Towards Activity Recommendation from Lifelogs

Gunjan Kumar[†], Houssem Jerbi[†], Cathal Gurrin[‡], and Michael P. O'Mahony[†]
Insight Centre for Data Analytics
[†]School of Computer Science and Informatics, University College Dublin, Ireland
[‡]School of Computing, Dublin City University, Ireland
{firstname.lastname}@insight-centre.org

ABSTRACT

With the increasing availability of passive, wearable sensor devices, digital lifelogs can now be captured for individuals. Lifelogs contain a digital trace of a person's life, and are characterised by large quantities of rich contextual data. In this paper, we propose a content-based recommender system to leverage such lifelogs to suggest activities to users. We model lifelogs as timelines of chronological sequences of activity objects, and describe a recommendation framework in which a two-level distance metric is proposed to measure the similarity between current and past timelines. An initial evaluation of our activity recommender performed using a real-world lifelog dataset demonstrates the utility of our approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Search and Retrieval; H.4.2 [**Information Systems Applications**]: Decision Support

General Terms

Algorithms, Experiments

Keywords

Lifelogging, Recommender Systems, Activity Recommendation, Activity Timeline Matching

1. INTRODUCTION

Nowadays, many moments of our lives are captured in digital format. Apart from routine online activities such as shopping, media consumption and social network activity, different aspects of our daily life can now be passively recorded using smartphones, smartglasses and a variety of wearable sensors. This shift to a digital form of life experience is supported by lifelogging. As defined by Dodge and Kitchin [7], lifelogging 'is conceived as a form of pervasive computing consisting of a unified, digital record of the totality of an individual's experiences, captured multi-modally through digital sensors and stored permanently as a personal multi-media archive.' In essence, lifelogging captures a detailed trace of one's life and its always-on and pervasive nature leads to the generation of vast quantities of rich user preference and contextual data. The availability of such data affords many new application scenarios and challenges from a research perspective.

To date, much of the research has focused on harnessing lifelogs to enable people visualise, monitor and review their personal experiences [8, 11, 12, 13, 14]. When the personal lifelogs of users are available, a new category of real-time recommender systems, which are capable of generating recommendations at the right time and in the right way for a given user and context, are facilitated. While recommender systems are frequently used by the likes of Amazon and Netflix to suggest products to consumers, here we consider a recommendation scenario in which the goal is to help people better plan their days, with potentially many interesting applications. For example, such a recommender system could be employed to automatically create daily activity plans, with alerts, to assist people with Alzheimer's disease (and their caregivers). Moreover, the system could be used to support users in their busy and fast-paced lives by suggesting *intentional* activities, which have been recognised to be a key influential factor of chronic happiness in the population [20].

In this paper, we propose a content-based recommendation approach as an initial step towards a personalised activity planning system. More specifically, our approach leverages a subset of a user's lifelogs to construct an activity timeline which is used to recommend the next activity to the user. Moreover, we argue that the sequence of activities and their contextual information (when they occurred, where, with whom, etc.) is specific to each user. Thus, we model the user activities record as a sequence of activity objects, where each object is characterised by a set of *features* that describe the context of the activity occurrence. It is worth noting that the activity features in a lifelog database depend on the devices and sensors used for the lifelog data collection.

To capture the personal habits and routines of users, our approach to recommendation identifies similar activity patterns between the user's current timeline and his past timelines to generate a recommendation. Activities to be recommended to the user are then derived from these similar timelines. While our approach supports more general recommendation scenarios (for example, the recommendation of sequences of activities, when and where and with whom they can be performed), in this initial treatment we consider the task of recommending the next activity to perform to the user.

The main contributions presented in this paper can be summarised as follows:

iiWAS '14, December 04 - 06 2014, Hanoi, Viet Nam

- A semantic view of lifelogs as a sequence of activity objects is formalised and the concept of an activity *timeline* is defined;
- A recommendation framework incorporating a new two-level similarity metric is proposed to assess the similarity between two sequences of activity objects; and
- Experiments based on a real-world lifelog dataset are performed which demonstrate that our activity recommendation approach achieves good overall performance.

The remainder of this paper is organised as follows. Related work is discussed in Section 2. Our activity recommendation framework is proposed in Section 3 and an evaluation of our approach is presented in Section 4. Section 5 provides conclusions and directions for future work.

2. RELATED WORK

Early developments in lifelogging research were motivated by the idea of external memories [5]. One of the pioneering projects in this field was MyLifeBits [4] which proposed the idea of comprehensively archiving all digital content related to a person. In order to passively capture different forms of lifelogs, wearable devices, such as SenseCam [16], smartglasses, and activity trackers are used. The 'always-on' and pervasive nature of lifelogs leads to a high volume of data which presents several research challenges, such as efficient data storage, retrieval, annotation, visualisation, etc. Gurrin et al. [13] proposed an architecture for maintaining, browsing and retrieving content from lifelog databases. Moreover, it has been recognised that the segmentation of lifelogs is essential for semantic extractions [9, 10]. For example, the software browser presented in [8] automatically segments sequences of collected images into events in order to facilitate their visualisation and exploration. These events are then manually annotated by the users.

Recently, new recommendation and prediction approaches have been inspired by the rich data generated by lifelogging. A classical example of such approaches is the prediction of user location and movement based on GPS data [2, 22]. Nevertheless, it has been shown that richer lifelog data may lead to a wider range of recommendation scenarios and more accurate recommendations. For example, it was demonstrated in [26] that using GPS data along with calendar log entries could improve movement prediction accuracy. A restaurant recommender system was proposed in [28], where GPS data, location history and the log of user actions on an Android device are used to infer the user's food preferences and location. TV program recommendation was considered in [23] based on the user web browsing history, the history of TV operations and GPS data. In [29], the logs of tunes listened to and the user context (weather, date and time) were employed to build a music recommendation system. These approaches tend to assist the lifelogger to make a choice related to a known activity; for example, which TV program to watch, which piece of music to listen to, where to eat out, etc. However, a lifelogger may be overwhelmed with activity choices, hence our aim is to recommend to her an activity to perform. To address this problem, our algorithm harnesses more generic lifelogging data which are not limited to user interactions with electronic devices (e.g. mobile phones, tablets, TV, etc.) and does not depend on any particular set of features.

Recent research has focused on activity recommendation to promote healthy and active lifestyles, typically based on the user's current context (i.e. location, time and weather). In [19], a constraint satisfaction approach is used to recommend physical activities based on user agenda, profile and current context. This approach, however, requires significant user effort, since users need to enter and update their agenda on a regular basis, which may impact the utility of the system. Another physical activity recommender system is proposed in [3] based on the calorie consumption and sedentary behaviour of users, as well as user context. Likewise, walking detours are recommended to users in [15] based on user context and profile. Moreover, the multidimensional recommender system model proposed in [1] was adapted in [32] in order to generate contextual activity recommendations. Unlike previous works, our algorithm considers the succession of user activities along with their contextual information to generate a recommendation. Further, our approach covers all everyday activities and can be applied in different scenarios.

Similarity assessment is a crucial step in any recommendation process. Similarity between two sets of elements are commonly calculated using classical measures, such as Jaccard distance, Hausdroff distance, cosine similarity, etc. However, these measures are insensitive to the order between the different elements. Different approaches have been proposed to assess the similarity between sequences [24], such as the edit distance between two strings which is the minimum cost of edit operations of single characters needed to transform one string into the other. The basic edit distance (or Levenshtein distance) [18] includes three edit operations (insertion, deletion, substitution) which are all assigned a unit cost. Other variations of the edit distance introduce additional edit operations [31] or assign different costs to the operations [25, 30]. While these distances apply for sequences of elements, other metrics have been proposed to assess the similarity between sequences of entities. For example, in token-based similarity metrics [6, 21], strings are viewed as sequences of substrings. In such approaches, the similarities between the pairs of entities are computed, and are then aggregated in order to derive the similarity between the sequences. However, the order of the entities in each sequence is ignored and does not contribute to the overall similarity between sequences. Smith and Waterman [27] introduced an extension of edit distance which offers a solution to this problem by allowing for better local alignment of the strings (i.e., substring matching). According to this metric, the similarity between two strings is the similarity score between the most similar substrings. Even though our work also explores similarity assessment between two sequences of entities (i.e. activities performed), we propose a hybrid similarity metric that considers both the sequence similarity and the similarities between the pairs of entities where a global alignment is applied (i.e., all entities of a sequence are matched to all entities of the other sequence).

3. ACTIVITY RECOMMENDATION

In this section, we formulate the problem of activity recommendation based on lifelog data and we present the supporting data model and the proposed recommendation algorithm.

3.1 Problem formulation

We address the problem of helping a lifelogger plan his day. Consider the scenario in which a lifelogger is performing an activity ao_c, and assume that the sequence of the user's past activities $<ao_1, ao_2, \ldots, ao_c>$ is available; the aim is to recommend to the lifelogger the next activity to perform. In this initial study, we focus on the recommendation of the activity name (referred to as *activity* for short).

The intuition behind our content-based recommendation approach is that people tend to carry-out similar patterns of activities during a particular time of the day (e.g., working in the morning and watching TV in the evening) or during weekdays versus weekends (e.g., commuting and working during weekdays, while shopping and walking during weekends). It is then important to detect similar patterns of activities in the user lifelog in order to efficiently infer the next activity that the user is likely to perform. While the list of activities may be similar for the same time period, their order may vary across different days. Different activities might also be performed due to specific circumstances. Hence, capturing the circumstances that impact the activities schedule for a given lifelogger is key to the activity prediction task. We assume that user activities are described by a set of features $f_1, \ldots, f_m$ and we capture such circumstances through the activity occurrence features. For example, when a lifelogger leaves work early (i.e., duration of activity *working* is short), she might *go shopping* rather than *go home*. Therefore, the extraction of similar past timelines should consider the different features of the activity objects. In consequence, a two-level similarity metric that considers both the order of activities and the similarities of activity object features is required to capture the similarity between two timelines.

3.2 Activity Timeline

In lifelogging, each activity performed is recorded along with the timestamp and other contextual information. Hence, a lifelog can be seen as a sequence of activities that are identified by different features. Each activity might occur several times in a given time range. We refer to each activity occurrence as an *activity object*.

The activity features range from information which describe the activity (e.g., name, duration, etc.) to information which are related to the activity occurrence context (e.g., starting time, location, temperature, luminosity, etc.).

When the set of activity objects that are performed within a time interval are arranged in a chronological order, the log of user activities represents a user *activity timeline* (or *timeline* for short). Formally, a user timeline $\mathcal{T}$ is a chronological sequence of n activity objects that are performed during a time interval δ:

$$\mathcal{T} =< ao_1, ao_2, ..., ao_n > , \quad (1)$$

where activity object ao_i represents the i^{th} activity object performed by the user during δ (δ is the duration between the starting time of ao_1 and the end time of ao_n) and is characterised by a set of m feature values:

$$ao_i = \left\{v_i^1, v_i^2, ..., v_i^m\right\} , \quad (2)$$

where v_i^j is the value of the feature f_j in activity object ao_i.

Example 1. Figure 1 shows an example timeline, $\mathcal{T}_1$, which includes three activity objects. Each object is characterised by four features: *Activity, Start time, Duration,* and *Location*. According to our model, timeline $\mathcal{T}_1$ is defined as follows: $\mathcal{T}_1 =< ao_1^1, ao_2^1, ao_3^1 >$, where, for example, $ao_1 = \{Working, 09{:}00, 65\,min, Science\,Center\}$.

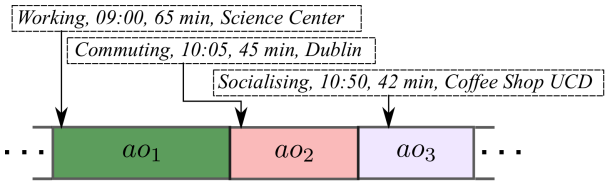

Figure 1: An example timeline.

3.3 Recommendation Generation

A key step of the activity recommendation process is to select past timelines that have similar patterns to the user's current timeline. Given the potentially very large quantity of recorded user timelines, selecting the best timelines has an important impact on the activity recommendation task.

Before proceeding further, we introduce some terminology as a guideline for what follows:

- *Current activity object* (ao_c): refers to the activity which occurs during the recommendation time; and
- *Current timeline* ($\mathcal{T}_c$): is the continuous sequence of activity objects spanning the time period from the start of the day to the recommendation time.

3.3.1 Distance between Timelines

In order to assess the similarity between two timelines $\mathcal{T}_1$ and $\mathcal{T}_2$ (i.e. two sequences of activity objects), we employ a two-level similarity algorithm, which first rearranges the activities to achieve the same activity sequence ($d_{activity}$), and then aligns the values of the features of the corresponding activity objects ($d_{feature}$).

In the first step, the edit distance between the two timelines $\mathcal{T}_1$ and $\mathcal{T}_2$ is computed as the minimum cost of edit operations of activity objects needed to transform the sequence of activities of $\mathcal{T}_1$ into the sequence of activities of $\mathcal{T}_2$ (only the activity names are considered). The following three edit operations are considered with specific costs:

- *Insert* an activity object into timeline $\mathcal{T}_1$ (cost c_{ins});
- *Delete* an activity object from timeline $\mathcal{T}_1$ (cost c_{del}); and
- *Substitute* an activity object with a different object in $\mathcal{T}_1$ (cost c_{sub}).

Each edit operation cost is weighted with the weight of an object, $w_{obj} = \sum_{k=1\ldots m} w_k$, where w_k is the weight of feature f_k. Thus, the edit distance between two timelines is given by:

$$\begin{aligned} d_{activity}(\mathcal{T}_1, \mathcal{T}_2) = &\sum_{i=1}^{r} w_{obj} \times c_{ins} \\ &+ \sum_{j=1}^{s} w_{obj} \times c_{del} \\ &+ \sum_{k=1}^{t} w_{obj} \times c_{sub} , \end{aligned} \quad (3)$$

where r, s, and t are respectively the numbers of insertion, deletion and substitution operations needed to transform the activity sequence of $\mathcal{T}_1$ into the activity sequence of $\mathcal{T}_2$.

In order to achieve a full alignment of $\mathcal{T}_1$ and $\mathcal{T}_2$, a second step of editing is performed to align the feature values of the corresponding activity objects in the two timelines. The distance between two activity objects ao_i^1 and ao_i^2 is the sum of the weights w_k of the individual features f_k that need to be updated to transform ao_i^1 into ao_i^2 as follows:

$$d_{feature}(ao_i^1, ao_i^2) = \sum_{i=1}^{l} w_i \ , \tag{4}$$

where ao_i^1 and ao_i^2 are the i^{th} activities in $\mathcal{T}_1$ (transformed as per Equation 3) and $\mathcal{T}_2$, and l is the number of features that need to be updated to transform ao_i^1 into ao_i^2.

Then, the overall distance between the timelines $\mathcal{T}_1$ and $\mathcal{T}_2$ is the sum of the two-level distances:

$$d(\mathcal{T}_1, \mathcal{T}_2) = d_{activity}(\mathcal{T}_1, \mathcal{T}_2) + \sum_{i=1}^{n} d_{feature}(ao_i^1, ao_i^2) \ . \tag{5}$$

Example 2. Consider timelines $\mathcal{T}_1$ and $\mathcal{T}_2$ below, where each activity object consists of four features (*Activity*, *Start time*, *Duration*, *Location*).

$\mathcal{T}_1 =< ao_1^1, ao_2^1, ao_3^1 >$, where:
$ao_1^1 = \{Working, 09{:}00, 65\ min, Science\ Center\}$,
$ao_2^1 = \{Commuting, 10{:}05, 45\ min, Dublin\}$,
$ao_3^1 = \{Socialising, 10{:}50, 42\ min, Coffee\ Shop\ UCD\}$.

$\mathcal{T}_2 =< ao_1^2, ao_2^2 >$, where:
$ao_1^2 = \{Working, 09{:}00, 95\ min, Science\ Center\}$,
$ao_2^2 = \{Socialising, 10{:}35, 50\ min, CSI\ Building\}$.

Assume that $c_{ins} = 1$, $c_{del} = 1$, c_{sub}=2, and the weights of the features *Activity, Start time, Duration* and *Location* are 3, 1, 0.5 and 0.5, respectively. Then, the weight of the activity object, w_{obj}, is equal to the sum of the weights of all features (i.e. 5).

The distance $d(\mathcal{T}_2,\mathcal{T}_1)$ between timelines $\mathcal{T}_2$ and $\mathcal{T}_1$ is computed as per Equation 5 as follows (see Figure 2):

- Step 1. In order to align the order of activities, ao_2^1 is inserted in $\mathcal{T}_2$, thus, $d_{activity}(\mathcal{T}_2, \mathcal{T}_1) = 5 \times 1 = 5$.
- Step 2. The features of the corresponding activity objects are aligned by updating the *Duration* of ao_1^2 and the *Start time*, *Duration* and *Location* of ao_2^2. Thus, $\sum_{i=1}^{2} d_{feature}(ao_i^1, ao_i^2) = (0.5) + (1 + 0.5 + 0.5) = 2.5$.
- Thus, $d(\mathcal{T}_2, \mathcal{T}_1) = 5 + 2.5 = 7.5$.

3.3.2 Algorithm

Essentially, our recommendation algorithm *ActivRec* attempts to find similar timelines to the current timeline from which it derives the activities for recommendation. The algorithm (see Algorithm 1) contains the following steps:

- Step 1. For a given recommendation time, the *current timeline*, $\mathcal{T}_c$, consisting of the sequences of activity objects which span the time period from the start of the day until the recommendation time, is extracted from the user lifelog. The *current activity object*, ao_c, is that which occurs during the recommendation time;

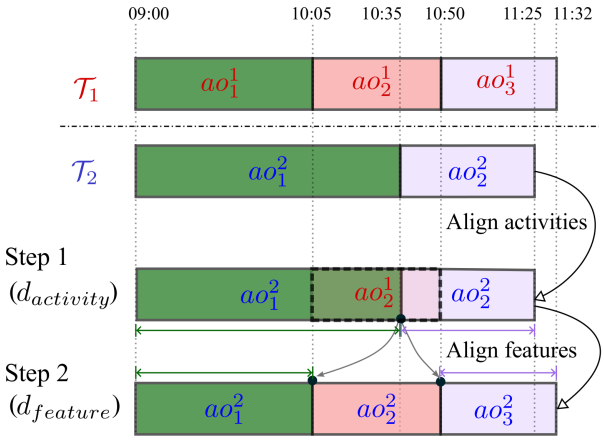

Figure 2: Edit distance between two timelines.

- Step 2. From each of the previous daily timelines in the user lifelog, a single *candidate timeline*, consisting of the sequence of activity objects from the start of the day and ending with an object having the same name as ao_c, is extracted. Daily timelines which do not contain an object with the same name as ao_c are excluded. When a daily timeline contains multiple activities with the same name as ao_c, the candidate timeline is that which has the closest distance to the current timeline;
- Step 3(a). After computing the edit distance between each candidate timeline and the current timeline, candidate timelines with distances exceeding a particular threshold are excluded;
- Step 3(b). A single activity object, ao_{rec}, is then recommended from each candidate timeline, i.e. that which occurs in the candidate timeline after the activity object with the same name as ao_c.
- Step 4. A relevance score for each recommended activity object, based on the distance between the candidate timeline in which it occurs and the current timeline, is then computed as per Equation 6:

$$Score(ao_{rec}^j) = 1 - \frac{d(\mathcal{T}_j, \mathcal{T}_c)}{\max\limits_{\mathcal{T}_p \in \mathscr{T}} d(\mathcal{T}_p, \mathcal{T}_c)} \ , \tag{6}$$

where $\mathcal{T}_c$ is the current timeline, $\mathcal{T}_j$ is a candidate timeline, ao_{rec}^j is the activity object recommended by $\mathcal{T}_j$, and $\mathscr{T}$ is the set of candidate timelines.

- Step 5. The top-N distinct activity names, ranked by the sum of the relevance scores of the activity objects in which they occur, are then returned to the user as recommendations.

Algorithm 1: ***ActivRec***

Input: User, u; current activity object, ao_c;
distance threshold, α; recommendation time, RT
Output: Top-N activity names

1. Extract the current timeline, $\mathcal{T}_c$, for user u at RT
2. Extract candidate timelines $\mathscr{T}$ (each $\mathcal{T}_j \in \mathscr{T}$ contains an activity ao_i^j, such that ao_i^j.name=ao_c.name)
3. **for** each $\mathcal{T}_j \in \mathscr{T}$ **do**
   Compute $d(\mathcal{T}_j, \mathcal{T}_c)$
   if $(d(\mathcal{T}_j, \mathcal{T}_c) > \alpha)$ **then**
   Delete $\mathcal{T}_j$ from $\mathscr{T}$
   else
   $ao_{rec} \leftarrow ao_{i+1}^j$
   $\mathcal{R} \leftarrow \mathcal{R} \cup ao_{rec}$
4. **for** each $ao_{rec} \in \mathcal{R}$ **do**
   Compute $Score(ao_{rec})$
5. **return** top-$N(ao_{rec}$.name $\mid ao_{rec} \in \mathcal{R})$

4. EVALUATION

The evaluation of the proposed activity recommender is performed on real-world lifelog data. In this section, we first describe the dataset used and the steps involved to convert the lifelog into timelines. Thereafter the experimental methodology is described, followed by a detailed evaluation of our proposed activity recommendation approach.

4.1 Dataset

Our dataset consists of lifelogging data in the form of timestamped images which were captured using Autographers[1], wearable cameras with hands-free automatic image capturing. This dataset contained 41,373 images captured for 5 users during a number of days over the period January–April 2014. Each image was (manually) classified based on a subset of activities proposed in [17]: *commuting, computer, eating, exercising, housework, on the phone, preparing food, shopping, socialising, watching TV* or *others*. Then daily timelines were created for each user, where each activity object corresponded to consecutive images labelled with the same activity. Activity objects were characterised by the following features: *user id*; *activity*; *date*; *start time*, given by the timestamp of the (first) image; and *duration*, given by the number of seconds until the next activity object. In total, our dataset contains timelines for between 10–14 weekdays and 0–4 weekend days for each user. We assume that the characteristics of timelines on weekdays and on weekend days are significantly different for users. As our dataset contains few weekend days we did not consider weekend timelines in our experiment.

Histograms of the number of distinct activities and the total number of activities per (weekday) timeline over all users are shown in Figures 3 and 4, while Table 1 shows the median and interquartile range (IQR) for these statistics per user. It can be seen that, on average, timelines contain between 3.0–5.5 distinct activities, while the total number of activities per day was much higher, between 21.5–44.5. Thus, we can conclude that reasonably rich timelines were generated for users using our dataset, with users performing

[1]http://www.autographer.com.

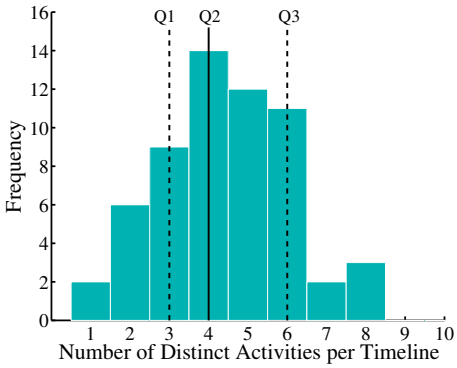

Figure 3: The number of distinct activities per timeline over all users.

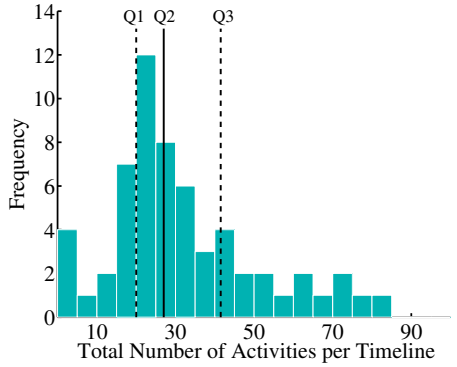

Figure 4: The total number of activities per timeline over all users.

Table 1: The number of distinct activities and the total number of activities per timeline per user.

User	Number of distinct activities		Total number of activities	
	Median (Q_2)	IQR	Median (Q_2)	IQR
1	3.0	1.0	27.0	22.0
2	3.0	2.0	23.0	19.8
3	5.5	3.0	34.0	31.0
4	5.0	1.0	21.5	7.0
5	5.5	2.0	44.5	28.0

a significant number of activities during each weekday for which lifelog data was available. Further, Figure 5 shows the number of times each activity appears over all weekday timelines. We observe that *computer* and *commuting* occur most frequently. Finally, Figure 6 shows the percentage of timelines in which an activity occurs at least once. This figure shows that certain activities are rare or common for some users; e.g. for user 2, *shopping* occurs in only a few timelines while for user 4, it occurs in most timelines.

4.2 Methodology

We conducted an offline evaluation of our approach where daily timelines were divided into training and test sets; test sets were comprised of the most recent 20% of each user's timelines. For each timeline in the test set, we generated top-N recommendations at different *recommendation times*

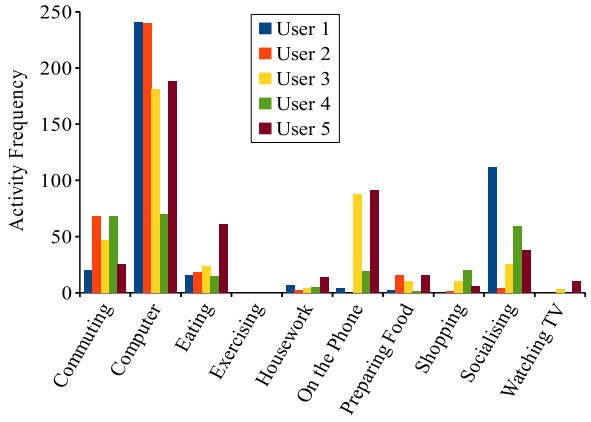

Figure 5: Number of times each activity occurs over all weekday timelines.

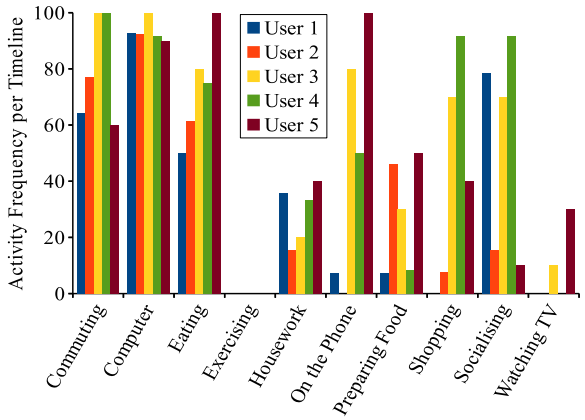

Figure 6: Percentage of weekday timelines in which each activity occurs.

(RT), which corresponded to the end time of each activity object in the timeline; in each case, the recommendation target was the next activity in the timeline. Recommendation accuracy was then evaluated using recall at top-N. The mean recall over each user's recommendation times were then computed. We also calculated, for each user, the recommendation coverage as the percentage of recommendation times for which at least one candidate timeline was available.

In our experiments, the following operation costs and feature weights were used when calculating edit distances between timelines: $c_{ins} = c_{del} = 1$, and $c_{sub} = 2$, i.e. the cost of the substitution operation was the sum of costs of the insertion and deletion operations; $w_{activity} = 3$, $w_{start\ time} = 1$, and $w_{duration} = 0.5$, i.e. the weight associated with updating an activity (name) was set to the highest value since the name of the activity is clearly the most important feature of an activity object, followed by activity start time and duration in descending order of feature importance.

4.3 Recommendation Performance

4.3.1 Recall vs. Distance Measure

Our recommendation algorithm (*ActivRec*) is based on a two-level edit distance that first aligns the sequence of activity objects then aligns the feature values of the related objects. In order to evaluate the performance of our distance metric, we use the following baselines:

- Sequence-based recommender (*LevenshteinRec*) which uses the same algorithm as *ActivRec*, but in this case the classical Levenshtein distance [18] is applied to the activity names to assess the distance between timelines, i.e. this baseline only considers activity sequences; and
- Most popular recommender (*PopRec*) is an algorithm in which one activity object is recommended from each of the daily timelines in the training set. For a given RT and daily timeline, $\mathcal{T}_j$, the activity object (ao^j_{rec}) with the closest start time to the RT is recommended with a score as per Equation 7:

$$Score(ao^j_{rec}) = 1 - min\left(1, \frac{|ST(ao^j_{rec}) - RT|}{60}\right), \quad (7)$$

where $ST(ao^j_{rec})$ is the start time of ao^j_{rec} in timeline $\mathcal{T}_j$ in minutes and RT is the recommendation time in minutes.

The top-N distinct activity names, ranked by the sum of the scores of the activity objects in which they occur, are returned as recommendations. Thus, the most frequently performed activities in previous timelines, at start times close to the RT, are recommended. As such, this baseline does not consider activity sequences, rather only the feature values of activity objects (in this case, a single feature, start time).

Figure 7 shows the average recall results of our algorithm and the two baselines across all users. It is clear that *ActivRec* considerably outperformed the *PopRec* baseline, where a percentage increase in recall of 107% and 43% were observed at top-1 and top-2, respectively. This finding clearly demonstrates the importance of the timeline concept and activity sequence matching for the activity recommendation task. Although the recall of our algorithm and *LevenshteinRec* was the same at top-1, our algorithm performed better thereafter, achieving an 8% and 4% percentage increase in recall at top-2 and top-5, respectively. While this increase in recall was not very high, we note the small number of activity features available in our dataset, which at least partially explains this finding. Overall, these results indicate that a two-level distance measure, which aligns activity sequences in addition to features, led to better quality recommendations.

4.3.2 Recall vs. top-N

The recall at top-N results achieved by the *ActivRec* algorithm are shown for each user in Figure 8. It is clear that the activity recommender provided good overall performance. For example, the recall at top-1 was in the range 0.30 (user 5) to 0.69 (user 1). This indicates that, over all recommendation times, the target category was ranked at the top of recommendation lists between 30% and 69% of the time. The minimum recall at top-2 recommendations increased to 0.51, while recall values between 0.68 (user 5) and 0.88 (user 2) were seen for top-5 recommendations.

In order to explain the trends observed above, consider user 2 and user 3. As can be seen from Table 1, user 3

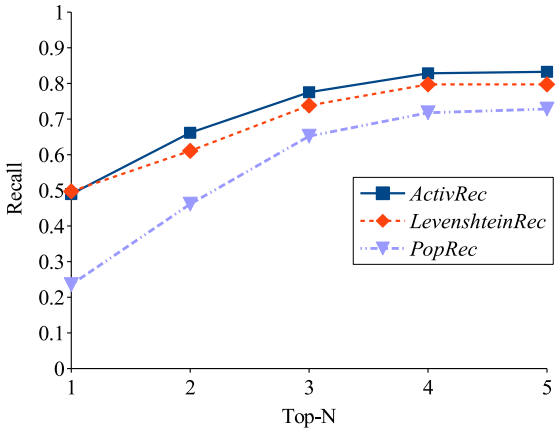

Figure 7: Recall at top-N for the *ActivRec* algorithm versus baseline approaches over all users.

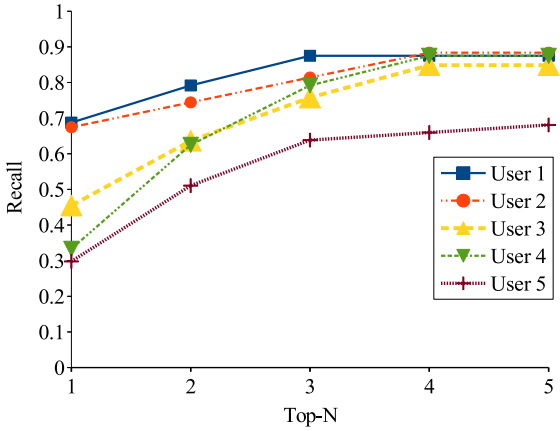

Figure 8: Recall at top-N for each user.

performed a greater number of activities compared to user 2. Moreover, it can be seen from Figures 5 and 6 that the activities performed by user 3 are more varied compared to user 2. For example, Figure 6 indicates that 6 activities occurred at least once in at least 60% of user 3's timelines; in contrast, there are only 3 activities which occurred in at least 60% of user 2's timelines. Hence, the superior recall achieved for user 2 was to be expected; however, it can be seen in Figure 8 that the recall for user 3 approached that for user 2 at larger recommendation list sizes.

Regarding recommendation coverage, values of over 97% were achieved for all users except for user 5, where a coverage of 55% was seen. The reason why low coverage was seen for user 5 is that one activity, *socialising*, occurred in this user's test set timelines only and in none of her training set timelines. Thus, for the 38 (out of a total of 85) recommendation times relating to this activity, no candidate timelines were available and hence no recommendation could be made.

4.3.3 Distance Thresholding

In the above experiments, all candidate timelines were considered at each recommendation time. However, some candidate timelines will be closer in terms of distance to current timelines, and as such may be a source of better quality recommendations. To test this hypothesis, the distributions of two-level edit distances between candidate and current timelines over all recommendation times are shown in Figure 9 for each user. In particular, it can be seen that the distance distributions for users 1, 2 and 3 are broadly similar, with relatively narrow interquartile ranges and median distance values between 60–69. While the distribution for user 4 is more uniform in nature, with a larger median value and interquartile range, the majority of distances for all these users are less than 200. However, the distribution for user 5 is significantly different, with peaks at distances of 45, 370 and 495, and with a much larger median and interquartile range.

There are a number of observations to make about these findings. Firstly, the recall achieved by the activity recommender for user 5 was relatively poor (see Figure 8), and this performance is at least partially explained by the differences observed between the distance distributions. Secondly, it seems likely that applying a distance threshold when selecting candidate timelines would be beneficial, particularly in the case of user 5. Thus, in the following experiment, a range of threshold values are considered, and candidate timelines which have a distance in excess of the threshold value to current timelines are excluded from the recommendation process. For each user, the percentage increase in recall at top-1 (compared to when no threshold was applied) versus threshold value is shown in Figure 10, and the corresponding recommendation coverage achieved at each threshold value is shown in Figure 11.

As expected, the application of distance thresholding had most effect for user 5 (i.e. the user with the highest variation in edit distance values). For example, a percentage increase in recall in excess of 100% was seen for this user for threshold values up to 150, thereafter the benefit was observed to decrease (Figure 10). However, this improved recall at low threshold values was achieved at the cost of a significant reduction in recommendation coverage; for example, coverage was seen to decrease from 0.55 when no threshold was applied to less than 0.25 for distance thresholds up to 200 (Figure 11). Regarding the remaining users in our study, optimal coverage was achieved for each of these users at a threshold value of 200 (as expected given the edit distance distributions). It is noteworthy, however, that the application of thresholds did not result in significantly improved recall for these users; rather, recall was seen to decline at very low threshold values for users 3 and 4. Thus, while there was no particular threshold value at which a reasonable tradeoff between both recall and coverage was achieved for the users in this study, nevertheless such an analysis is merited for our approach and may lead to enhanced performance in other datasets.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a content-based approach to activity recommendation based on user timelines that are modeled as sequences of activities. Our approach relies on timelines that have similar activity patterns to a user's current timeline in order to recommend the next activity. An adapted two-level edit distance is applied to measure the dissimilarity between timelines. Using a real-world lifelog dataset, we have demonstrated that good-quality recommendations can be made using our approach in the context of the activity recommendation task considered.

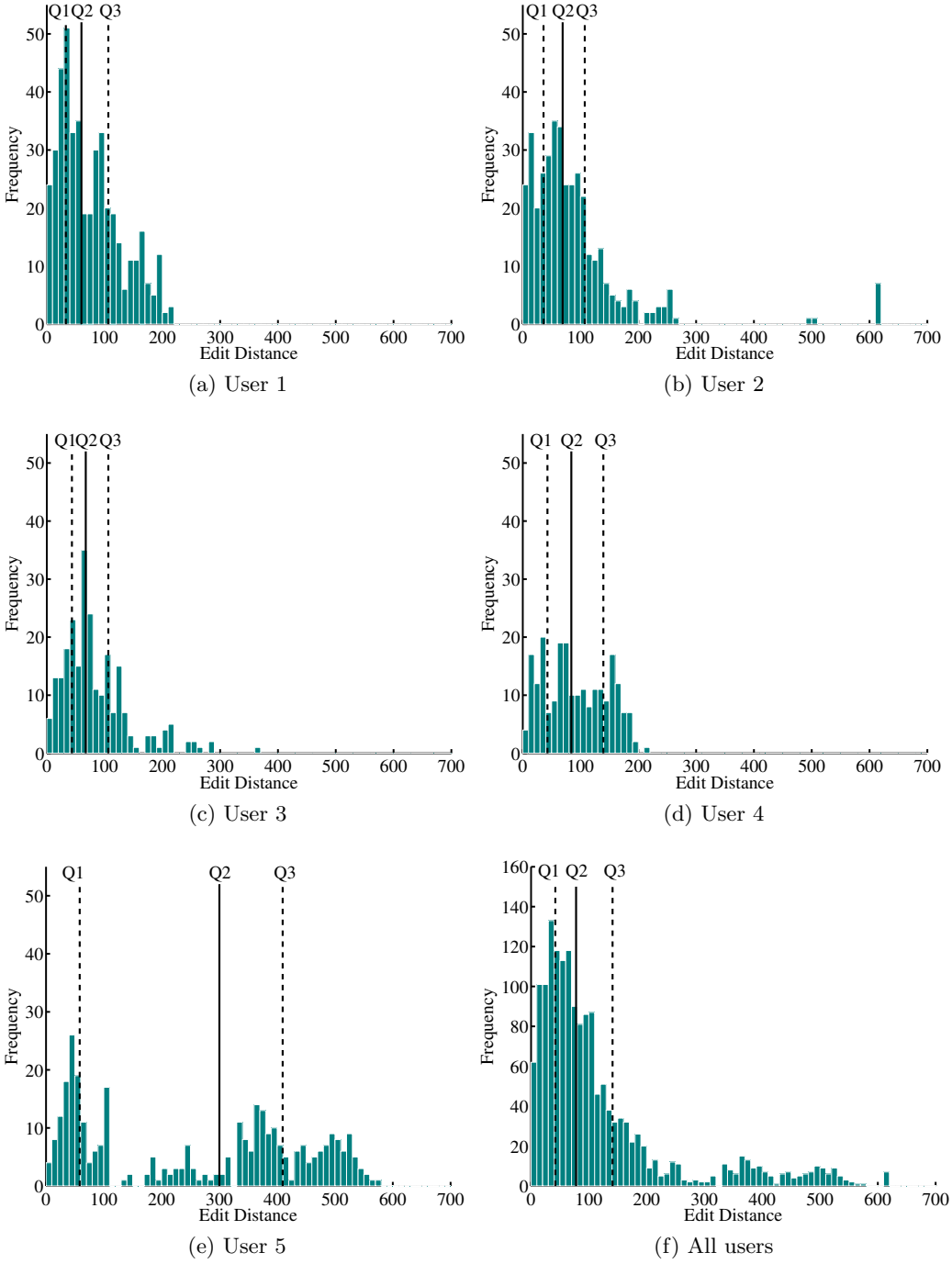

Figure 9: Histograms of two-level edit distances between candidate and current timelines.

We acknowledge some limitations in the work presented in this paper. Firstly, the study was performed using a small dataset, with a limited number of activity features. While promising performance was achieved, a larger dataset is needed to validate our findings; indeed, a larger dataset may lead to improved performance given the greater number of candidate timelines that would be available as a source of recommendations. Secondly, more sophisticated recommendation scenarios are possible; for example, recommending a sequence of activities to users, where these activities should take place, for how long and with whom they can be performed. We will consider the above in future work, in addition to a collaborative filtering approach which will facilitate the recommendation of novel and diverse activities to users.

6. ACKNOWLEDGMENTS

The Insight Centre for Data Analytics is supported by Sci-

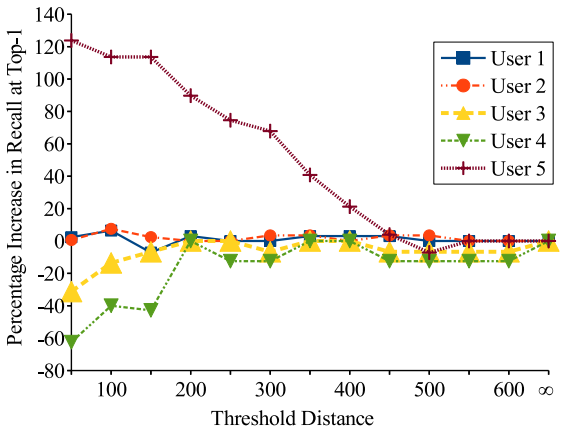

Figure 10: Percentage increase in recall at top–1 versus distance threshold.

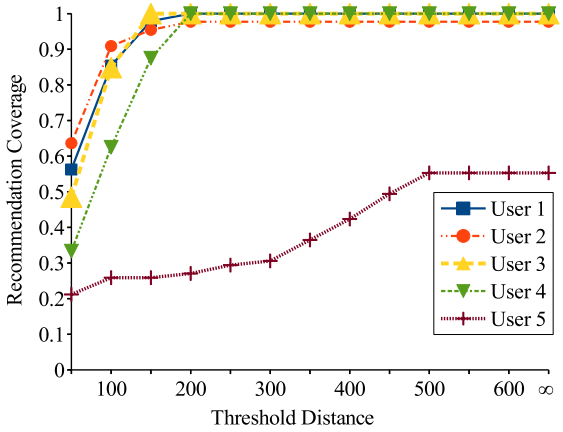

Figure 11: Recommendation coverage versus distance threshold.

ence Foundation Ireland under Grant Number SFI/12/RC/2289.

7. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, January 2005.

[2] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, October 2003.

[3] H. Badawi and A. E. Saddik. Towards a context-aware biofeedback activity recommendation mobile application for healthy lifestyle. *Procedia Computer Science*, 21(0):382–389, October 2013.

[4] C. G. Bell and J. Gemmell. *Total Recall: How the E-memory Revolution Will Change Everything.* Dutton, 2009.

[5] V. Bush. As we may think. *SIGPC Notes*, 1(4):36–44, April 1979.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string metrics for matching names and records. In *ACM KDD Workshop on Data Cleaning and Object Consolidation*, pages 73–78, August 2003.

[7] M. Dodge and R. Kitchin. Outlines of a world coming into existence: Pervasive computing and the ethics of forgetting. *Environment and Planning B: Planning and Design*, 34(3):431–445, March 2007.

[8] A. R. Doherty, C. J. A. Moulin, and A. F. Smeaton. Automatically assisting human memory: A SenseCam browser. *Memory*, 19(7):785–795, October 2011.

[9] A. R. Doherty and A. F. Smeaton. Automatically segmenting Lifelog data into events. In *Proceedings of the 9th International Workshop on Image Analysis for Multimedia Interactive Services*, pages 20–23. IEEE, May 2008.

[10] A. R. Doherty, A. F. Smeaton, K. Lee, and D. P. W. Ellis. Multimodal segmentation of lifelog data. In *Proceedings of RIAO '07, Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, pages 21–38. Le Centre De Hautes Etudes Iinternationales D'informatique Documentaire, May-June 2007.

[11] J. Gemmell, L. Williams, K. Wood, R. Lueder, and G. Bell. Passive capture and ensuing issues for a personal lifetime store. In *Proceedings of the 1st ACM Workshop on Continuous Archival and Retrieval of Personal Experiences*, pages 48–55. ACM, October 2004.

[12] A. Gomi and T. Itoh. A personal photograph browser for life log analysis based on location, time, and person. In *Proceedings of the 26th ACM Symposium on Applied Computing*, pages 1245–1251. ACM, March 2011.

[13] C. Gurrin, D. Byrne, N. O'Connor, G. J. F. Jones, and A. F. Smeaton. Architecture and challenges of maintaining a large-scale, context-aware human digital memory. In *Proceedings of the 5th International Conference on Visual Information Engineering*, pages 158–163. IET, July 2008.

[14] C. Gurrin, A. F. Smeaton, D. Byrne, N. O'Hare, G. J. F. Jones, and N. O'Connor. An examination of a large visual lifelog. In *Proceedings of the 4th Asia Information Retrieval Conference on Information Retrieval Technology*, pages 537–542. Springer-Verlag, January 2008.

[15] Q. He and E. Agu. On11: An activity recommendation application to mitigate sedentary lifestyle. In *Proceedings of the Workshop on Physical Analytics*, pages 3–8. ACM, June 2014.

[16] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood. Sensecam: A retrospective memory aid. In *Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 177–193. Springer-Verlag, September 2006.

[17] D. Kahneman, A. B. Krueger, D. A. Schkade, N. Schwarz, and A. A. Stone. A survey method for characterizing daily life experience: The day reconstruction method. *Science*, 306(5702):1776–1780, December 2004.

[18] V. Levenstein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of*

Information Transmission, 1(1):8–17, 1965.

[19] Y. Lin, J. Jessurun, B. de Vries, and H. Timmermans. Motivate: Context aware mobile application for activity recommendation. In *Proceedings of the 2nd International Conference on Ambient Intelligence*, pages 210–214. Springer-Verlag, November 2011.

[20] S. Lyubomirsky, K. M. Sheldon, and D. Schkade. Pursuing happiness: The architecture of sustainable change. *Review of General Psychology*, 9(2):111, June 2005.

[21] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 267–270. AAAI, August 1996.

[22] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: A location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 637–646. ACM, June-July 2009.

[23] Y. Nakamura, T. Itou, H. Tezuka, T. Ishihara, and M. Abe. Personalized tv-program recommendations based on life log. In *Digest of Technical Papers, International Conference on Consumer Electronics*, pages 143–144. IEEE, January 2010.

[24] G. Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, March 2001.

[25] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[26] M. Nishino, Y. Nakamura, T. Yagi, S. Muto, and M. Abe. A location predictor based on dependencies between multiple lifelog data. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, pages 11–17. ACM, November 2010.

[27] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[28] H. Tezuka, K. Ito, T. Murayama, S. Seko, M. Nishino, S. Muto, and M. Abe. Restaurant recommendation service using lifelogs. *NTT Technical Review*, 9(1), January 2011.

[29] A. Uno and T. Itoh. MALL: A life log based music recommendation system and portable music player. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 939–944. ACM, March 2014.

[30] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974.

[31] R. A. Wagner and R. Lowrance. An extension of the String-to-String Correction Problem. *Journal of the ACM*, 22(2):177–183, April 1975.

[32] C.-Y. Wang, Y.-H. Wu, and S.-C. T. Chou. Toward a ubiquitous personalized daily-life activity recommendation service with contextual information: A services science perspective. *Information Systems and E-Business Management*, 8(1):13–32, January 2010.