Excluding art, the 4 largest ones will be further divided in subcategories.

The amount and diversity of images in each category is affected both by its popularity in the Pinterest platform and by how narrow or broad it is. For example, fashion is both very popular among Pinterest users and includes a lot of subcategories like clothes, shoes, hair and accessories. On the other end of the spectrum, the kids and babies category is very specific and targets mostly parents with young kids.

The existence of broad categories with a lot of images is an indication that it would be interesting to further split these categories to subcategories. Pursuing this direction, we selected the fashion, entertainment, vehicles and sports categories as they had the most images, were popular and a subdivision of neither too specific nor too ambiguous subcategories was possible. As far as the art category is concerned, it does have more images than both vehicles and sports, but it is also more ambiguous carrying a lot of different meanings and defining subcategories is not a straightforward task. The way we divided the four previously mentioned categories is shown in Figure 5.

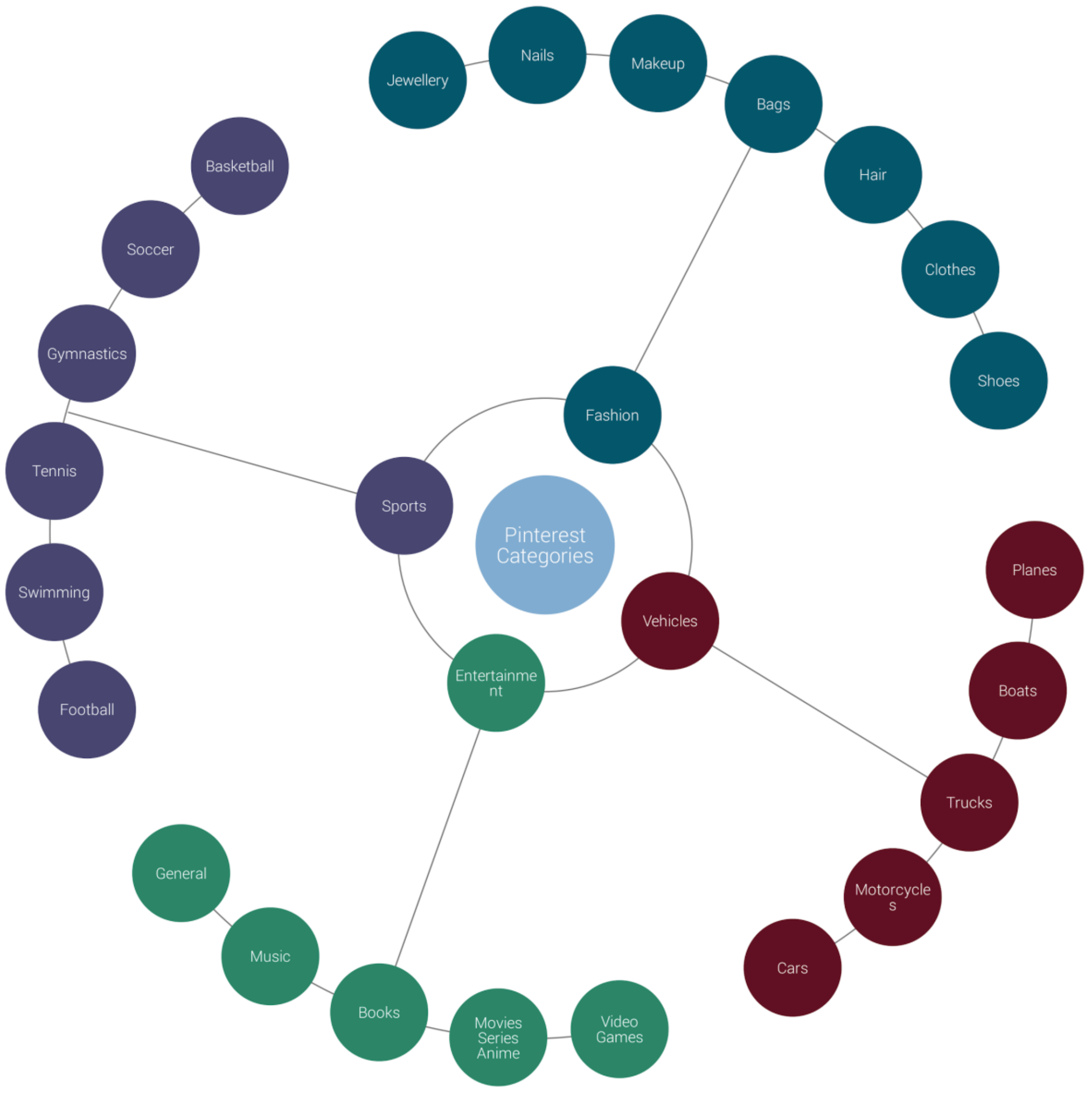


Figure 5. Four of the five largest categories were divided into subcategories as depicted in the figure.

To collect the images, the search queries, displayed on Table 6, were matched trivially to the appropriate subcategories. We note that the subcategories did not replace the parent categories, but instead added another depth level to the dataset that will be exploited to build a hierarchical classifier. The hierarchical classification scheme will be presented and analyzed in Section 3.3.

Looking more closely to the sports category, it probably is the one with the most clear subdivisions. We have included some of the most popular sports as subcategories, such as soccer and basketball, but inevitably these are only a very small portion of the existing sports. Fortunately, however, this is not the first attempt of recognizing various sports activities. Li et. al. [39] in 2007 collected a sports event dataset, which is publicly available and consists of 8 sport events for a total of 1579 images. The included sport events are not the most common ones, but it would be potentially beneficial to incorporate them in our dataset. Example images from each category are shown in Figure 6 and Figure 7 for our dataset and the UIUC Sports Event Dataset respectively.

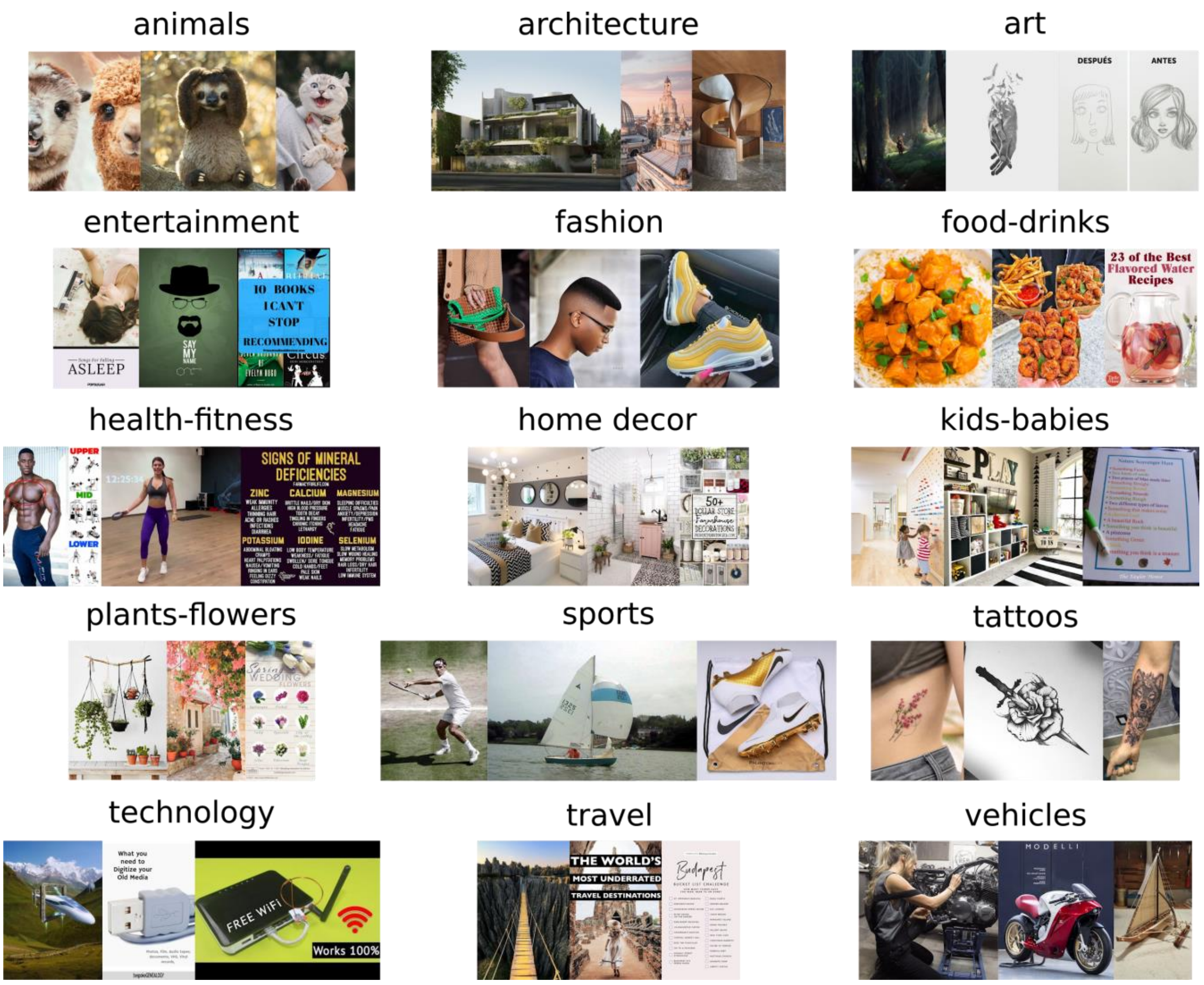


Figure 6. These are some sample images from the Pinterest dataset we constructed, organized in 15 coarse interest categories.

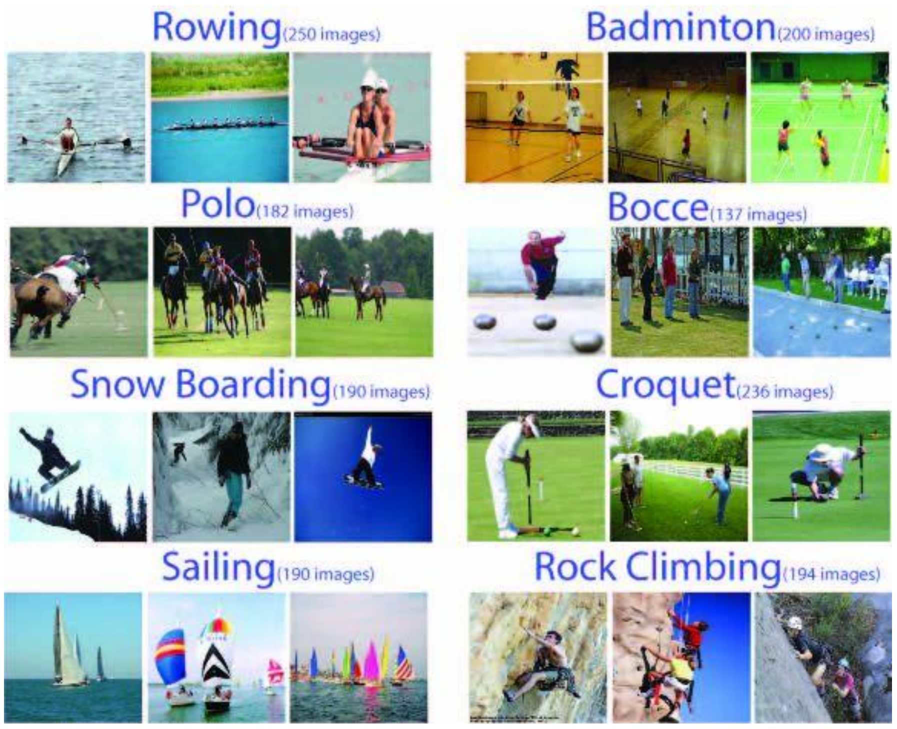


Figure 7. The image is taken from the UIUC Sports Event Dataset's website[15] and depicts some sample images of the dataset divided into 8 sports categories.

Pinterest Profiles Evaluation Dataset

To evaluate the generated user profiles we need a set of users for whom we have some ground truth profiles. Unfortunately, as there is no means of automatically obtaining these labels for Pinterest users, we have to resort to manual labeling. Although manual labeling is time consuming and not scalable, it is still valuable because it can provide us with a sense of how well our algorithm is performing. We need, of course, to keep in mind that results produced from small sample sizes are inherently infested with high variance.

Knowing these limitations, we randomly selected 12 Pinterest users and downloaded their Pinboards' images while maintaining an upper limit of 100 images per Pinboard. Then, to the best of our ability we assigned to each Pinboard a label from the predefined interest categories according to their image content. However, this was not always possible as some Pinboards did not correspond to any of the categories, while others were an indistinguishable mix of the predefined categories. An example of the former would be humour related Pinboards; humor is a very broad theme and in fact, many humoristic pins really make sense only if the text embedded in them is taken into account, which cannot possibly be understood from just an image recognition framework. An example of the latter, that is an indistinguishable mix of the predefined categories, on the other hand, would be a Pinboard named "Things I love". Although the pins included in such a Pinboard would probably be highly informative, because we label images at the granularity of Pinboards we would not be able to properly make use of them.

Therefore, as far as the first category is concerned, we decided to include all of those images to the users' data, but not take them into account for the calculation of the ground truth profile. This

[15] vision.stanford.edu/lijiali/event_dataset/

way, these images can be perceived as noise that do not contribute any significant information for the profile we want to construct, but they exist as they are expected to in a real user image collection. For the images of the latter category, that do convey significant information of the user's profile, but, unfortunately, cannot be taken into account when creating the ground truth labels, we decided to not include them in the dataset as they could influence the user's profile without it being reflected in the ground truth labels.

Finally, to construct the ground truth profiles each image inherited the label of the Pinboard it belonged to and the label distribution of a user's images determined their profile; an example for a hypothetical user is shown Table 7.

Table 7. This table offers an example calculation of the interest profile of a hypothetical user.

Pinboard Name	Images	Category	Profile
Cocktails	15	Food-Drinks	0.3
Gorgeous Dresses	20	Fashion	0.4
Puppies	10	Animals	0.2
Workouts	5	Health-Fitness	0.1

Some statistics of the collected dataset are shown in Figure 8 and Figure 9.

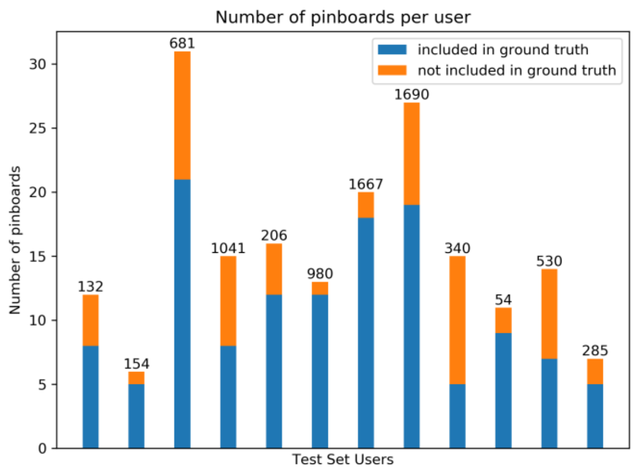


Figure 8. The figure shows the total number of Pinboards per user in the test set, along with a visual cue about how many of those were included in the calculation of the user's ground truth profile. The number above the bars indicates the total number of images per user. We note that all of these images were included in the user's image set regardless of whether the corresponding pinboard was included in the calculation of the ground truth profile.

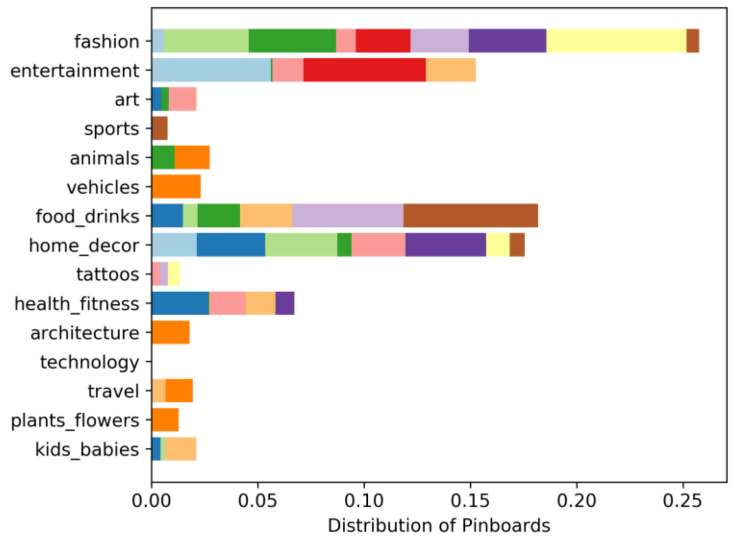


Figure 9. Distribution of Pinboards in the interest categories. Each color represents a different user and the width of the bar is proportional to the number of Pinboards classified in the corresponding interest category.

3.3 Hierarchical Classification of User Interests using CNNs

To construct content aware user profiles our idea is to classify the images in interest categories and interpret the interest distribution of a user's images as the user's profile. To train and test our model we will use the dataset we created and described in Section 3.2, consisting of approximately 62,000 images. The classification strategy will employ a hierarchical scheme that will be used to construct user profiles at both a coarse and a fine detail level. The coarse level will correspond to the 15 categories displayed in Table 6. Upon request a finer profile will be computed that will attempt to identify more specific interests that could be useful for more accurate user matching.

CNN Architecture

The first step of this process will be a CNN, which is hardly surprising considering their widespread deployment and success. Nowadays, there are a great many different CNN architectures that have been proposed and none of them has been able to dominate as the best one. There is always a trade-off among the CNN characteristics and the right balance depends on the specific application; depth and width are at odds with efficiency and memory and power consumption, but also the CNN structure where its fully connected, residual or inception-like can be significant aspects. We will compare the performance of a deep network based on residual connections with the architecture of a mobile network designed specifically to be lightweight, preserving compute and memory resources. Furthermore, we will observe the impact on accuracy of applying quantization and transforming the model to a form suitable for execution on the Android platform. Specifically, we chose to use EfficientNet-B3 [40], a state of the art CNN superior in both accuracy and size as Figure 13 later in this chapter shows, and to meet the mobile requirements, the second version of the MobileNet architecture, overviewed in Section 2.5. Both networks were pretrained on ImageNet and we fine-tuned them for the classification task.

Hierarchical Classification

Instead of training a flat model for the classification task we opted for a hierarchical one because it can provide better flexibility and accuracy. The classification categories as presented in Section 3.2 naturally fit a hierarchical tree with two levels. The images of a branch share a lot of similarities and can be difficult to identify the correct subcategories. For example, clothes and bags are two subcategories of fashion, but an image of a woman holding a bag can be misleading about what is at the center of interest. So, it makes sense to recognise first the general category and then proceed to a finer granularity. To be more specific, for each of the categories fashion, entertainment, sports and vehicles that are analyzed in subcategories we are going to build a local classifier. Although this increases the number of models and thus the storage and computational requirements, it does make the task of the coarse classifier easier and provides the opportunity for more accurate results in the finer categories. The local classifiers will be based on the same CNNs as the coarse classifier, pretrained on ImageNet.

Thresholding

Another issue that we should take into account as we design our model is the fact that we do not expect all the available images for a user to be informative. It is quite possible that some of them will not convey any information about the interests of a user, the most obvious example being photos that were taken by accident as well as blurry and distorted photos. To simulate this possibility in our experiment we have included as described in Section 3.2, images that do not fit the categories we have defined and thus do not contain useful information for our model. To provide some robustness against such noisy pictures we make use of a filtering mechanism that rejects the images that our model classifies as not informative enough. A non-informative image in our case would correspond to an image that the model cannot sufficiently discriminate against. From an information theory perspective, the uninformative images would be those that maximize the Entropy [41] of the outputted probability distribution , which occurs when the latter are close to the uniform distribution. In this case, the output probability distribution would not have any distinct peak and the model would not confidently predict a category. Therefore, when our model outputs a probability distribution entirely below a predefined threshold, it would be reasonable to discard the corresponding images. The most suitable threshold is a hyperparameter that can only be empirically determined.

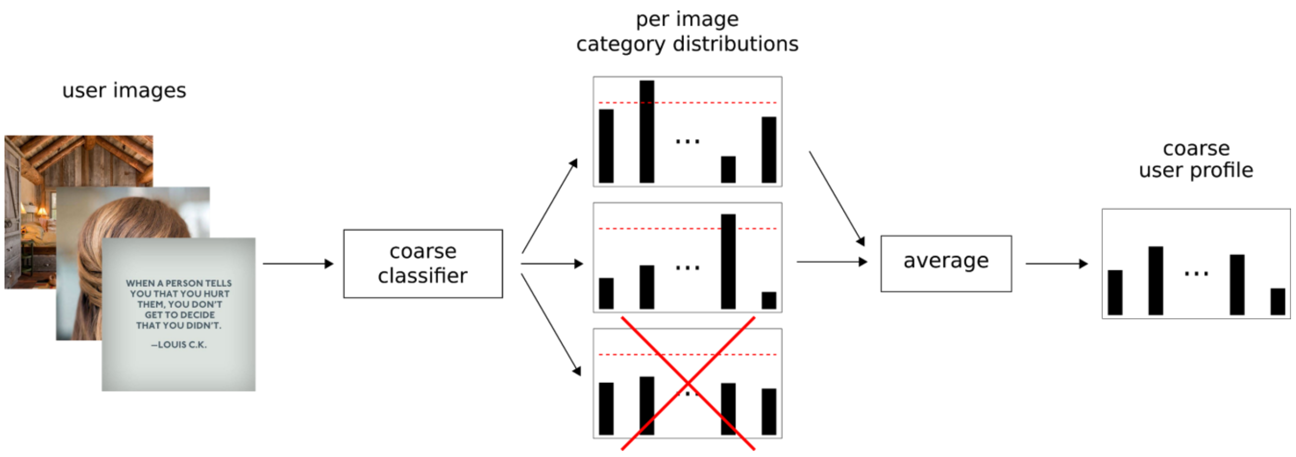

user images
WHEN A PERSON TELLS YOU THAT YOU HURT THEM, YOU DON'T GET TO DECIDE THAT YOU DIDN'T.
–LOUIS C.K.
coarse classifier
per image category distributions
average
coarse user profile

Figure 10. The coarse classification pipeline. Firstly, the user images pass through a CNN based coarse classifier to extract the per image category distributions. If it is the case that an image is not classified confidently enough, it is discarded. The remaining distributions are then averaged to produce the user's coarse profile.

Formal Definition

The concrete calculations of the coarse and fine grained user profiles are as follows; assuming that a user has N available images $I_i, i = 1, \ldots, N$ in his collection, we feed them first to the coarse classifier c_{coarse} to produce the per image coarse categories distributions $d^i{}_{coarse}$,

$$d^i_{coarse} = c_{coarse}(I_i), i = 1, \ldots, N.$$

Then, to calculate the user's coarse profile p_{coarse} we would discard the images with distribution below the threshold t and take the mean of the rest, as is visualized in Figure 10. Mathematically, let the set D_t be defined as

$$D_t = \{i \in \{1, \ldots, N\} \mid \exists \, \text{category } j \text{ s.t. } d^i_{coarse}[j] \geq t\}$$

then the user's coarse profile would be

$$p_{coarse} = \frac{1}{|D_t|} \sum_{i \in D_t} d^i_{coarse} \, . \tag{3.1}$$

On the other hand, to calculate the fine-grained profile we would need to pass the images also from the local classifiers $c_{fashion}, c_{entertainment}, c_{vehicles}, c_{sports}$, corresponding to the fashion, entertainment, sports and vehicles categories respectively. We could pass all the user's images from each local classifier, but that would be a waste of resources, as for example an image of a bedroom would not produce a meaningful distribution when fed to the fashion classifier. Therefore, it is reasonable for the local classifiers to only take as input images of the same category as the one they were trained on and for the rest to assume that the output is approximately uniform. The final probabilities for the subcategories are calculated by weighting the output probabilities of the local classifiers by the probability of the parent category.

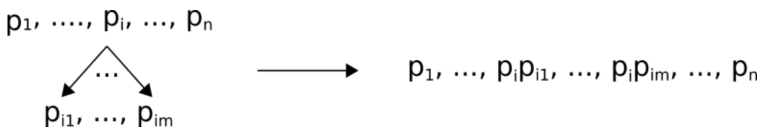

$p_1, \ldots, p_i, \ldots, p_n$
…
$p_{i1}, \ldots, p_{im}$
$p_1, \ldots, p_i p_{i1}, \ldots, p_i p_{im}, \ldots, p_n$

Figure 11. The output of the coarse classifier, displayed with single subscript, is combined with the output of the local classifier, displayed with double subscripts, to produce the final output of the fine classification pipeline. The calculation is based on distributing the probability of the parent category according to probability distribution of the subcategories.

Formally, let us assume that $i_{fashion}$ is the index of the fashion category and define an intermediate variable

$$\tilde{c}_{fashion}(I_i) = \left\{ \begin{array}{ll} \frac{1}{F} & d^i_{coarse}[i_{fashion}] < t \\ c_{fashion}(I_i) & d^i_{coarse}[i_{fashion}] \geq t \end{array} \right\},$$

where F is the number of fashion subcategories. We will also define the output distribution of the fashion classifier as

$$d^i_{fashion} = d^i_{coarse}[i_{fashion}] * \tilde{c}_{fashion}(I_i), i = 1, \ldots, N.$$

The final distribution for the fine-grained classification scheme can be obtained from the coarse distribution by substituting the coarse probabilities of the parent categories with the previously defined $d^i{}_{parent\ category}$. Doing the substitution for only the fashion category, the fine distribution would be

$$d^i_{fine} = [d^i_{coarse}[0], \ldots, d^i_{fashion}[0], \ldots, d^i_{fashion}[F], \ldots, d^i_{coarse}[C]] \, ,$$

where F is the number of fashion subcategories, C is the number of coarse categories and ${d^i}_{fashion}[0]$ was inserted at the $i_{fashion}$ index. Similarly, we would insert at the appropriate indices the distributions corresponding to the other categories that have subcategories. Finally, the fine grained user profile would be

$$p_{fine} = \frac{1}{|D_t|} \sum_{i \in D_t} d^i_{fine}. \tag{3.2}$$

A visual representation of the process is shown in Figure 12.

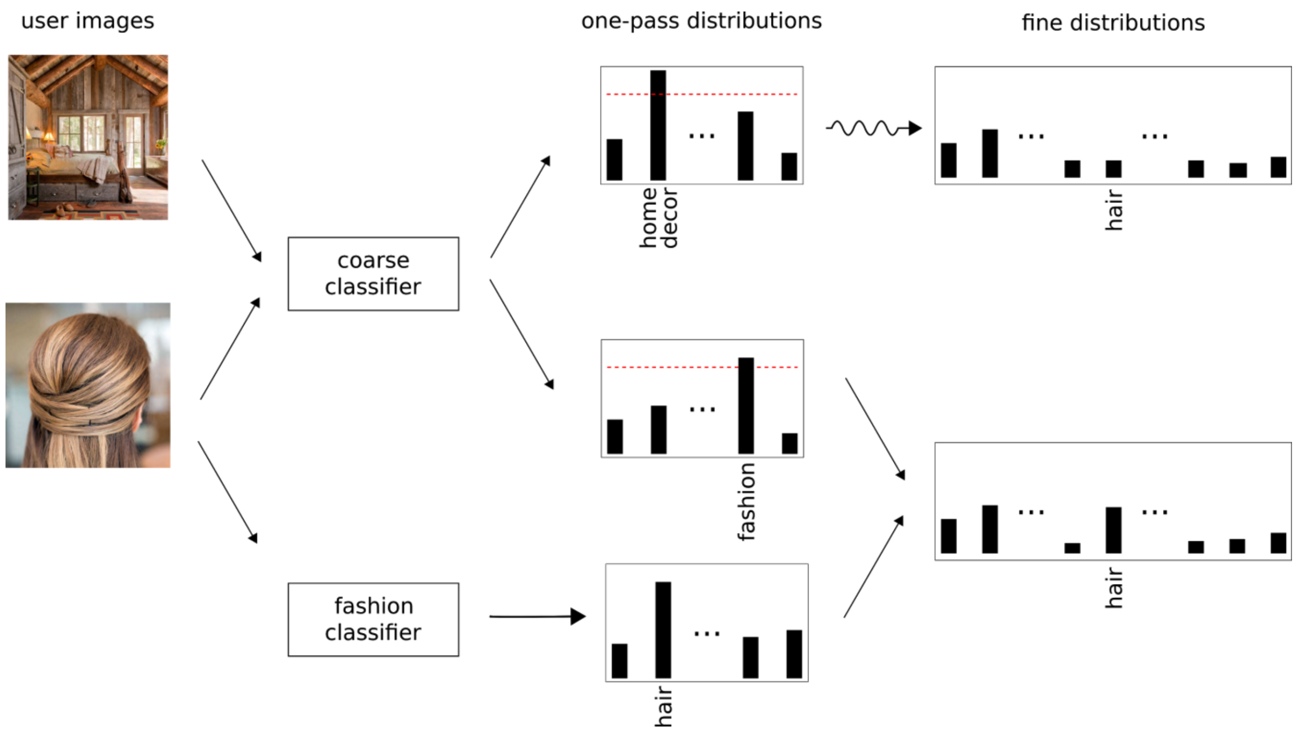


Figure 12. The fine classification pipeline. The user images first pass through the coarse classifier and if an image is confidently classified in one of the four parent categories, that is the probability exceeds the predefined threshold, it is also passed from a second, local classifier, specifically trained on the subcategories of the parent category. The two distributions are then appropriately combined, as in Figure 10, to produce the final category distribution of the image. Finally, as in the case of the coarse classification, although not explicitly shown, the user profile will be calculated by averaging the distributions of their images.

Implementation Details

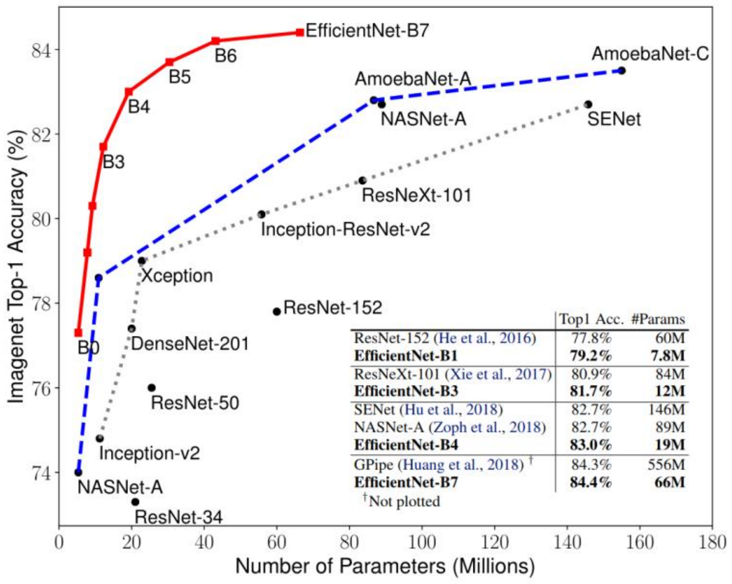
The model was implemented using the Keras and Tensorflow frameworks as well as TFLite for the model's mobile version. The dataset was first split in training and validation sets with proportions 4 to 1. In all cases the CNN was pretrained on ImageNet and we fine-tuned the network for 10 epochs with empirically determined initial learning rates 10^{-3} for coarse and 10^{-4} for local classifiers. For best results on coarse classifiers their learning rate was further reduced at epochs 5 and 8 by a factor of 10, while for the local ones it was reduced only on epoch 8 by the same factor. All models were trained on a single GTX 1070 card and no model required more than 4 hours of training.

To create the mobile versions of the models, we used the TFLite framework and converted the corresponding MobileNetV2 based models to TFLite binaries using the default optimizations, which most importantly include the quantization of the weight parameters. These binaries are suitable for execution in an Android environment and are compatible with a Java based project.

Hierarchical Classification Results

The classification accuracy of our models on a left-out validation set is presented in Table 8. We aim to compare the accuracy we can achieve in the Pinterest Interests Dataset using a performant, but computationally intensive server side CNN model with more lightweight models that can be used in a mobile setting.

The server side model is based on the EfficientNet-B3 CNN [40] pretrained on ImageNet and fine-tuned for our purposes. Tan et. al. propose a family of models, which they name EfficientNet, ranging from the EfficientNet-B0 to the EfficinentNet-B7. Higher B numbers correspond to larger, but more accurate models and the exact relationship as well as a comparison among other popular CNNs is shown in Figure 13, replicated from the original paper. As the EfficientNet family seems to be the best performing CNN, we chose to use the EfficientNet-B3, that represents a good trade-off between accuracy and size, in our experiments for the server side model.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Figure 13. Model Size vs ImageNet accuracy, source: [40]

The second model is based on the MobileNetV2 CNN, presented in section 2.5, and it is most suitable for mobile execution. It is a compact model and well established in the deep learning field, capable of fast inference time without sacrificing much of the accuracy. The third model is a weights-only quantized version of the previous one, created with the TFLite framework and the default optimization parameters.

The results are shown in Table 8 and measure the classification accuracy in 5 tasks. The first one, labeled coarse, refers to the task of classifying the images in 15 coarse categories as shown in Table 6. The other four, labeled as fashion, entertainment, sports and vehicles, refer to the local classifiers trained only on the corresponding coarse category with the subcategories as shown in Figure 5. For the coarse classification task we provide both top-1 and top-5 accuracies, but for the local classifiers the top-5 accuracy is not relevant as the number of subcategories is small.

The performance of the models is ranked as expected with the computationally intensive EfficientNet-B3 being first in all tasks and the MobileNetV2 naturally outperforming its compressed and quantized TFLite version. The EfficientNet-B3 achieves 70.7% top-1 accuracy in the coarse classification task which is reasonable, while the fashion, sports and vehicles classifiers perform even better achieving 92.8, 81.6 and 90.3 respectively. The entertainment classifier seems to be the one lagging behind, but this is understandable as its subcategories, namely music, books, movies-series-anime, video-games and another general one, are the most varied in terms of visual appearance.

The accuracy of the MobileNetV2 based model is less than that of the EfficientNet based one, but the drop is no more than 4% for the coarse, fashion, entertainment and vehicles classifiers, but 5.4% for the sports category. The story is pretty similar for the quantized TFLite model as far as the coarse, fashion and vehicles classifiers are concerned, but there is a significant drop of 10.7% and 11.7% in the case of the entertainment and sports classifiers which is unfortunate and for that reason we consider not quantizing the models in these two categories.

Table 8. Classification results of 3 different models for the coarse classifier as well as the local ones on the left out validation set.

	Coarse		Fashion	Entertainment	Sports	Vehicles
Models	top-1	top-5	top-1	top-1	top-1	top-1
EffientNet-B3	70.7	94.4	92.8	67.7	81.6	90.3
MobileNetV2	66.7	93.7	91.8	64.7	76.2	87.3
Quantized TFLite	62.9	88.3	86.8	54.0	64.5	87.0

3.4 Profiling Evaluation

The classification results of Table 8 show that our models can achieve reasonable accuracy, but for our purposes, classification is only used as a means for constructing user interest profiles. As such, it would be beneficial to have a way to directly evaluate the accuracy of the profiles instead of the intermediate classification results and this was the motivation for the test set we constructed and presented in Section 3.2. To briefly summarize, the test set consists of 12 randomly selected Pinterest users whose images have been downloaded and organized according to their Pinboards. A ground truth profile is calculated for each user by manually labelling their Pinboards excluding some that cannot be properly labeled.

To evaluate the profiles we use the metrics presented in Section 2.3 treating the problem similar to a ranked information retrieval task. These metrics are the Area Under the Curve (AUC), Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). We calculate these metrics per user and then average them to produce the final scores per model and granularity, which are shown in Table 9 and

Table **10**.

However, our proposed method of calculating the interest profiles involved an additional hyperparameter, the confidence threshold t. For the calculation of the profiles we take into account only the images that have an output interest distribution with a peak higher than the threshold. The effect of t for the three metrics and the different models and granularities is shown in Figure 14. We see that the t value does indeed influence the output and it seems that a value of $t = 0.9$ produces better results than lower thresholds and is more consistent than higher ones. Therefore, this was the value of choice for the results of Table 9 and

Table **10**.

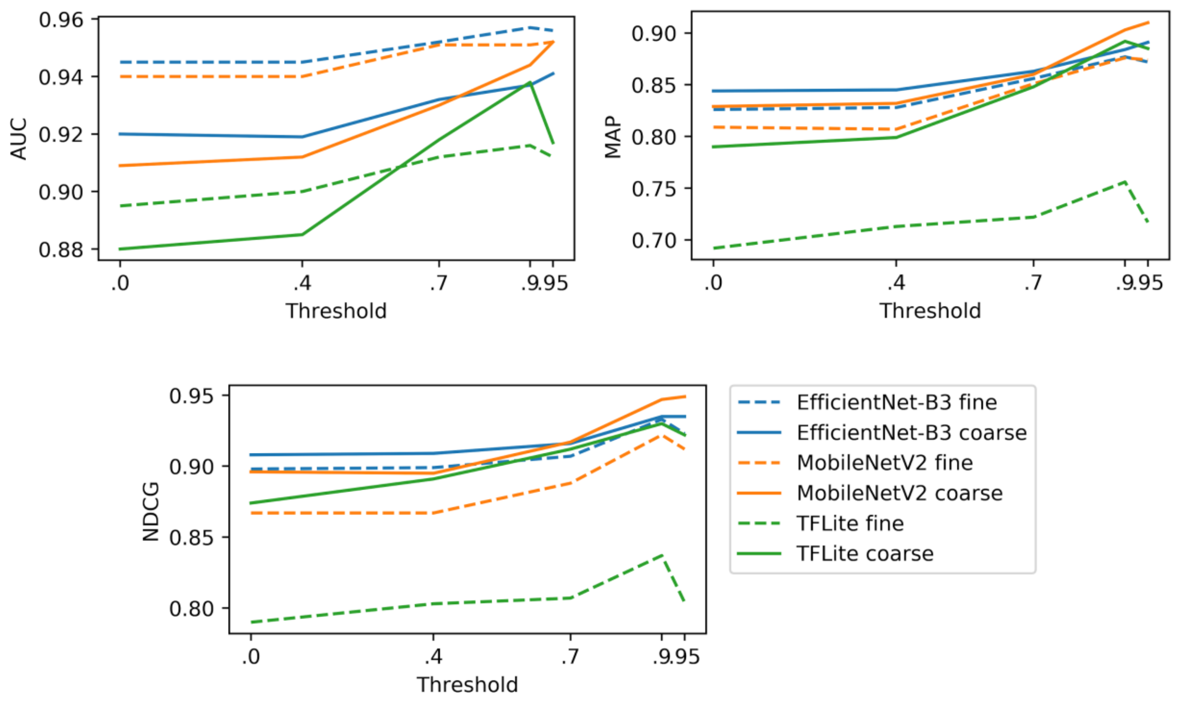


Figure 14. AUC, MAP and NDCG at different threshold values for each model and granularity.

Table 9. AUC, MAP and NDCG at the 0.9 threshold, chosen based on the results of Figure 14, for the different models at coarse granularity. The means and standard deviations are taken over the users.

Coarse	AUC		MAP		NDCG	
Models	Mean	Std	Mean	Std	Mean	Std
EffientNet-B3	93.3	4.45	87.6	8.76	93.0	7.00
MobileNetV2	94.4	4.82	89.8	6.57	94.3	5.25
Quantized TFLite	93.6	4.86	88.5	7.42	92.5	6.30

Table 10. AUC, MAP and NDCG at the 0.9 threshold, chosen based on the results of Figure 14, for the different models at fine granularity. The means and standard deviations are taken over the users.

Fine	AUC		MAP		NDCG	
Models	Mean	Std	Mean	Std	Mean	Std
EffientNet-B3	95.4	5.13	87.0	11.1	92.8	6.87
MobileNetV2	95.0	6.73	87.3	11.4	91.6	7.66
Quantized TFLite	91.4	5.94	74.3	15.92	82.9	16.79

The fine-grained model has more categories and thus constructing the fine profile is a more challenging task, but it could in return provide better matching suggestions as it captures more information about the user. In

Table **10** we can see that in general fine profiles are performing close to the corresponding coarse ones except for the TFLite fine model that shows a significant drop in the MAP metric as compared to the TFLite coarse model.

It is, also, interesting that the coarse case MobilenetV2 seems to outperform the corresponding EfficientNetB3 models, even though the latter has higher classification accuracy as shown in Table 8. The difference, however, is not large and taking into account the high standard deviations, it should not be treated as statistically significant. The high standard deviations are to be expected as 12 users is a small sample size and, therefore, the results, in this section are not to be interpreted as conclusive, but rather as providing an indicative view of the model's performance at the task of interest retrieval. Increasing the sample size is very time consuming at this stage as the labelling is done manually and is left as an issue for further work.

3.5 User Matching

Aside from constructing the users' profiles, this deliverable also aims to demonstrate their usefulness in the context of user matching. User matching is a common denominator in most of the social media platforms as it aligns with the core idea of social media, that is a place where users can find and connect with other users, extend beyond physical boundaries and have novel interactions. However, predicting meaningful user matches can be as daunting and complicated as predicting human behaviour and thus we will only be able to approximate this problem based on the inferred users' interests.

The user profiles we have constructed so far, if stripped from any semantic meaning, are merely n-dimensional real coordinate vectors, where n is the number of interest categories, and whose coordinates sum up to one. These vectors can be embedded in $\Re^n$, the real n-dimensional vector space with addition and scalar multiplication defined as usual. If we also define an inner product on $\Re^n$, we will have a normed vector space structure with the induced norm,

$$||x||^2 = \langle x, x \rangle, x \in \mathbb{R}^n,$$

where ||·|| and <·,·> stand for the norm and inner product respectively. A normed vector space also has an induced metric *d* defined as

$$d(x, y) = ||x - y||.$$

The metric d and the inner product are relevant for our purposes because they respectively correspond to the notions of distance and similarity between two vectors. We could therefore match the users based on how close or how similar they are. In the special case that the vectors are normalized, meaning they have the same norm, the two measures coincide in the sense that

$$||x - y|| < ||x - z|| \implies \langle x, y \rangle > \langle x, z \rangle \, ,$$

that is if x is closer to y than z, then x is more similar to y than z.

The usual inner product in $\Re^n$ is defined as

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i = x^T I y \, ,$$

but this is not the only possible definition; it can be easily shown that if *A* is a positive definite matrix, then

$$\langle x, y \rangle = x^T A y \tag{3.3}$$

is a valid inner product definition. The difference is that while the usual definition assumes that each dimension is equally important and orthogonal to each other, the more general definition above makes no such assumption. This distinction is important in our case because the dimensions are indeed correlated; for example, an interest in health and fitness could be an indicator of an interest in sports. Therefore, if we pick a positive definite matrix A that suitably reflects the similarity between the categories, the calculation of the similarity between vectors could be improved which would lead to a better matching algorithm. In the following we will explore some of the ways to pick such a matrix, but before that we will state without proof a useful theorem[16] bounding the eigenvalues of a matrix inside disks, which will prove useful later on to ensure that the eigenvalues are positive, thus providing a criterion for positive definiteness.

> **Theorem** (Gershgorin circle theorem). Let a complex $n \times n$ matrix have entries a_{ij}. For $i \in \{1, \dots n\}$ let $R_i = \sum_{j \neq i} |a_{ij}|$ be the sum of the absolute values of the non-diagonal entries of the i-th row and $D(a_{ii}, R_i) \subseteq \mathbf{C}$ be a closed disc centered at a_{ii} with radius R_i. Then, every eigenvalue of the matrix lies within one of those discs.

Intuitively, the theorem states that the eigenvalues of a matrix cannot be too far from its diagonal elements as long as the off-diagonal elements are small enough. This is a handy result that can be used to assert that all the eigenvalues of a matrix are positive and therefore that the matrix is positive definite.

[16] https://en.wikipedia.org/wiki/Gershgorin_circle_theorem

One way to approximate the correlation between the different categories is the Jaccard index, also known as Jaccard similarity coefficient[17]. The Jaccard index is defined over sets as the intersection over union, that is,

$$J_{ij}(U_i, U_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|},$$

where, in our case, U_i is the set of users with images from category i. Intuitively, this quantity expresses the ratio of users that have images in both i and j categories to the users that have images in either i or j. This ratio reflects the measure of correlation between the categories as when it is close to 1 the presence of images from category *i* implies with high probability the presence of images from category j, while J_{ij} close to 0 implies the opposite. We could, therefore, use the Jaccard index to define an inner product, but simply setting $A_{ij} = J_{ij}$ does not guarantee that *A* is positive definite. However, $J_{ii} = 1$ for every i and so if the sum of the off-diagonal elements of each row is restricted to be less than 1, then according to Gershgorin's theorem all the eigenvalues would be positive and the matrix A would indeed be positive definite. A simple way to enforce this restriction would be to normalize the off-diagonal elements to sum to a number $s < 1$in case the sum exceeds 1. This way we can define the similarity of two users' profiles through the inner product or their distance through the norm of their difference. In general, however, these two ways will not lead to the same ordering because A is not guaranteed to be an isometry and as such the new norm of the users' profiles could be different. The problem, however, with this technique is that to accurately calculate the J_{ij} elements we need a lot of user profiles as training data. Unfortunately, in our case the training data are restricted to the interest classification problem and only the testing data contain hand-labeled user profiles.

A different approach is to approximate the J_{ij} elements with the semantic similarity of the categories, which can be obtained with the commonly used word2vec embedding, and leave the rest of the process of calculating the matrix *A* unchanged. The drawback, however, in this case is that the categories can be correlated in a more complex way and not merely through semantic similarity of their labels. For example, it could be possible that the users who like sports are more probable to also be interested in cars, which cannot be explained through semantic similarity of sports and cars as they are not semantically related, but it could be attributed to the more complex behavioral patterns of humans.

The third option is more indirect and aims to take advantage of Pinterest's suggestions. For each topic explored, Pinterest suggests other topics related to that one. Following this path recursively one can build a graph of various topics. The idea is to then calculate the similarity between the root nodes based on the constructed graph. One measure of such similarity is the commute time distance [42] which is defined as the average number of steps a random walker, starting from node *i*, takes before entering node j for the first time and returning to. Without a loss of generality, we will consider commute time to span two step units when $i = j$. The commute time distance is symmetric and can be computed in closed form from the Laplacian matrix of the graph. However, for our purposes we need to scale the computed similarity between 0 and 1. To do this, we propose to define the similarity $s(i,j)$ as

$$s(i,j) = cd^{-a}(i,j)$$
$$c = 2^a$$

[17] https://en.wikipedia.org/wiki/Jaccard_index

where $d(i,j)$ is the commute time distance between nodes i and j and α is a positive constant that controls how fast the similarity decays. c is a normalization constant to make sure that the similarity is equal to 1 when then distance is 2. After these calculations we can proceed to the definition of the matrix A as originally described.

The success of this option clearly depends on the constructed graph's structure, which in turn depends on how interconnected the suggested related topics are. The worst case scenario would be that the graph has as many connected components as the number of categories, in which case no useful information can be extracted and A would be equal to the identity matrix corresponding to the usual definition of the inner product. A parameter that directly influences the graph's structure is the depth of our search. Depth equal to 1 would correspond to only considering the root node's related topics, depth equal to 2 would also consider the related topics of the related topics and so on. Another issue is that this technique depends on Pinterest's opaque algorithm for suggesting related topics, but we expect that the algorithm would be based on the users' behaviour patterns, thus revealing useful information. However, early experiments showed that the constructed graph was indeed too sparse with many disconnected components and, therefore, we decided to not pursue this method further.

4 Image based Profiling through User Similarity

4.1 Motivation and Contribution

In Section 3 we proposed profiling the users through a hierarchical classification scheme that involved some predefined interest categories. This approach is reasonable and has the benefit that the constructed profiles correspond to meaningful human concepts, namely distributions over interest categories. Each user profile can be interpreted in this way and so can the user matchings. This is a very desirable property as it paves the way for user profile and matching introspection features.

However, it also comes at a cost, the categories as previously mentioned are predefined and as such the semantic variability that we are able to capture is inherently limited. This is to be expected as it is often the case that model expressivity is at odds with model interpretability. Tipping the scale towards the expressivity side, in this section we propose a method that in principle is able to capture significantly more semantic nuances, while sacrificing some of the model's interpretability. This, however, cannot be achieved with just some small modifications to the interest categories model described in Section 3, but rather requires a different approach. To achieve a more expressive model, we have to turn away from predefined concepts and instead approach our goal of user profiling more indirectly. We propose to do this by exploiting user similarity data to construct a latent user representation that retains as much of the original similarity structure as possible. Learning from similarity data closely resonates with the field of Deep Metric Learning (DML) and as such we propose the use of the triplet loss during training. Triplet loss requires batches of triplets that consist of an "anchor" user (reference user) along with a similar and a dissimilar user. This way the model learns to build user representations that reflect the original user similarity structure and can effectively be interpreted for our purposes as user profiles. Because the user representations are not trained directly on manually specified concepts, the model has the capacity to discover through the available data the most suitable way to represent the users.

Unfortunately, however, the Pinterest dataset we used to train the interest profiling model of Section 3 does not even have embedded the notion of users, let alone a similarity measure between them. To realize our plan of using the triplet loss to train suitable user representations, we resorted to a new data source that contains users along with their photo collections and we annotated it with user similarity data. The construction of such a dataset is the subject of the following section.

4.2 Constructing a User Similarity Dataset from YFCC100m

In order to overcome the shortcomings of the aforementioned Pinterest dataset, we decided to leverage information from the YFCC100m dataset[18]. This dataset is one of the largest publicly available multimedia collections, which contains over 99 million Creative Commons-licensed photos and videos annotated with user related data (user id and user nickname), geolocation data (longitude and latitude) as well as user and device generated tags.

One way to derive similarity metrics among users, would be to estimate the similarity between the sets of the machine[19] and/or user tags[20] of their photos among them. The downside of this

[18] https://multimediacommons.wordpress.com/yfcc100m-core-dataset/
[19] A set of labels automatically assigned to a photo by the device that it was shot.

approach is that the machine tags associated with each photo, usually contain information regarding the device used to shoot the photo, whereas the user tags associated with each photo, are tags that are written by users themselves, and sometimes contain words or pseudowords that have no meaning and/or might not be related with the objects that the photos are depicting. Both of the above features constitute a dataset with noisy data and extracting user similarity based on those tags would likely lead to inconsistent results.

To address the previous challenge, we exploited one of the several released expansion packs[21] of the same dataset, which provides additional information regarding different *visual concepts* (such as architecture, food, animals) that are present in each photo. Each of these concepts is called an *autotag* and has a value ranging from 0 to 1 assigned. Those *autotags*, and the corresponding assigned values for each of the photos, were generated following a deep learning scheme. More specifically, binary Support Vector Machine (SVM) classifiers were trained on features from the AlexNet, proposed in [43]. The training set of the SVM consisted of 15 million crowd labeled Flickr images. Each trained classifier provides a confidence score between 0.5 and 1, whereas if the score was below 0.5, the concept was not considered to be present in the photo. Those annotations constitute the *autotags* information which we leveraged, and the total count of the concepts was 1570. Therefore, each photo could be represented as a vector of 1570 dimensions (one dimension per autotag).

Due to the sheer size of the initial dataset (>99 million photos), we opted for one of its subdatasets. Specifically, we used the dataset available for the placing task of the MediaEval 2016 Benchmark[22]. This subset of the dataset contains in total 5,016,634 photos from 171,850 users.

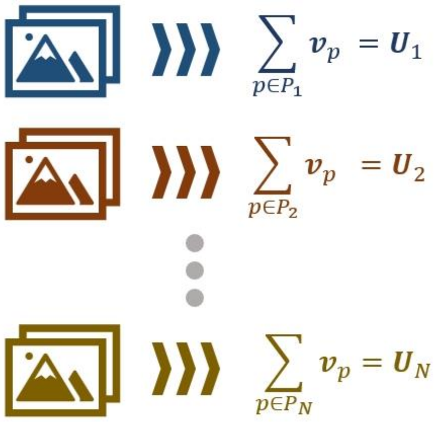

$\sum_{p \in P_1} v_p = U_1$
$\sum_{p \in P_2} v_p = U_2$
$\sum_{p \in P_N} v_p = U_N$

Figure 15. User vector calculation scheme. The vector representation of each user arises from the addition of the autotags vectors of all the photos of the user. U_1 ,U_2,... ,U_N represent the vectors for each user, while v_p corresponds to the vectors of each photo. P_1, P_2, ...,P_N refer to the sets of photos of different users

In order to extract a user profile from the above vectors, we performed vector addition over all the photos of a user, and we did the same for all users. This scheme created a vector of 1570 dimensions for each user, in which each dimension represents the amount of presence of the

[20] A set of labels manually assigned to a photo by the user.
[21] https://multimediacommons.wordpress.com/yfcc100m-expansion-packs/
[22] http://www.multimediaeval.org/mediaeval2016/placing/index.html

corresponding *visual concept* in the photos of that user. This procedure is better illustrated in Figure 15.

The last step was to estimate the similarity among the user vectors, so that we can derive a ***N***x***N*** user similarity matrix, where ***N*** denotes the number of users. To do so, we calculated the cosine similarity for each pair of users, according to

$$S(U_k, U_j) = \frac{\sum_a U_k(a)U_j(a)}{\sqrt{\sum_a U_k^2(a)}\sqrt{\sum_a U_j^2(a)}},$$

where U_k, U_j denote k-th and j-th users respectively and a denotes each one of the 1570 autotags.

It is worth noting that with the above formula, we get a symmetric metric for the user similarity task, meaning that the produced similarity matrix will have the same value at cell [i, j] and [j, i]. This process is represented in Figure 16:

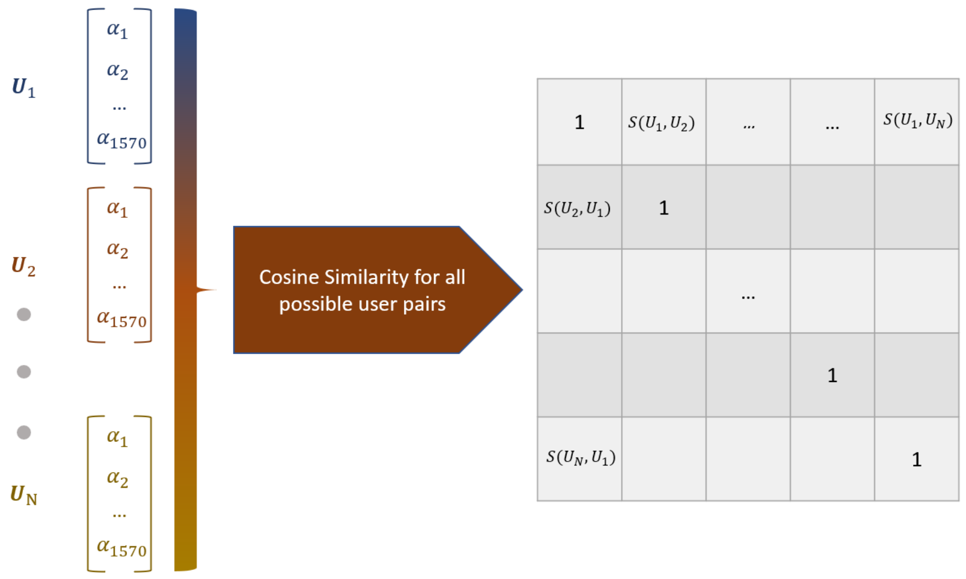


Figure 16. Generation of the symmetric user similarity matrix, based on the user autotags vectors.

4.3 User Profiling Training with Deep Metric Learning (DML) and Triplet Loss

As discussed in Section 2.4, Deep Metric Learning (DML) is focused on learning a distance function to measure the similarity between data samples. It does that by approximating an embedding function that maps samples in a feature space where relevant samples are closer than irrelevant ones. In our case the samples correspond to the users and the embedding function will map the users to their learned profiles. The user similarity labels we calculated from the autotags expansion pack of the YFCC100m dataset, as described in the previous section, will serve the purpose of revealing the relevant as well as the irrelevant users with respect to an anchor user.

An anchor user along with a similar user, referred to as the positive sample, and the dissimilar user, referred to as the negative sample, constitute a triplet. These triplets are the input to our model at each training step and as such they are pivotal to our discussion; in particular we will delve into how to define an appropriate loss on them, which we will call the triplet loss and what mining strategies to use.

Denoting the anchor, positive and negative samples as x, x^+ and x^-, intuitively an appropriate loss function would take high values when x^- is closer to x than x^+ is and low values when the opposite happens, ideally with a reasonable margin separating the two. A formulation that reflects this idea is

$$L_{tr} = max(0, D(x, x^+) - D(x, x^-) + m) \tag{4.1}$$

where D is the distance function we approximate with our model and m > 0 is a margin parameter to ensure a sufficiently large difference between the anchor-positive and anchor-negative distances. This loss function is known as the triplet loss.

Having settled on the triplet loss as our loss function, we need next to consider an appropriate sampling strategy or in other words how we will generate the triplets at each iteration of the training process. This is a less straightforward matter as there are a lot of reasonable solutions that are hard to objectively evaluate and thus more than one can be used at occasion. First, let us point out that training on all the possible triplets, which is of order N^3, is infeasible and thus we have to be more selective. The triplets we want the most are the ones that do not trivially satisfy the loss constraint, but rather violate it and thus provide valuable feedback to the training process. We specifically chose to user semi-hard triplets, which are the triplets (x, x^+, x^-) that produce

$$0 < L_{tr}(x, x^+, x^-) < m \text{ (semi-hard rule)} \tag{4.2}$$

that is, samples that do satisfy $D(x, x^+) < D(x, x^-)$, as desirable but not within the appropriate margin m.

The triplets are created online and so it is also important to ensure that each batch has enough semi-hard examples. For this reason, we use a large batch size of 512 and create the triplets by first selecting all the (x, x^+) pairs within the batch and then for each such anchor-positive pair we select a negative sample x^- such that the semi-hard rule is satisfied.

What we have left unspecified until now is how the positive pairs (x, x^+) and the negative pairs (x, x^-) are defined in our case. As we have explained, these stem from the user similarity labels, created based on each image's autotags, but these labels only assign a similarity score between $0$and 1 for each user pair. We then need to define a threshold to translate these scores to positive and negative examples. We heuristically chose to consider all the user pairs with similarity score above 0.8 as positive and those with similarity below 0.4 as negative.

The last part we need to specify is the architecture of the model that will approximate the distance function D between the users and is shown in Figure 17. The primary input to our model is each user's images, from which we will extract features with a pretrained CNN. So, each user will be associated with a number of vectors which we will need to summarize by taking their mean in order to minimize the computational overhead. This is the first part of the network, which we will not train specifically for the task, which although possible, the expected benefit seems to not be high enough as the number of users in our dataset, approximately 60,000, is not large. The trainable part of the network is the fully connected network that follows and consists of three linear layers with ReLU activations in between. The output of this network is the user embeddings, which we will use at inference as the user profiles, and the distance between the embeddings is defined by taking the usual Euclidean distance.

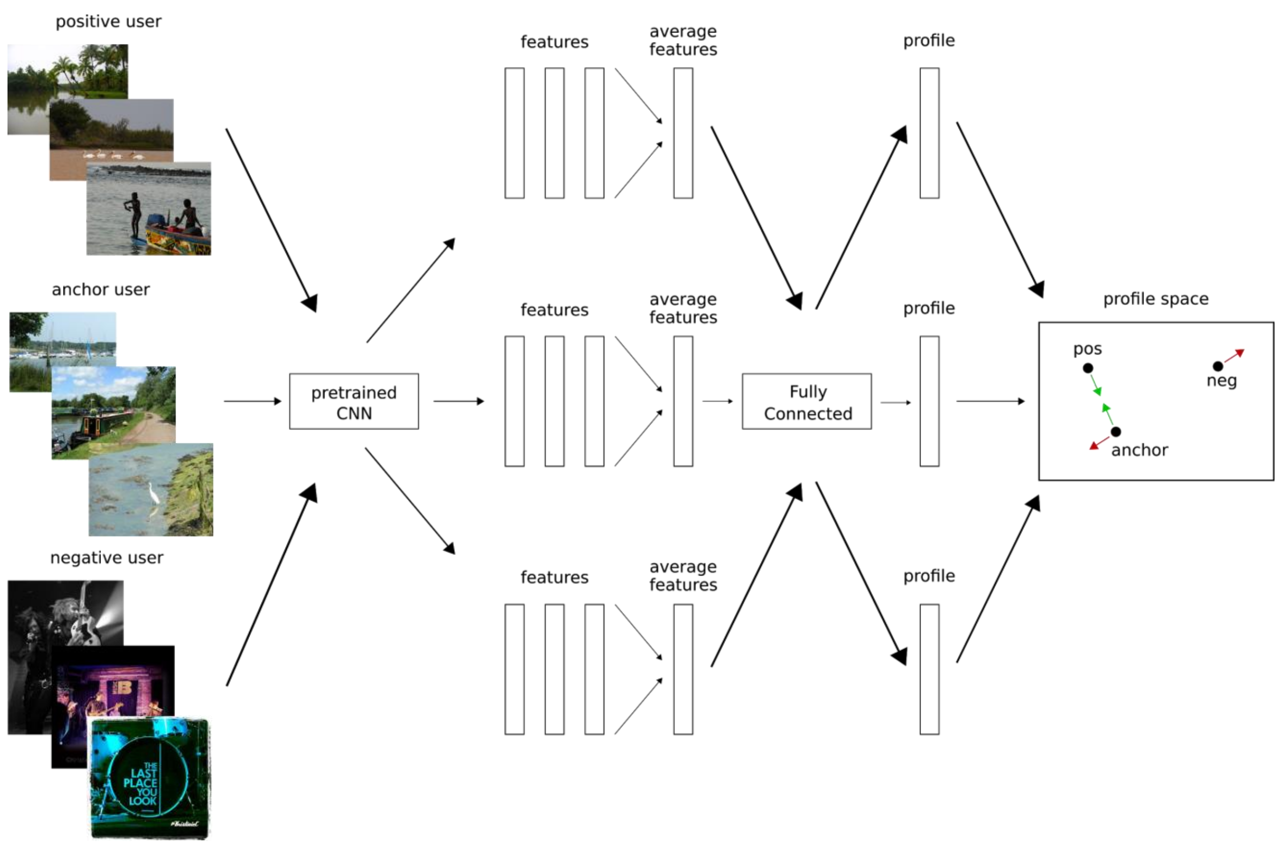


Figure 17. The training pipeline of our Deep Metric Learning model. We first construct triplets based on an anchor user, a positive similar user and a negative dissimilar user. We pass their images through the model, construct their profiles and compute the triplet loss which motivates the model to represent similar users close in the profile space and push dissimilar users apart.

The model was implemented using the PyTorch[23] framework and trained on a single GTX 1070 card. The features from the images were computed offline using the EfficientNet-B3 pre-trained CNN model and averaged for each user. The fully connected module takes as input these features and consists of a linear 1536×512 layers followed by a ReLU activation, another linear 512×256 layers followed by a ReLU and a final 256×256 linear layer. The model is trained for 12 epochs with 10^{-3} learning rate.

4.4 Results

In Figure 18 the training and validation curves for the average non-zero triplets are displayed. This metric measures how many triplets on average in each batch are associated with a positive loss. These triplets have an anchor-positive distance that is larger than the anchor-negative distance minus the margin, thus violating the desirable state. During the training the model learns to construct more appropriate embeddings and thus the non-zero triplets are decreasing. Because we use a large batch size of 512 samples, we expect the number of non-zero triplets to be high in absolute value as there are a lot of triplets being generated.

[23] https://pytorch.org/

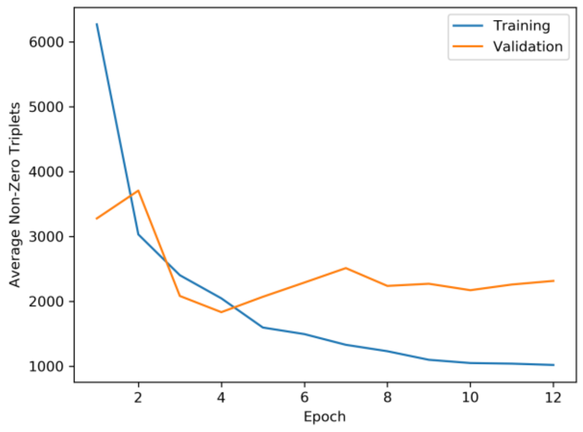


Figure 18. Average Non-Zero Triplets after each training epoch.

To provide some further insight to our algorithm, we devised an additional evaluation scheme. From the left out validation set we collected the 1,000 most common tags and created a bag of words representation of each user. This time we did not use the machine generated tags, but rather the user defined tags that accompanied the images. Based on these tags we calculate the Jaccard similarity between two users as the ratio of the size of the intersection to the size of the union of the tags of a user pair. We are interested in measuring the Jaccard similarity, as previously defined, between each user and the k-th most similar user according to our model as this is calculated from the euclidean distance between the user embeddings. We expect that the Jaccard similarity will have a decreasing trend as k increases as the users become more dissimilar and this is also reflected in their tags. Our hypothesis is indeed validated in Figure 19.

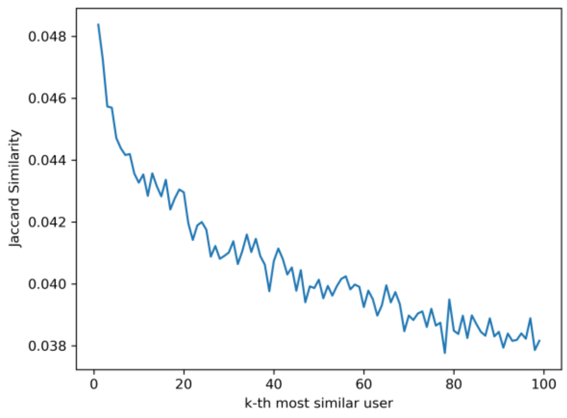


Figure 19. Jaccard similarity averaged over all users and their k-th most similar user pair according to the MobileNetV2 coarse interests model.

5 Text based Profiling through Text Extraction from User Images

5.1 Motivation and Contribution

Up until now, our focus was on profiling users based on the semantic cues present in their images. However, there is still a valuable source of information that both previously mentioned methods fail to capture; that is textual data. It is not uncommon for user images to contain textual information and while CNNs have shown remarkable ability to extract semantic information from images, it is unreasonable to expect that a network trained on a more general purpose image dataset would be able to sufficiently respond to textual cues.

In general, text can be found both in natural scenes, for example in inscriptions and signs, and document images. On the one hand, analyzing text in natural scenes could on one hand reveal some user preferences concerning clothing, food and entertainment or even the user's job, however, it is a considerably more challenging task that in many cases would not yield useful results [44]. On the other hand, document images are easier to analyze and optical character recognition systems enjoy high success rates. They are also quite common on the Internet and therefore possible to end up in a user's image collection if we broadly include in this category text excerpts or quotes, guides and information brochures or even media posts and advertisements. As such, exploiting textual content in these kinds of images could be beneficial to the construction of a more nuanced user profile and worthy of our study.

We use FastText - developed by Facebook, a frequently used library for text classification. It is specifically developed for fast processing on a variety of devices, including mobile. The library also provides pre-built models which can be supervised on unsupervised. It allows hyperparameter autotuning which automatically determines the best hyperparameters for the data. One of the disadvantages of methods such as word2vec and GloVe is that while they provide viable representations for words observed during training, they fail to yield embeddings for out-of-vocabulary (OOV) words — words that were unseen at training time. To do this, we use the package Misspelling Oblivious Embeddings[24] based on fasttext with a supervised task that embeds misspellings close to their correct variants. With this misspellings on text from images, as well as acronyms and others can be vectorized and learned in space.

5.2 Text Extraction from User Images

In this section, we will describe the method we used to extract text from user images. The images we are concerned with do not include natural scenes but are rather limited to posters, infographics, quotes or document images. This problem is generally known as Optical Character Recognition (OCR) and it is an old topic that has received a lot of attention through the years and still does as better tools and techniques emerge.

For the purposes of our task we are going to use the open source Tesseract OCR engine[25], which is well established with almost 35k GitHub stars and a rich history as it was open source

[24] https://github.com/facebookresearch/moe
[25] https://github.com/tesseract-ocr/tesseract

in 2005 by HP and since 2006 is developed by Google. There have been 4 major releases, the latest of which is based on LSTM networks.

Figure 20 shows the processing pipeline; first, we binarize the input image to increase the contrast and make the OCR task easier and then we feed the binarized images in the Tesseract OCR engine. The output of the engine is a text file with the recognized words ordered as detected including new line separations. Sometimes, the engine fails to properly recognize some words as it does in the given example for the site’s name, calligraphically printed, at the bottom of the image. In general, we can feed any kind of image to the OCR engine, but for images with no clear text the output will be blank or gibberish that we can easily filter out with a dictionary.

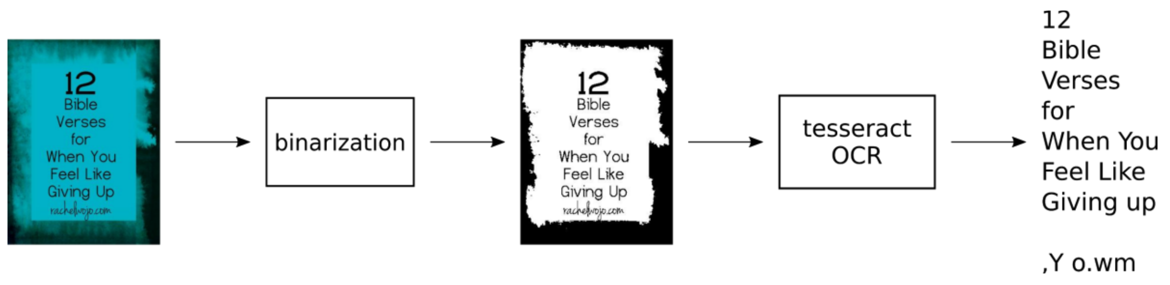


Figure 20. Example of the text extraction procedure. First the image is binarized and then passed to the Tesseract LSTM based OCR engine. It is possible that the output includes incomprehensible words that we will need to clean up later.

5.3 User Classification through Text Embeddings

The extracted text from the images first needs to be pre-processed and all the stop words to be removed. Then, the text is stemmed and lemmatized. A simple frequency analysis of words based on the categories is performed.

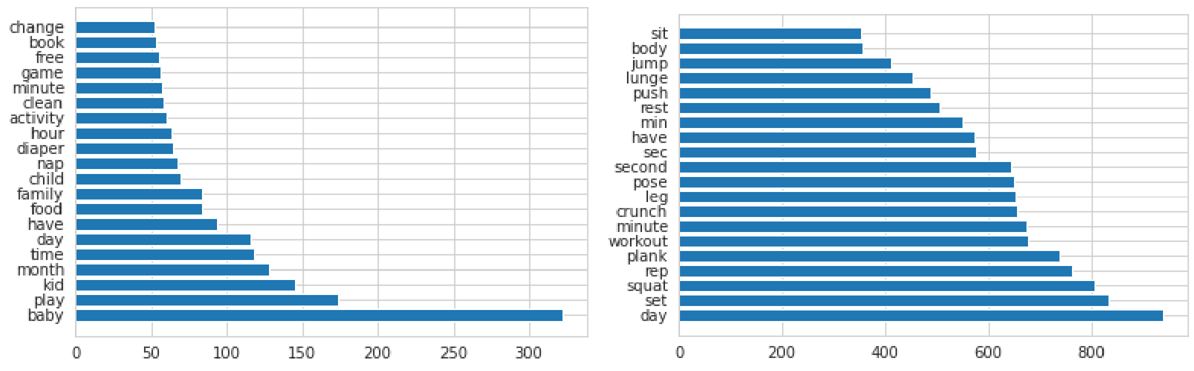


Figure 21. Top 20 most frequent words for the categories *kids_babies* (left) and *health and fitness* (right).

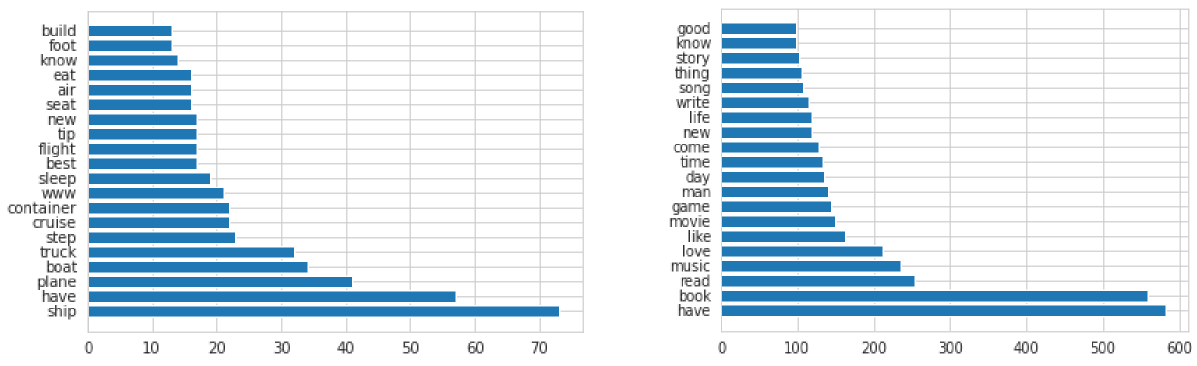


Figure 22. Top 20 most frequent words for the categories *vehicles* (left) and *entertainment* (right).

Two users can be compared based on the 20 most common words of a particular category over a predefined period of time. With time we can compare word variations between users as time progresses.

We can furthermore model users' interests based on their inclination towards active topics (interests) representing the text as a bag-of-words and then apply cosine similarity to determine the similarity between the users in order to infer common interests.

Latent Dirichlet Allocation (LDA) is one of the well-known unsupervised techniques used for identifying latent topics from a corpus of documents. However, being designed for regular documents, it may not perform so well on short, noisy and informal texts and might suffer from the sparsity problem. It is used for topic modeling, helping us to understand collective behavior and latent communication structures. Of particular interest is whether such analysis can elucidate the latent behavior of users. Generally, LDA detects related topics from associated and co-occurring elements within a collection of documents. Such latent topics have a probability distribution over words, and the underlying strategy is to assign each word to a topic in the corpus with a varying degree of membership. In other words, LDA can produce multiple topics for a single post and each topic is defined by a distribution over words, taking into account that ordering of words is ignored and that the words in each document are known. As highlighted by Cambria and White in [45] LDA "involves the use of machine-learning techniques to perform semantic analysis of a corpus by building structures that approximate concepts from a large set of documents" without the need to rely on external knowledge bases. Essentially, the distribution of topics altogether, as well as the distribution of topics for each document is learned from the data. The topic distribution is then represented as a vector, which is used for computing the distance between the documents, which then provides information about their similarity. Stated differently, similar documents (posts) will contain similar topics distribution and thus be closer in the vector space, which can be used to identify the relationship between documents (posts).

There are two specific measures to be considered with LDA modeling. The *Alpha parameter* represents document-topic density - if one uses a higher Alpha parameter it is assumed that a post is made of more topics and results in more specific topic distribution per document. The *Beta parameter* - represents topic-word density, higher Beta parameter means that topics are thought to have been made up from most of the words and therefore result in a more specific word distribution per topic. For the symmetric distribution, a high alpha-value means that each document is likely to contain a mixture of *most* of the topics, and not any single topic specifically. A low alpha value puts less such constraints on documents and means that it is more likely that a document may contain a mixture of just a few, or even only one, of the topics.

If, on the other hand, the distribution is asymmetric, a high alpha-value means that a specific topic distribution (depending on the base measure) is more likely for each document. Similarly, high beta-values means each topic is more likely to contain a specific word mix defined by the base measure.

Topic Coherence. A single topic is measured for the degree of semantic similarity between high scoring words in the topic. This helps to distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference.

To increase the performance of the model, tuning of the alpha, beta parameters, for all topics was performed and topic coherence was measured. Table 11 lists some of the results:

Table 11. Topic coherence for different values of alpha and beta.

Topics (N=20)	Alpha	Beta	Coherence
2	0.01	0.01	0.358735726296909
2	0.01	0.31	0.349102749788402
2	0.01	0.61	0.367154566595308
2	0.01	0.91	0.39359050640169
2	0.01	symmetric	0.312659448637189
2	0.31	0.01	0.327571571648674
2	0.31	0.31	0.366281616437541
2	0.31	0.61	0.354291438472752
2	0.31	0.91	0.34142737351088
………….	………….	……..	…………..
19	0.91	symmetric	0.437175319498444
19	symmetric	0.01	0.416960021675226
19	symmetric	0.31	0.54818077250476

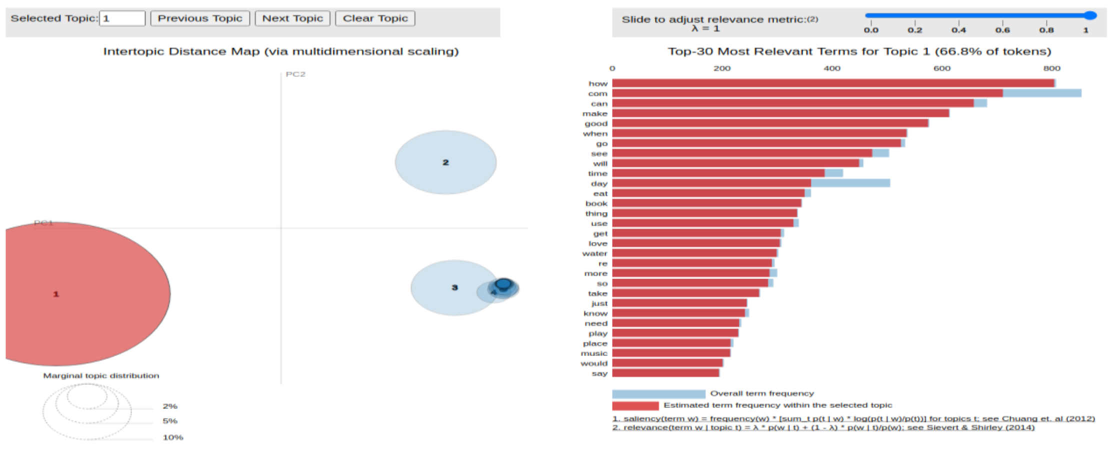
19	symmetric	0.61	0.582774999765285
19	symmetric	0.91	0.539397403985694
19	symmetric	symmetric	0.454346058715738
19	asymmetric	0.01	0.406728813396778
19	asymmetric	0.31	0.554312884824434
19	asymmetric	0.61	0.536115094669246
19	asymmetric	0.91	0.471435213631262
19	asymmetric	symmetric	0.465416835640429

Figure 23. shows the results of the analysis on three categories.. The text from all categories was combined together and analysed without the Pinterest categories. A different strategy comprises selecting each category by itself and carrying out LDA analysis per category. This is useful in the presence of several subcategories, as is the case in category “entertainment”, “sports”, “fashion”, etc.

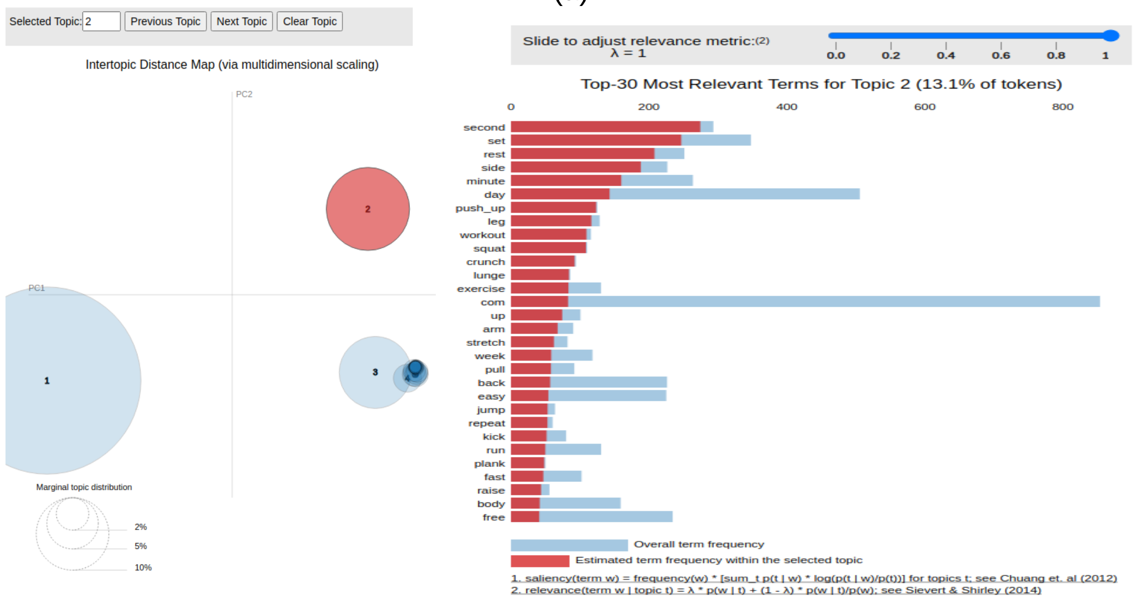
The right hand side of the plots represent the relevance of a term (the 30 most used terms) to a topic, given a weight parameter, $0 \leq \lambda \leq 1$, as

$$\lambda \log\big(p(term|topic)\big) + (1-\lambda)\log\left(\frac{p(term|topic)}{p(term)}\right)$$

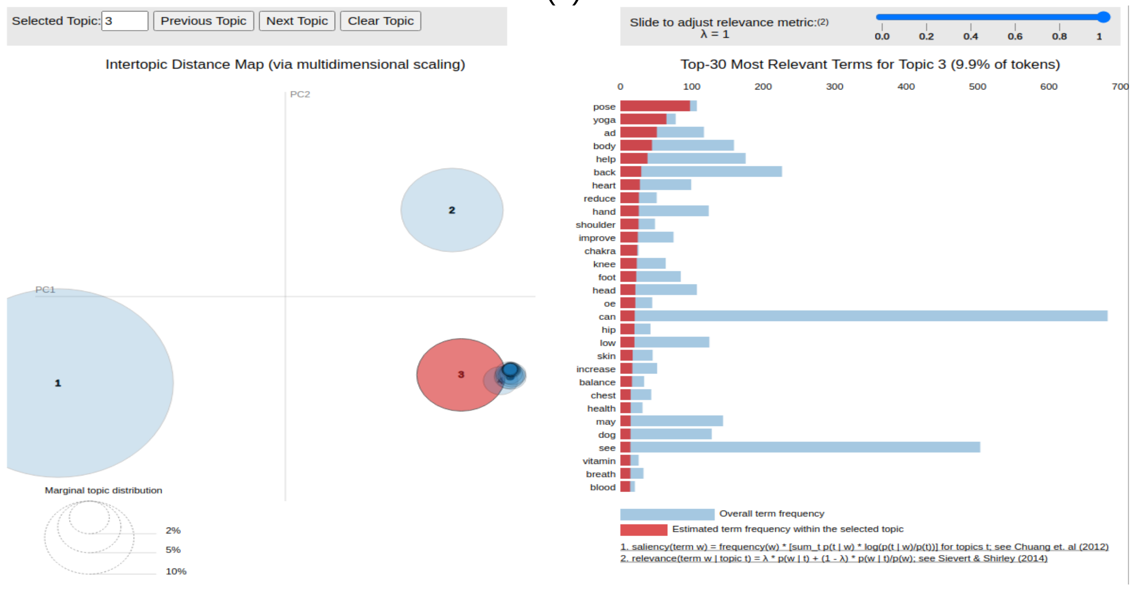
The red bars represent the frequency of a term in a given topic, (proportional to $p(term|topic)$), and the blue bars represent a term's frequency across the entire corpus, (proportional to $p(term)$). The left hand side plot depicts the topics as circles in the two-dimensional plane whose centres are determined by computing the Jensen–Shannon divergence between topics.


Selected Topic: 1
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 1 (66.8% of tokens)
0 200 400 600 800
how
com
can
make
good
when
go
see
will
time
day
eat
book
thing
use
get
love
water
re
more
so
take
just
know
need
play
place
music
would
say
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

(a)


Selected Topic: 2
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 2 (13.1% of tokens)
0 200 400 600 800
second
set
rest
side
minute
day
push_up
leg
workout
squat
crunch
lunge
exercise
com
up
arm
stretch
week
pull
back
easy
jump
repeat
kick
run
plank
fast
raise
body
free
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

(b)


Selected Topic: 3
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
Marginal topic distribution
2%
5%
10%
Top-30 Most Relevant Terms for Topic 3 (9.9% of tokens)
0 100 200 300 400 500 600 700
pose
yoga
ad
body
help
back
heart
reduce
hand
shoulder
improve
chakra
knee
foot
head
oe
can
hip
low
skin
increase
balance
chest
health
may
dog
see
vitamin
breath
blood
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

(c)

Figure 23. Results of LDA analysis on (a) topic 1, (b) topic 2, and (c) topic 3.

5.4 Combining Text-Based and Interest Profiling

In this section, we described a mechanism that aims to match users based on common LDA topics of words extracted from their images. This process is similar to extracting user interest profiles in that it provides a conceptual understanding of user images. The main difference between LDA topics and user interest profiles is that the former allocate the general-purpose concepts pertaining of natural language terms into different learned topics, whereas the latter aim to learn embeddings that can be used to predict specific interests.

These above two approaches effectively utilize complementary types of information match users. Hence, we consider an improved mechanism that combines them to achieve a multi-faceted understanding of which users to match. To do so, we consider an aggregation mechanism between their matches features, in which the similarity between two users A and B can be computed as a weighted aggregation of their topic and interest similarities:

$$sim(A,B) = \sum_{F \in LDA\ topics} w_F P(F \in A) P(F \in B) + \sum_{I \in interests} w_I P(I \in A) P(I \in B) \tag{5.1}$$

where $P(F \in A) = \left\{ \frac{1}{\sqrt{|F : F \in LDA\ topics, F \in A|}}\ if\ F \in A, 0\ otherwise \right\}$ is the L2-normalized indicator function of whether a topic F is assigned to user A,

$$P(I \in A) = \frac{p_A[i]}{\sqrt{\sum_{i\ \in interests} p_A[i]}}$$

where p_A is the profile of user A, is the same indicator function for interests I and w_F, w_I are the weights placed on all topics F and interests I respectively. If $w_F = 0.5$ for all topics F and $w_I = 0.5$ the above equation is reduced to averaging the cosine similarity between topics and the cosine similarity of interests.

Given this scheme, in the next period we aim to learning parameters w_F and w_I over a training dataset of matched users.

6 Context Aware User Profiling and Matching

6.1 Spatio-Temporal Context Aware User Profiles

An important feature of HELIOS is that it operates in multiple spatio-temporal contexts, as described in D4.1; users can switch between contexts in which they perform social actions. This enhances user experience in HELIOS applications by allowing them to respond to changes in their perceived environment. However, to claim such responsiveness, all the supporting HELIOS features should also be context aware and of course user profiling and matching is no exception. In this section, our goal is to discuss how spatio-temporal awareness can be incorporated into the user profiling models described in Sections 3, 4 and 5.

While two users can be very different if all their photos are taken into account, conditionally on a specific spatio-temporal context they could be a lot more similar. An example of this case is illustrated in Figure 24. We have illustrated the images of two users from the YFCC100m dataset partitioned according to the place and the season, summer, autumn, winter or spring, that the image was taken. The first user predominantly has photos from Finland and most of them show an interest in music. This is especially true for autumn, winter and spring, but in summer there seems to be more variation with images taken also in Portugal, Spain and France. The theme of the images is also deviating from the rest of the photos as the ones from Spain and France show an interest in architecture and sightseeing, while the ones in Portugal are connected with the sun and the beach.

On the other hand, the second user seems to be travelling throughout the year and not specifically in summer. Their images overall show an interest in architecture and sightseeing with also an occasional visit to Guadeloupe where the sea and the beach and the scenery seem to be appreciated.

In general, their profiles are different and we would not consider these users similar, but there is a particular context where their interests seem to align, that is the spatio-temporal context that we could define as “summer in Paris”. It would be useful to be able to recognize this similarity and thus we propose the use of a straightforward attention mechanism over the images that prior to calculating the user similarities would weight the images according to the date and time taken and the currently active spatio-temporal context. There are many reasonable approaches to define the appropriate weights, but we propose a binary one[26] where the spatial context would be defined as within a city range, approximately a radius of 25 km, from the current location and the temporal context would be multi-valued according to whether we distinguish between seasons, weekdays or times of the day, such as morning, afternoon and evening. In this way, conditioned on the current spatio-temporal context we would calculate multiple profiles corresponding to variations in the temporal context definition.

Returning to the previous example, our MobileNetV2 coarse interests model would output if given all the images a cosine similarity between the users of approximately 0.05. However, if we set the current spatio-temporal context as summer in Paris and calculate the conditional user

[26] We could also consider defining a smoother distribution that decays gradually the further away we are from the current location. The distribution could be Gaussian with an appropriate standard deviation controlling the cut-off point or even a von-Mishes Fischer that has some additional nice properties like symmetry when the data are spherical.

profiles, the cosine similarity would quadruple, reaching 0.2. We should note that we are interested in the ranking of the user similarities rather than the absolute numbers and thus the context could potentially have a big impact on user similarity.

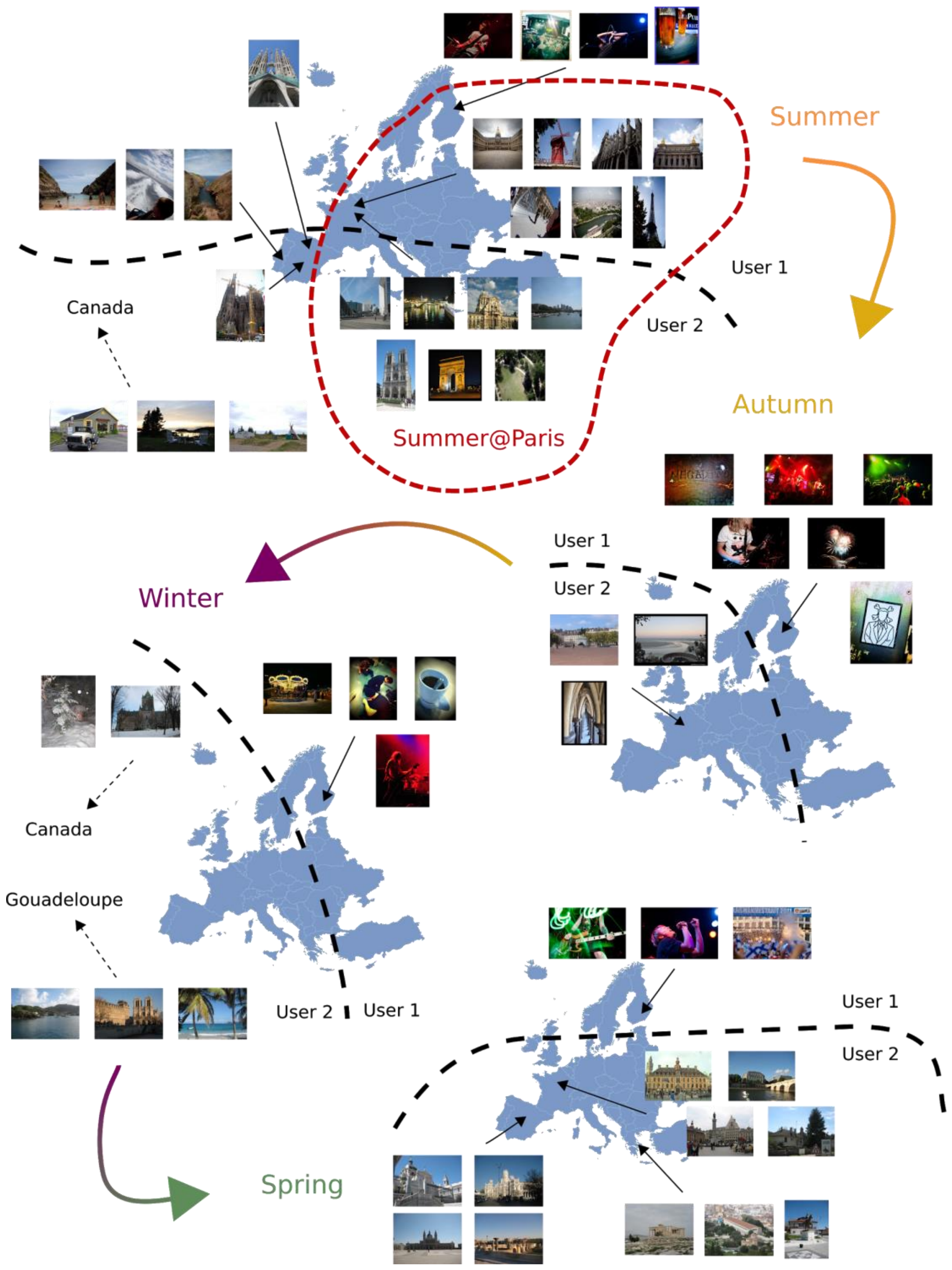


Figure 24. Illustration of two YFCC100m users' photos as they are distributed over the time and place where they were taken. We partition the time in seasons and we depict most locations in a map of Europe. The first user's photos are displayed on the right of the black dashed line as it is traced from right to left. The red dashed line emphasizes the spatio-temporal context Summer@Paris where they are most similar.

So far we have assumed that the spatio-temporal context was given by the platform, but it would be interesting if we could also discover such contexts without supervision. We are specifically interested in contexts that reveal similarities between the users that fade out when considering their whole photo collection. A way to discover such contexts is by maximizing the cosine similarity between the content aware profiles of two users with respect to the attention vector over the images. An attention vector that achieves a local maximum similarity can be considered as a context under which two users are particularly similar. Mining the places and dates of the attended images we can then attempt to produce a description of the context meaningful to humans. Although an intriguing idea, it is still in a preliminary research stage. In particular, carrying out the previously mentioned optimization problem in a decentralized way would require some form of communication and cooperation between the two nodes that would raise concerns about the convergence of the optimization algorithm and the computational cost of the process.

6.2 Evaluation of User Matching

In this section we will evaluate the proposed algorithms for the task of user matching.

Dataset

To evaluate the effectiveness of our user profiling algorithms at the task of user matching we need to have a dataset consisting of users along with their images and annotations of the connections between them. As we have already discussed, such datasets are not available and therefore we created one from Pinterest using a data extraction method. The users were chosen with a snowballing method from the following and follower links and based on those, the dataset was also annotated. The connections between the users were considered symmetrical meaning that both follower and followee relationships were treated identically. We note that a pair of users is flagged as connected if at least one of them follows the other and we make no distinction in the case that they follow each other.

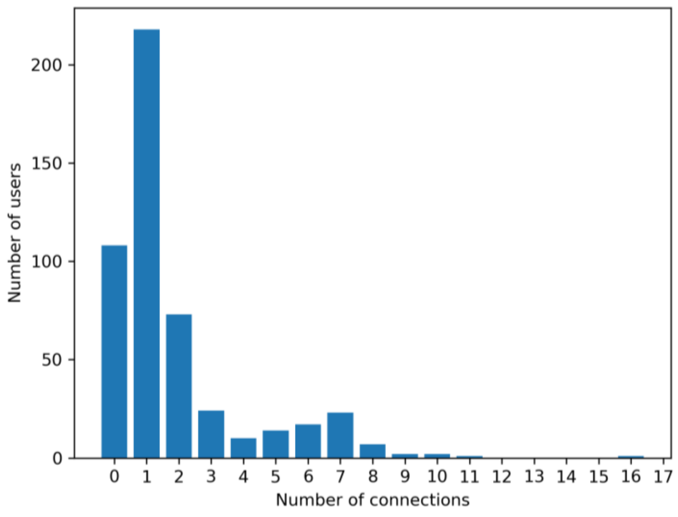


Figure 25. Histogram of user connections from the Pinterest matching evaluation dataset.

We initially collected 500 users, but we removed 78 of them that had more than 10,000 images, as they would considerably increase the number of images needed for download and more importantly they frequently belonged to companies rather than individuals. So, the final dataset consisted of 422 users with approximately 210,000 images. However, these removals created a situation where more than 100 users were reported to have no connections at all, thus making the dataset challenging but considering that it is indeed possible that users are not connected with any other user, we decided to retain them. A histogram of the user connections is shown in Figure 25.

Experiments

To evaluate the performance of both the interest categories based model and the Deep Metric Learning one we first calculated the profiles of the 422 users, who were completely new and not previously seen by any of the models, and then the similarity scores between all pairs using cosine similarity. The top-k matching recommendations for one user will then correspond to the k other users that have the highest similarity score with that user. At each k level we are interested in measuring the model's recall@k, that is the proportion of the follower-followee relationships that we discover in the top k recommendations. This value lies in the range [0,1], where higher values indicate better recommendation capabilities. However, even small values (e.g. higher than 0.1) can show the matching to be of practical use, as a user looking at a recommendation list of length 10 would be expected to find at least one recommendation of interest. In this way, the recall@k curve is shaped and depicted in Figure 26 for k up to 20. We also included for comparison a random baseline that was calculated by assigning a random order to the user recommendations.

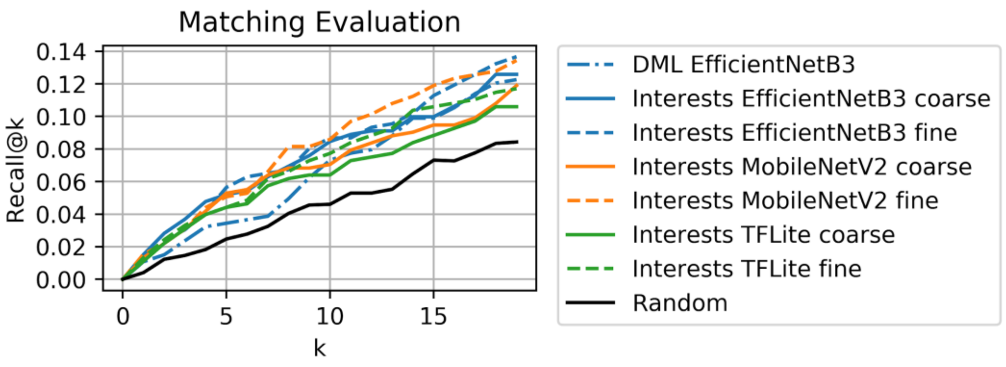


Figure 26. Recall@k for the interest categories based models and the DML one. The black line represents the random baseline.

As we can see all methods outperform the random baseline, but the one struggling the most for $k < 10$ is the DML model based on user similarities. We should, however, keep in mind that the Interest models were all trained on a dataset extracted from Pinterest, as was this particular connections dataset, while the DML model was trained based on a subset of the YFCC100m dataset which was created from Flickr data. Although both Pinterest and Flickr are popular image sharing platforms, the distributions of their hosted images significantly differ, making the comparison favourable for the interest categories based model.

From the plot in Figure 26 we can see that for $k = 5$ the MobileNetV2 implementation of the interests model achieves 5% recall, while the random baseline is 2.5%, and for $k = 10$ the former achieves 8.6%, while the latter 4.6% This indicates that the proposed user matching approach does significantly better (87-100%) compared to the random baseline.

Note also that the absolute recall values in the task should not be interpreted as realistic estimates of a real-world user recommendation module. This is due to the fact that the collected user profiles have very few connections among them (mostly just 1-2), thus recommending 5 or 10 connections is surely going to end up with low recall values (e.g. for a user with one known connection, recommending 10 connections would lead to a top-10 recall of either 0 or 10%). This does not mean that the remaining recommendations are not relevant but that we do not have information to assess their relevance.

7 Integrating Profiling in Social Graph Mining

In this deliverable, we have analysed mechanisms that help understand how user content pertains to profiles of interests. Those mechanisms can directly associate users with their interests and we have seen that they can also be adapted to match users of similar interests.

An important aspect of user interest profiles is that they reside on devices that are effectively nodes of the Heterogeneous Social Graph described in Deliverable 4.2. Then, this graph can be enriched by considering its nodes to also hold the interest profiles extracted through the processes described in this deliverable. The outcome of this enrichment can be considered to be content- aware in that it captures the user interests pertaining to the textual or image content of each user.

In Deliverables 4.2 and 4.3 social graphs were considered to comprise nodes V and edges $E \subseteq V \times V$such that edges $(u, v) \in E$ indicate relations between users $u, v \in V$. Then, the content-aware enrichment can be thought of as a process that generates user interest profiles vectors U_v for nodes v through their on-device content, where vector elements $U_v\,[i]$ correspond to how much each user v is estimated to pertain to interest i.

Under this formalization, user interest profiles are essentially a type of information that is accessible to graph nodes and, as such, can both be mined to reveal important structural organization of the graph and be the target of mining that aims to understand latent user preferences and how these pertain to user relations. These two types of analysis strongly relate to the social graph mining practices analysed and developed in Deliverable 4.3. Hence, our subsequent analysis builds upon the work of that deliverable.

7.1 Propagating User Profiles through the Social Graph

The first type of analysis we conduct relies on the notion of homophily described in Section 2.6 which indicates that neighbors of social graphs often exhibit similar interests. Motivated by this observation, we consider mechanisms that propagate user interests to their neighbors and move the latter's interests towards the propagated ones. This step can be performed multiple times across all user devices so that they all arrive at an unchanging understanding of user interests. In this case, each user's interests are indirectly propagated through their neighbors to their neighbors-of-neighbors and so on, until they are diffused in the whole graph.

This type of graph diffusion has already been investigated in Deliverable 4.3, but for the sake of completeness we will also present a formal description that better matches the understanding of this deliverable.

Graph Filters

The core building block of our analysis is the notion of diffusing each user's interests to their immediate neighbors. In the general domain of graph signal processing overviewed in the subsection 3.1 of Deliverable 4.3, we can write the operation of propagating a single vector s (also called a graph signal), whose elements $s[v]$ indicate the relatedness of all users v to a given interest, to the immediate neighbors as:

$$r_1 = Ws \Leftrightarrow r_1[u] \ = \Sigma_{v\in V}\, W[u,v]s[v]$$

where W is a square matrix whose elements $W[u,v]$ are proportional to how much u is influenced by user v. This operation is equivalent to propagating the following relatedness to the examined interest to nodes u, but does not necessarily perform an aggregation with the relatedness of the latter. It can also be iterated to propagate the relatedness to interests to users k hops away in the social graph:

$$r_k = W^k s \tag{7.1}$$

In this scheme, for $k = 0$ we obtain the initial estimations of the non-propagated user interests $r_0 = s$. Then, all propagations can be aggregated with corresponding weights a_k through a weighted averaging scheme:

$$r = H(W)s \qquad \text{where} \qquad H(W) = \textstyle\sum_{k=0}^{\infty} \quad a_k W^k \tag{7.2}$$

The quantity $H(W)$ is called a graph filter on merit that it can effectively translate the graph's propagation mechanism W to the Taylor expansion of any matrix function. Two popular graph filters are personalized PageRank, which is obtained for $a_k = (1-a)a^k$ where $a \in [0,1]$ is a diffusion constant and HeatKernels, which is obtained for $a_k = e^{-t}t^k/k!$ where $r = 1,2,...$ is a constant of how many hops away should be influenced more by the propagation.

Besides the parameters of the graph filter, a computational aspect that affects the outcome of this graph signal processing mechanism is the choice of propagation weights W. These need to be selected so that propagations are performed towards graph neighbors and the values of $W^k s$ remain bounded for arbitrarily large k. Hence, the propagation mechanism is typically chosen as normalization of the adjacency matrix M whose elements indicate whether the corresponding edge between nodes exists, i.e. $M[u,v] \ = \{1\; if\; (u,v) \in E, 0\; otherwise\}$.

In particular, the normalization typically employed by personalized PageRank models is a Markov process randomly walking the graph, and hence satisfies$\Sigma_{v\in V} W[u,v] = 1$. If D is the diagonal matrix of node degrees with diagonal elements $D[u,u] = \Sigma_{v\in V} M[u,v]$ and zero everywhere else, thIS normalization can be obtained by the formula:

$$W = MD^{-1} \Leftrightarrow W[u,v] = M[u,v]/\Sigma_{k\in V} M[u,k]$$

However, recent works on graph signal processing have moved to adopt a symmetric normalization that satisfies$W[u,v] = W[v,u]$ so as to relate to (the also normalized version of) the Laplacian operator, which is the graph equivalent to discrete derivation, by expressing the latter as $L = I - W$. This type of adjacency matrix normalization can be expressed as:

$$W = D^{-1/2} M D^{-1/2} \Leftrightarrow W[u,v] = M[u,v] \,/\, (\sqrt{\Sigma_{k\in V} M[u,k]}\, \sqrt{\Sigma_{k\in V} M[k,v]}\,)$$

Unsupervised Selection of Graph Filters

Based on the understanding of graph filters, we now explain how user interests can be propagated throughout the graph to improve the interests of other neighbors. To do this, we consider a matrix U whose rows U_v correspond to the interests of users v. Then, applying a graph filter described by Equation (7.2) on each of those interests separately is equivalent to obtaining an updated version of user interest profiles U' through the following equation:

$$U' = H(W)U \tag{7.3}$$

As a first step to devising graph filters for diffusing interest profiles, we conducted research [46], [47] over how to select the best ones for a given graph, such as the social networks developed on the HELIOS platform. In particular, we tackle the problem of evaluating user profiles through unsupervised procedure when actual profiles are unknown.

To this end, we argue that, since homophily frequently permeates many types of graphs, such as social networks, we can evaluate the quality of how much the relatedness between nodes of similar metadata attributes (i.e. in our case of common interests) can help reconstruct the graph's edges. If we used the dot similarity -or cosine similarity if a graph filter that performs L2-normalization is considered- to find the similarity of user interest profiles as $sim(u,v) = U'_u U'_v$, we can express an estimation $\widehat{M}$ of the network's adjacency matrix whose elements capture that similarity, i.e. $\widehat{M}[u,v] = sim(u,v)$, as:

$$\widehat{M} = U'U'^T = H(W)UU^T H^T(W)$$

Since this is a quadratic form around the initial profile similarities UU^T, requiring $sim(u,v)$ to perform a high-quality prediction of the network's edges after applying the graph filter is equivalent to performing a diffusion of the initial interest profile similarities throughout the graph's edges. As a result, $\widehat{M}$ can closely approximate the adjacency matrix M only if both the initial profiling process can correctly matches users based on interest similarities (although it is allowed to neglect matching users) and the filter $H(W)$ exhibits a good understanding of how interests need to be diffused throughout the social graph.

Given this theorization, we propose measuring how well filters of the social graph can enhance the quality of extracted user interest profiles by the AUC of predicting the non-diagonal entries of M using $\widehat{M}$, where the latter is obtained by the cosine similarities between the filtered profiles of U'. In our published research, we call this measure *LinkAUC* and experimentally show that most of the time it yields more accurate assessment of which graph filter to use compared to unsupervised evaluation measures of structural community quality, compared to the ideal ones that would be obtained by AUC and NDCG if the interests were known.

Hence, since the interests of social graph users are not known to facilitate supervised evaluation of graph filters, we resort to selecting the graph filters that maximize LinkAUC. This selection can be done offline on social networks with similar structure to HELIOS. However, our research has shown that sampling strategies of the graph's adjacency matrix can also lead to a good selection of the best graph filters. Therefore, we aim to further investigate in future work if the graph filters used to propagate the initial user interest profiles can be selected so that they work on a per-user basis in the decentralized setting of HELIOS. Investigating and implementing these practices will be done in the next period.

Applying Graph Filters to HELIOS

Given that we select the best filters with which to propagate estimated user interests throughout the social graphs, the next step would be to deploy those on HELIOS applications.

In doing so, we expect a qualitative and quantitative improvement of user interest estimations. In particular, the updated estimations better match the true underlying user interests that pertain to homophilous behavior. At the same time, the propagation mechanism of graph filters can help discover relatedness to interests that may not be captured by the explicit content available on devices. For example, if users feel uncomfortable in allowing profiling of their content, an estimation of their interests can still be achieved through their structural positioning in the social graph with respect to other people that exhibit those interests.

The challenge in implementing these practices lies in the decentralized nature of HELIOS platform, which hinders the direct implementation of Equation (7.3), as the initial user interest profiles are stored across different devices and there is no central organization to help gather and propagate them. As a result, it is necessary to follow decentralized adaptations that help approximate the outcome of graph filters. Previous research has proposed several methods in doing so for personalized PageRank [48]–[51], but requires on-demand or ongoing communication between user devices that may end taking a lot of bandwidth or fail as users go offline. In the next period we will also research adaptations of decentralized graph filters that can enhance interest profiles under this kind of dynamism, for example by adjusting the decentralized community detection protocol developed in Deliverable 4.3 or updating interest profile estimations based on those of only the interacting users.

7.2 Combining Social Graph Mining and Profile Matching

Besides the informativeness of discovering how much users pertain to interest profiles, in this deliverable we have also shown that such information can be used to match users so as to recommend social actions, such as relations. This recommendation relies on the similarity of user profiles, as this is calculated either through the cosine similarity of their vector representations or through a DML model of user similarity that outputs the euclidean distance between user embeddings.

Combining Similarity Scores

The above mechanism matches users based on their content. However, Deliverable 4.3 already defines similar recommendation mechanisms that consider the positional placement of users in the social graph to extract latent relational or interaction-related preferences with which to match them. Then, it is of interest to combine these two different approaches, that utilize different types of user data, so that more accurate user matching is achieved. In particular, the resulting matching mechanism would account for both the content-based interest preferences and the latent behavioral preferences of users.

The simplest method to achieve this is by performing a trade-off between user similarity scores produced by these two mechanisms. For example, if $sim_{inter}(u,v)$ and $sim_{pref}(u,v)$ correspond to interest- and preference- based matching scores respectively, these could be transformed through a function $f(\cdot)$ into a space where they can be linearly aggregated, to produce the similarity:

$$sim(u,v) = f^{-1}(\ a_{inter}f(sim_{inter}(u,v) + a_{pref}f(sim_{pref}(u,v)\ ) \tag{7.4}$$

where $a_{inter} + a_{pref} = 1$. A clear advantage of this kind of approach is that the selection of the transformation function (e.g. $f(x) = x, f(x) = ln(x)$) and the aggregation weights a_{inter}, a_{pref} can be chosen through offline training that needs not necessarily be aware of the fact that latent user preferences are different for different social graphs (e.g. between the graphs involved in selecting the aggregation parameters and the social graph of HELIOS) and between different experiments on the same graph.

Combining Similarity Features

Although combining similarity scores can easily integrate both interests and latent preferences of users, it does not necessarily take into account that these two types of user characterization can work cumulatively to provide an improved latent understanding of content and behavioral and interest preferences. In practice, this means that there could exist a latent space of user attributes that encapsulates these two types of understanding of the user. Then, we expect that user matching performed in that space to yield even higher recommendation capabilities compared to the only combining similarity scores.

An important shortcoming of this type of approach compared to other user matching mechanisms presented in this deliverable is that it heavily depends on the per-network extracted user preferences. Therefore, although additional neural layers can be used to combine the user latent preferences and interest profiles into a low-dimensional common understanding, these cannot be trained in an offline setting.

A first take in this direction, which we aim to explore decentralized training schemes of neural networks that would allow us to deploy such a scheme over the social graphs of HELIOS. A promising approach in this regard is to organize such types of mechanisms as (potentially multilayer) graph neural networks, as those are described in Deliverable 4.3, which learn to recommend social actions based not only on the time-evolving structure of the graph but also the outcome of context mining. These could potentially involve the principles developed in that deliverable on how to train such architectures by exchanging information only during interactions and regularizing towards similar parameters between neighbors.

8 Practical Aspects and Considerations

8.1 Mobile Deployment

As the models we have presented in this deliverable are meant to be deployed in a mobile environment, in this section we will discuss some practical issues concerning this process.

The framework we chose for the deployment of the deep learning models was TFLite. TFLite is part of the TensorFlow ecosystem and is geared towards packaging small and efficient models in a binary executable format. This binary can be executed from within Java code with the assistance of TFLite's Java API. In this way it can be embedded into an Android app and incorporated into the app's logic.

Apart from creating the executable that will be called from within the app, TFLite offers some more conveniences regarding the optimization of the models to be deployed. In a mobile environment both memory and computation are limited and to provide a good user experience we have to reduce both the size of the model and the calculations needed for inference. As discussed in Section 2.5 the optimizations that can provide the most immediate benefit and are easy to apply are weight and activation quantization and pruning, both of which are provided within TFLite. For our models we opted for weights-only quantization, but in the event that this proves to not be enough there is room for further resource savings. It should be noted, however, that each of these transformations introduces an accuracy penalty and we should achieve the right trade-off.

To be more precise, the models ready for mobile deployment are the interest-based ones described in Section 3. The DML model presented in Section 4 is currently developed only using as base the EfficientNet-B3 CNN, which is computationally demanding and not suitable for mobile deployment, but it is straightforward to convert to a more appropriate form as we would need to replace the EfficientNet-B3 with the MobileNetV2 CNN that we also used for the mobile versions of the interest-based models.

To assess the performance of the TFLite interest-based model in a realistic mobile environment, we created a standalone Android application that would run the model in question 10 times on a demo image and would report the mean inference time. The TFLite model was added as an asset to the application and required only 2.3 MBytes and the RAM consumption of the whole application was less than 40 MBytes. The results of running the application on four different smartphones are shown in Table 12.

Table 12. Average Inference Time for a single forward pass of the interests based TFLite model.

Model	Inference Time (ms)
Xiaomi Mi 8	140
Xiaomi Redmi 4	288
Xiaomi Redmi Note 5	290
Huawei Y7	400

The smartphones are all considered to be on the low end spectrum with the Xiaomi Mi 8 approaching the mid-range category and achieving 140ms mean inference time. So, in a scenario that a user has 100 images in his collection 14s would be needed to extract his profile. However, each image needs to be processed only once through the TFLite model and once we save the output, the calculation of the profile is very fast. Storage limitations in this case are not a problem as the output of the interests based model is a 15 dimensional vector in the coarse case and less than 50 in the fine. In the case of the DML model the output for each image is 256-dimensional which would require only 0.1 MBytes to be stored if we assume 32-bit floating point representation and 100 images.

In a mobile environment, equally important to the execution speed, if not even more, is the energy consumption of the model. Deep learning models are known to be power intensive as in order to speed up the calculations, they require probably more than one CPU core and even the GPU present in most of the recent smartphones. However, as discussed earlier if we cache the model's output, we do not need to run the TFLite model more than once on each photo and thus the required calculations could be scheduled in the background, preferably when the device is charging, and only for the newly added photos. This method works equally well for the context-aware profiles discussed in Section 6, as the attention vector is to be applied after the CNN has finished processing the images.

8.2 Privacy Considerations

The HELIOS platform adopts the principles of privacy-by-design and security-by-design, which are the cornerstones of the system's security. The decentralized nature of the platform makes accessing and collecting user data a challenging task. The data consumed by the Content-Aware profiling module to produce meaningful user profiles are located in the respective user devices.

To help protect user's privacy in the Content-Aware profiling module, all the analysis is carried out on the device and only a minimal amount of profiling data is shared with others. The module cannot directly expose the collection of images used to generate interest profiles or share profiling information with other peers, unless the user explicitly gives permissions. The Content-Aware profiling module is going to further protect the confidentiality of shared data by leveraging the results of the work produced by Task 3.4 and the resulting HELIOS security module.

At the application level, the user should be fully informed about the data that the Content-Aware profiling module processes and produces and should have control of whether these can be accessed, analysed and shared. Additionally, extracted user interests should be possible to edit and be approved by the user, and only be shared with others with whom a certain level of trust is established, as this is foreseen by the Trust Module that is currently being developed in Task 4.5.

Overall, given that the primary function of HELIOS is to enable social networking among people, sharing profile data between peers is inevitable. However, the design of HELIOS components accounts for the data privacy and security needs and aims at minimizing privacy risks that could arise in adversarial settings.

9 Conclusions and Future Work

9.1 Conclusions

This deliverable explored the problem of developing a content aware social graph. We proposed new techniques for adding content aware features on top of the on the Heterogeneous Social Graph (HSG).

We started with content aware user profiling through predefined interest categories. This led us to the design of a deep learning model capable of inferring both coarse and fine grained user profiles based on their photo collections. We trained the model with a novel dataset we created and tested it on Pinterest users with manually labeled profiles achieving 93.6, 88.5 and 92.5 mean AUC, MAP and NDCG respectively, using a small, efficient TFLite model suitable for mobile deployment. Furthermore, the content aware user profiles created in this way are interpretable as they correspond to humanely meaningful concepts, an important feature when aiming to be transparent with the users about how the platform algorithms work.

To address the disadvantage of relying only on predefined categories, we created an additional model with lower interpretability, but with the potential to capture more concepts with higher semantic coverage. This model was based on Deep Metric Learning and approximates a function that can map a user's photo collection to an embedding space where similar users are closer and dissimilar users are further apart. The model was trained on a subset of the YFCC100m dataset annotated with autotags and we demonstrated that there is correlation between the closeness of the users' embeddings and the similarity of the users based on the tags they provided to their photos.

Another kind of semantic information that can be available in an image and cannot be captured with either of the previous methods is textual information (e.g. memes, document snapshots). To extract this information a text processing model is needed. Therefore, we designed a model that can first extract the text present on images with an OCR engine and then perform textual analysis. To this end, the text is first pre-processed by removing stopwords, lemmatization and stemming. Then, word frequency analysis is performed, including TD-IF, bigram and trigram modeling. Finally, for the purposes of this report topic analysis is carried out through LDA. By performing topic analysis on the aggregated set of text items of a user, it is possible to construct text-based semantic profiles.

The previous models are able to extract varied semantic information from a collection of images, but HELIOS users also operate on multiple spatio-temporal contexts. For this reason, the content aware features should also be able to adapt to the different contexts and that is addressed by creating conditional user profiles. The conditional profiles are calculated with any of the previous models, but with an attention vector applied on the images that focuses on the most relevant ones to the active context.

To validate the proposed models at the task of user matching we created an additional dataset involving 422 Pinterest users and the connections among them. While the dataset is particularly challenging due to the very few connections present (mostly just 1-2 per user), we achieved 5% recall with the top-5 recommendations compared to the 2.5% of the random baseline, demonstrating that the produced recommendations are meaningful.

The models developed in this deliverable provide a solid foundation for the incorporation of content aware features in the HELIOS platform. We can provide multiple content aware user profiles that are complementary to each other and could be exploited in different HELIOS modules such as the social graph mining developed in D4.3. We are also in a position to provide content aware recommendations for user matching based on the constructed user profiles.

Finally, it is important to note that special care was taken throughout the development of this deliverable to design models that are suitable to be deployed in a mobile environment, that is they can be compressed in a small binary package, require modest computational resources and can operate in the background (and while the device is charging to limit power consumption). The architecture is also meant to respect the privacy of the users, there are no external servers involved in any step of the process and the data never leaves the user's device.

9.2 Future Work

Integration with the Helios platform

The Content Aware Profiling API, briefly described in the first annex, is part of the HELIOS platform extension modules. It has not been completed yet and as such a discussion of how it is going to be integrated in HELIOS is ongoing.

When deployed, the Content Aware Profiling Module requires access to the user's photo collection from which the content aware user profile will be calculated. We have described in this deliverable three different models, each emphasizing a different semantic representation of the visual content: the interest-based model focuses on the predefined interest categories, the DML model is adaptable to the provided data and the text model analyzes the text extracted from the images. However, it is not clear yet how the module will combine these three representations. It would be reasonable to provide the option to calculate the content aware user profile using any of them or even define an appropriate combination of them, but that would impact the overall size of the application as multiple models will have to be packaged as well as the required calculations. To make a proper decision we will have to measure more precisely the impact of each model to the application's resource usage and study the respective trade-offs.

In Section 8.1 we discussed some of the issues concerning the mobile deployment of the models and what steps we can take to alleviate them. We proposed using the MobileNetV2 as the base CNN for our models which is fast and efficient, quantizing the weights and the activations or pruning the network. Not all of these have been implemented yet on all models; the interests based model is trained with MonbileNetV2 with weights-only quantization post training, however we have only trained the DML model with EfficientNetB3, a server-side model and also as far as the text processing is concerned, we have not yet tested the applicability of the Tesseract OCR engine on the Android platform. It is important that these issues are addressed as the HELIOS platform matures and the module implementations start being tested. In the next section we will discuss an interesting research avenue of improving the models even after their deployment, based on the user interaction data.

Online Weight Adjustment

The development of a machine learning model does not stop after achieving satisfactory validation accuracy, but rather it also needs to be tested in real world conditions. There are many reasons that could lead to poor performance in practice and a major one is different input data distributions. This is particularly relevant to our task of developing content aware social graphs as HELIOS accepts input from many heterogeneous sources, that is different people from possibly different countries and with different images, and all of them cannot be appropriately represented in any reasonably sized dataset. Therefore, an important aspect of the models we create is whether they can adapt to better fit unforseen data.

However, development in the context of HELIOS is also challenging in another way as no data can leave the device and thus the adaptation process cannot be offloaded to an external server, but rather it has to be done on device with the usual computational limitations as well as the limited information one node has available on its own. The concept of federating learning[27] is very relevant to this topic, but in the remaining we will briefly discuss the possibility of leveraging implicit signals from user interactions to continuously train the models in each user's device.

The idea is to take advantage of interactions that would suggest that two conditional content aware user profiles should be close. The type of interaction we have in mind is simply when two users are in contact under the same spatio-temporal context. This information makes it quite probable that their content aware profiles conditioned under the same spatio-temporal context, in the sense described in Section 6, are also close. We can thus calculate on the devices of these two users a weight update in the direction that pushes the two conditional profiles closer.

There is a significant issue, however, with this approach; after some time each HELIOS user will have a different version of the model. This can be considered a desirable feature as we want the models to be personalized to better fit each user's needs, but at the same time it raises the risk that the users' content aware profiles could grow to unpredictable states. For example, in the case of two users with identical content stored on their devices, their past interactions could make their profiles seem distant while we would normally expect them to also be identical.

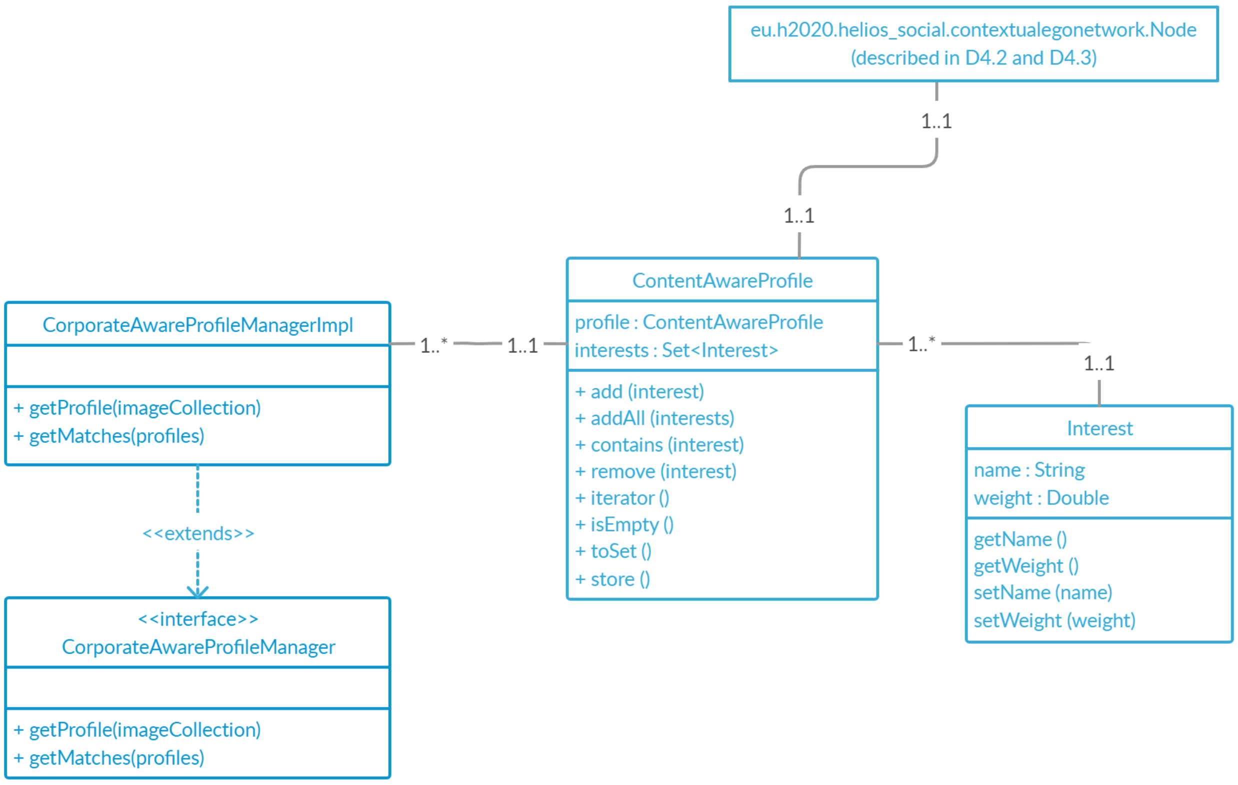
Therefore, although we believe that online weight adjustment as previously described is a promising idea that could enable our models to adapt to unforeseen circumstances, more research needs to be done to provide both qualitative and quantitative analysis on the evolution of the weights and the content aware profiles.

[27] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

Annex I - Content Aware Profiling Module

The Content Aware Profile Manager handles the creation of the users' interest profiles as well as the matching between users. With a user's available image collection as input, the manager can construct their interests profile using a deep learning algorithm. Furthermore, the manager can provide a metric on the quality of the match between two users based on their constructed interest profile. The Content Aware Profiling Module depends on core modules eu.h2020.helios_social.core.contextualegonetwork and eu.h2020.helios_social.core.storage.

Our main considerations when defining the module were: i) the usability of the module, developers should be able to use the module within 10 minutes after reading the documentation and ii) the extensibility of the module, to allow and make easy for other developers to build their own solutions and integrate their solutions into HELIOS platform. An overview of the Content Aware Profiling module is presented in the UML diagram below.



Bibliography

[1] B. Thomee *et al.*, “YFCC100M: The new data in multimedia research,” *Communications of the ACM*, vol. 59, no. 2. Association for Computing Machinery, pp. 64–73, Feb. 01, 2016, doi: 10.1145/2812802.

[2] A. V. Savchenko, K. V. Demochkin, and I. S. Grechikhin, “User Preference Prediction in Visual Data on Mobile Devices,” Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.04519.

[3] S. Wieczorek, D. Filipiak, and A. Filipowska, “Semantic Image-Based Profiling of Users’ Interests with Neural Networks.”

[4] Y. Xiong, K. Zhu, D. Lin, and X. Tang, “Recognize Complex Events from Static Images by Fusing Deep Channels.”

[5] L. Wang, Z. Wang, Y. Qiao, and L. Van Gool, “Transferring Deep Object and Scene Representations for Event Recognition in Still Images,” *Int. J. Comput. Vis.*, vol. 126, no. 2–4, pp. 390–409, Apr. 2018, doi: 10.1007/s11263-017-1043-5.

[6] K. Ahmad, N. Conci, and F. G. B. De Natale, “A saliency-based approach to event recognition,” *Signal Process. Image Commun.*, vol. 60, pp. 42–51, Feb. 2018, doi: 10.1016/j.image.2017.09.009.

[7] L. Laib, M. S. Allili, and S. Ait-Aoudia, “A probabilistic topic model for event-based image classification and multi-label annotation,” *Signal Process. Image Commun.*, vol. 76, pp. 283–294, Aug. 2019, doi: 10.1016/j.image.2019.05.012.

[8] D. M. Blei, A. Y. Ng, and J. B. Edu, “Latent Dirichlet Allocation Michael I. Jordan,” 2003.

[9] Q. You, S. Bhatia, and J. Luo, “A Picture Tells a Thousand Words -- About You! User Interest Profiling from User Generated Visual Content,” Apr. 2015, [Online]. Available: http://arxiv.org/abs/1504.04558.

[10] M. Cheung, J. She, and Z. Jie, “Connection Discovery using Big Data of User Shared Images in Social Media,” 2015.

[11] M. Cheung, J. She, and X. Li, “Non-user Generated Annotation on User Shared Images for Connection Discovery,” in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Dec. 2015, pp. 204–209, doi: 10.1109/DSDIS.2015.113.

[12] M. Cheung and J. She, “Detecting Social Signals in User-Shared Images for Connection Discovery Using Deep Learning,” *IEEE Trans. Multimed.*, vol. 22, no. 2, pp. 407–420, Feb. 2020, doi: 10.1109/TMM.2019.2930043.

[13] G. Piao and J. G. Breslin, “Exploring Dynamics and Semantics of User Interests for User Modeling on Twitter for Link Recommendations,” 2016. [Online]. Available: http://twitter.com.

[14] J. Kang, H. S. Choi, and H. Lee, “Deep recurrent convolutional networks for inferring user interests from social media,” *J. Intell. Inf. Syst.*, vol. 52, no. 1, pp. 191–209, Feb. 2019, doi: 10.1007/s10844-018-0534-3.

[15] W. Meira *et al.*, “Of pins and tweets: Investigating how users behave across image-and text-based social networks.” [Online]. Available: http://bit.ly/1ksdYHv.

[16] K. Q. Weinberger and L. K. Saul, “Distance Metric Learning for Large Margin Nearest Neighbor Classification,” 2009.

[17] R. Hadsell, S. Chopra, and Y. Lecun, “Dimensionality Reduction by Learning an Invariant Mapping.” [Online]. Available: http://www.cs.nyu.edu/~yann.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” Mar. 2015, doi: 10.1109/CVPR.2015.7298682.

[19] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, “Deep Metric Learning with Angular Loss,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.01682.

[20] W. Chen, X. Chen, J. Zhang, and K. Huang, “Beyond triplet loss: a deep quadruplet network for person re-identification,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.01719.

[21] K. Sohn, “Improved Deep Metric Learning with Multi-class N-pair Loss Objective.”

[22] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, “Deep Metric Learning via Lifted Structured Feature Embedding,” Nov. 2015, [Online]. Available:

http://arxiv.org/abs/1511.06452.
[23] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative Learning of Deep Convolutional Feature Point Descriptors.” [Online]. Available: https://github.com/etrulls/deepdesc-release.
[24] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.04861.
[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” Jan. 2018, [Online]. Available: http://arxiv.org/abs/1801.04381.
[26] R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper,” Jun. 2018, [Online]. Available: http://arxiv.org/abs/1806.08342.
[27] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, “What is the State of Neural Network Pruning?,” Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.03033.
[28] A. Ignatov *et al.*, “AI Benchmark: All About Deep Learning on Smartphones in 2019,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.06663.
[29] M. Mcpherson, L. Smith-Lovin, and J. M. Cook, “BIRDS OF A FEATHER: Homophily in Social Networks,” 2001.
[30] G. Berry, A. Sirianni, I. Weber, J. An, and M. Macy, “Going beyond accuracy: estimating homophily in social networks using predictions,” Jan. 2020, [Online]. Available: http://arxiv.org/abs/2001.11171.
[31] L. M. Aiello, A. Barrat, R. Schifanella, C. Cattuto, B. Markines, and F. Menczer, “Friendship prediction and homophily in social media,” in *ACM Transactions on the Web*, May 2012, vol. 6, no. 2, doi: 10.1145/2180861.2180866.
[32] M. Dehghani *et al.*, “Purity homophily in social networks,” *J. Exp. Psychol. Gen.*, vol. 145, no. 3, pp. 366–375, Mar. 2016, doi: 10.1037/xge0000139.
[33] G. A. Huber and N. A. Malhotra, “Political Homophily in Social Relationships: Evidence from Online Dating Behavior,” *J. Polit.*, vol. 79, pp. 269–283, 2017.
[34] K. Bischoff, *We love rock “n” roll: Analyzing and predicting friendship links in Last.fm*. 2012.
[35] M. De Choudhury, “Tie Formation on Twitter: Homophily and Structure of Egocentric Networks,” in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 465–470.
[36] A. De Salve, B. Guidi, L. Ricci, and P. Mori, “Discovering Homophily in Online Social Networks,” *Mob. Networks Appl.*, vol. 23, no. 6, pp. 1715–1726, Dec. 2018, doi: 10.1007/s11036-018-1067-2.
[37] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, “You Are Who You Know: Inferring User Profiles in Online Social Networks,” in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010, pp. 251–260, doi: 10.1145/1718487.1718519.
[38] A. De Salve, M. Dondio, B. Guidi, and L. Ricci, “The impact of user’s availability on On-line Ego Networks: A Facebook analysis,” *Comput. Commun.*, vol. 73, pp. 211–218, Jan. 2016, doi: 10.1016/j.comcom.2015.09.001.
[39] L. J. Li and L. Fei-Fei, “What, where and who? Classifying events by scene and object recognition,” 2007, doi: 10.1109/ICCV.2007.4408872.
[40] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” May 2019, [Online]. Available: http://arxiv.org/abs/1905.11946.
[41] R. M. Gray, *Entropy and Information Theory*, 2nd ed. Springer Publishing Company, Incorporated, 2011.
[42] F. Fouss, M. Saerens, and M. Shimbo, *Algorithms and Models for Network Data and Link Analysis*. Cambridge: Cambridge University Press, 2016.
[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” [Online]. Available: http://code.google.com/p/cuda-convnet/.
[44] Q. Ye and D. Doermann, “Text Detection and Recognition in Imagery: A Survey,” *IEEE*

Trans. Pattern Anal. Mach. Intell., vol. 37, no. 7, pp. 1480–1500, Jul. 2015, doi: 10.1109/TPAMI.2014.2366765.

[45] E. Cambria and B. White, “Jumping NLP curves: A review of natural language processing research,” *IEEE Computational Intelligence Magazine*, vol. 9, no. 2. Institute of Electrical and Electronics Engineers Inc., pp. 48–57, 2014, doi: 10.1109/MCI.2014.2307227.

[46] E. Krasanakis, S. Papadopoulos, and Y. Kompatsiaris, “LinkAUC: Unsupervised Evaluation of Multiple Network Node Ranks Using Link Prediction,” in *Studies in Computational Intelligence*, 2020, vol. 881 SCI, pp. 3–14, doi: 10.1007/978-3-030-36687-2_1.

[47] Emmanouil Krasanakis Symeon Papadopoulos and Y. Kompatsiaris, “Unsupervised Evaluation of Multiple Node Ranks by Reconstructing Local Structures.”

[48] S. Michel, J. X. Parreira, D. Donato, and G. Weikum, “Efficient and Decentralized PageRank Approximation in a Peer-to-Peer Web Search Network *,” 2006. [Online]. Available: https://www.researchgate.net/publication/221310188.

[49] Z. Bar-Yossef and L.-T. Mashiach, *Local Approximation of PageRank and Reverse PageRank*. 2008.

[50] A. Das Sarma, A. R. Molla, G. Pandurangan, and E. Upfal, “Fast Distributed PageRank Computation,” Aug. 2012, doi: 10.1016/j.tcs.2014.04.003.

[51] M. Kamgarpour and C. Tomlin, “Convergence properties of a decentralized Kalman filter,” in *2008 47th IEEE Conference on Decision and Control*, 2008, pp. 3205–3210.