LEMoRe: A Lifelog Engine for Moments Retrieval at the NTCIR-Lifelog LSAT Task

Gabriel de Oliveira Barra
University of Barcelona
gabriel.deoliveira@ub.edu

Alejandro Cartas Ayala
University of Barcelona
Funded by CONACyT, Mexico
alejandro.cartas@ub.edu

Marc Bolaños
University of Barcelona
marc.bolanos@ub.edu

Mariella Dimiccoli
Computer Vision Center
mariella.dimiccoli@cvc.uab.es

Xavier Giro-i-Nieto
Polytechnic University of Catalonia
xavier.giro@upc.edu

Petia Radeva
University of Barcelona
petia.ivanova@ub.edu

ABSTRACT
Semantic image retrieval from large amounts of egocentric visual data requires to leverage powerful techniques for filling in the semantic gap. This paper introduces LEMoRe, a Lifelog Engine for Moments Retrieval, developed in the context of the Lifelog Semantic Access Task (LSAT) of the NTCIR-12 challenge and discusses its performance variation on different trials. LEMoRe integrates classical image descriptors with high-level semantic concepts extracted by Convolutional Neural Networks (CNN), powered by a graphic user interface that uses natural language processing. Although this is just a first attempt towards interactive image retrieval from large egocentric datasets and there is a large room for improvement of the system components and the user interface, the structure of the system itself and the way the single components cooperate are very promising.

Team Name
LEMoRe Team from the University of Barcelona and Technical University of Catalonia.

Subtasks
Lifelog Semantic Access Task (LSAT)

Keywords
lifelogging, egocentric images, semantic image retrieval

1. INTRODUCTION
With the advances of wearable technologies during the last years, lifelogging has become a common trend nowadays. However, since lifelogging implies the collection of a huge amount of data, it requires powerful data management techniques to extract and retrieve the information of interest. The goal of the NTCIR Lifelog Semantic Access Task (LSAT) is to design new techniques for retrieving specific moments from the lifelog of a person. The challenge dataset consists of 90,586 images captured by three different users during a period of almost a month. Table 1 summarizes the characteristics of the dataset as well as of the users who contributed to gather the data. A subset of 89,593 images was enriched with tags of location and activity, and a 1,000 classes object classifier was used to assign object categories to the pictures.

In this paper, we describe LEMoRe, a new Lifelog Engine for Moments Retrieval, developed to participate in the NTCIR LSAT task. LEMoRe is based on the idea that lifelogging image annotations could provide the basic semantic context to retrieve a specific event. In other words, a particular moment of a person’s life can be retrieved based on the objects present in that moment (food, laptop, beer, etc.), its location (bar, office, home, etc), the activity performed (shopping, working, etc.), and time along the day (morning, afternoon, etc), in an interactive fashion.

A first introduction to LEMoRe is given in sec. 2 and its architecture is detailed in sec. 3. In sec. 4, the required data preprocessing tasks are explained. A detailed list of functionalities offered by the engine is presented in sec. 5. A description of the performed run trials and their results is presented in sec. 6 and 7, respectively. Finally, sec. 8 provides some concluding remarks.

2. LEMORE
LEMoRe is an interactive Lifelog Engine for Moments Retrieval, whose front-end is a web-based user interface. Since LEMoRe relies on the semantic context (objects, places, ac-
activities and time) to retrieve a specific moment, the most important element considered in the interaction with the user is a heatmap of the semantic context in terms of tags scores by day and hour/minute, as shown in Fig. 1. Its purpose is to sort the images by the time, they were taken and highlight the ones that contain a certain input set of tags that express the context of a moment.

For example, if user u1 is looking for a moment, when he was riding a red colored train, then the search could start by using the tags train and train station. The corresponding heatmap plot for this combination of tags (see Fig. 1) identifies three different dates, where the object train and the location train station might coincide in the images. After inspecting the images from the most probable hours of the suggested dates, the user could find the specific moment, he is looking for. In this case, some representative images that the actual search could retrieve are shown in Fig. 2, and for this example a good answer might be the image in Fig. 2b.

Another key idea considered is that events may have similar settings at different times and dates. For instance, a moment like watching the TV in the living room is restricted to a place and can happen at the same time in different days or weeks. For the purpose of finding similar events across different days, LEMoRe can retrieve images with similar low-level features features, as it is further explained in sec. 3.

3. OVERALL ARCHITECTURE

As shown in Fig. 4, our proposed architecture for egocentric image retrieval consists of two main components: an image features engine, and a semantic engine. The image features engine retrieves images based on low-level features such as their color and textures characteristics, i.e. the task of this engine is to search for similar images. Specifically, our system uses auto color correlogram (CL), edge histogram (EH), joint composite descriptor (JCD), and pyramid histogram of oriented gradients (PHOG). The retrieval system is the open source Lucene Image Retrieval engine (LIRE) [1].

The semantic engine retrieves images based on their tags (high-level concepts) and temporal information. The main idea of the semantic engine is that the tagged object detection, times, locations, and actions offer good semantic descriptors for events retrieval. The engine combines matrix numeric processing and database queries for object and temporal tags, respectively. Our engine offers four different operations as web components, that are fully described in sec. 5. These components were implemented in Python (NumPy) and (SQLite).

4. DATA PREPROCESSING

In order to achieve better results and faster response-times from our semantic engine, four data preprocessing tasks were carried out: 1) merging the semantic tags, 2) computing the score histogram for each tag per minute/hour, 3) creating a tag similarity matrix, and 4) filling a database with this data. In particular, merging the semantic tags and the creation of a similarity matrix had the purpose of aggregating data, while the rest served for saving computing time during browsing the images. The following subsections describe the four data preprocessing tasks.

4.1 Tags Confidence

The tags confidence matrix contains the score values of all tags for each image in the dataset, where the rows and columns correspond to images and tags, respectively. The objective of this task is to merge in a single matrix the information about the location, activity and tags provided by the dataset and additionally the tags obtained from the LSDA object detector [2]. All the tags have a score value between 0 and 1 that represents the level of confidence of the annotation. Since the activity and location were manually annotated for some of the images, their scores were set to 1. In order to simplify the number of places, the locations provided were manually clustered. For instance, the words Tesco and Lidl were clustered together as Supermarket.

Since the tags scores provided by the original dataset were obtained using a CNN classifier trained on Caffe [3], only a small set of the images that present big centered objects might have been correctly classified. For instance, Fig. 3 shows the difference between a well-tagged image and badly-
Table 2: The final number of tags for the merged sources.

<table>
<thead>
<tr>
<th>Class ID</th>
<th>First class name</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>673</td>
<td>mouse</td>
<td>0.17</td>
</tr>
<tr>
<td>527</td>
<td>desktop computer</td>
<td>0.13</td>
</tr>
<tr>
<td>508</td>
<td>computer keyboard</td>
<td>0.13</td>
</tr>
<tr>
<td>564</td>
<td>monitor</td>
<td>0.12</td>
</tr>
<tr>
<td>826</td>
<td>desk</td>
<td>0.12</td>
</tr>
<tr>
<td>782</td>
<td>screen</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**Figure 3:** Examples of the top five tags for a couple of images using the provided Caffe concepts by the NTCIR.

tagged image. Therefore, the LSDA object detector [2] was employed to enriching the number of concepts in the dataset. From its output, the three most probable detected objects were added as tags for each image with a value of 1.

The problem of merging tags from different sources is their repetition. So if a couple of tags had the same name, then its score for each image was calculated as max(s_1, s_2), where s_1 and s_2 are the original score values. Table 2 shows the final number of tags for the merged sources.

### 4.2 Tags Histogram

In this task, a score histogram h_t by day and minute/hour is created for each tag t. Moreover, each histogram is normalized by dividing it with its maximum value.

### 4.3 Tags Similarity

The tags similarity matrix represents a graph structure that links the level of similarity between tags. The similarity matrix is a symmetric matrix, whose values are between 0 and 1, where 0 and 1 are the lowest and highest level of similarity, respectively. The similarity between tags was obtained using WordNet [4]. The similarity value between two tags was calculated as the maximum similarity given by WordNet between the synonyms of both tags.

### 4.4 SQL database

In order to perform time-related queries, a SQL database was created. Its schema is focused on image ordered by its time, date, and user; Fig. 5 depicts the designed schema. In addition, the tags matrix row corresponding to each image was added as an attribute. This attribute is the link to perform operations with the image timestamps and the rest of the annotated data.

### 5. FUNCTIONALITIES

The functionalities were designed to minimize the response time. Basically, their development followed two strategies: (1) using fast and efficient algorithms and (2) splitting the matrices of data in rows for fast memory access.

#### 5.1 Scores histogram

The scores histogram functionality computes the sum of the tag scores histogram \( H \) by hour or minute given a set of \( n \) input tags. Essentially, it is an OR query operation that calculates:

\[
H = \sum_{i=1}^{n} h_{t_i}
\]

for a set of tags \( \{t_1, t_2, \ldots, t_n\} \). This functionality sums the histograms already computed as stated in sec. 4.2.

#### 5.2 Tag similarity

The tag similarity functionality suggests \( m \) semantically related tags given a set of \( n \) input tags. This service computes the set \( T \) of the \( m \) most similar tag indexes:

\[
T = \arg \max_{i, j \in \{1, \ldots, m\}} I_{ij} - \sum_{i} s_{t_i}.
\]
Recall

Interpolated Precision

Run 1 Sec 10
Run 1 Sec 120
Run 1 Sec 300
Run 2 Sec 10
Run 2 Sec 120
Run 2 Sec 300
Run 3 Sec 10
Run 3 Sec 120
Run 3 Sec 300

Figure 6: Event-level interpolated precision over recall for all submitted runs on seconds 10, 120, and 300.

Mean NDCG

Run 1
Run 2
Run 3

Figure 7: Mean normalized discounted cumulative gain (Mean NDCG) results over time for each run of event-level retrieval.

where \( t_i \) is an index of the similarity matrix and \( s_{t_i} \) one of its rows. In order to make this functionality faster, first it only loads in memory those requested rows from the similarity matrix. After the sum of the \( n \) rows is done, it returns the first \( m \) indexes using the Wirth partial sorting algorithm [5].

5.3 Images

The image functionality queries the database and returns a list of images given a user, date, and hour or minute. Additionally, it can sort the images by their sum class score given a set of \( n \) input tags. This is done by computing the class sum score:

\[
T = \sum_{i=1}^{n} s_{t_i}, \quad (3)
\]

where \( t_i \) is the tag index of each image, and then sorting them by it.

5.4 Top scored images

The top scored images functionality returns a list of \( m \) images with the highest class score given a set of \( n \) input tags. First, the functionality computes the minutes histogram sum of the requested \( n \) tags (sec. 5.1). Then, it obtains the largest \( m/2 \) minutes by partially sorting the resulting matrix \( H \) using [5]. Since it is expected that every minute has at least two images, then the sum of the tag scores is done for each image in every minute, thus resulting on \( m \) images. Finally, the sorted list of images by its scores is returned.

6. RUN TRIALS

In the Lifelog Semantic Access Task (LSAT) subtask, the participants have to retrieve a number of specific moments in a lifelogger’s life. We define moments as semantic events or activities that happened throughout the day. We submitted three different runs to the challenge competition. Each run was made by 4 different users, doing 12 out of the 48 queries each. Before querying the system, the users were instructed to follow the procedure described in Algorithm 1. The first run was made by users who were familiar to the competition. For the second run, a tag suggestion service was added in order to observe possible performance improvement of the system. This test was done by the same users as the test one, but performing a different set of queries each. The objective of the third run was to simulate a real-case scenario and test the usability of the system. Therefore, this run was made by users who were not familiar with the competition and also using the tag suggestion functionality.

7. RESULTS

On the tables 3 and 4, we present a quantitative overview for image- and event-level retrieval results for each run at
8. CONCLUSIONS & FUTURE WORK

After a testing process of the LEMoRe in three submission runs, we obtained deeper insight about the system and its performance. Our results showed that its performance was better on moments that take place over a continuous time lapse rather than sparse. Moreover, we consider that better semantics improves event retrieval results, as the difference between the first and the second runs scores showed.

We will further work on improving the semantics of the system. Specifically, we explore techniques that produce tags for common daily settings that describe better the context. Additionally, we will also provide better interactive feedback to the user based on his/her selected images.

Since our final goal is to create a general purpose tool that is not limited to a specific kind of user, our efforts will be focused on two main objectives. First, the results showed a gap between experienced and non-experienced users that will be solved by making a friendlier interface. Second, the system still requires to further improve scalability and privacy issues.

9. ADDITIONAL AUTHORS

Additional authors: Maedeh Aghaei (University of Barcelona. email: maghaieigavar@ub.edu) and Marc Carné (Polytechnic University of Catalonia. email: marc.carne.herrera@estudiant.upc.edu).

10. REFERENCES

<table>
<thead>
<tr>
<th>Runs</th>
<th>Seconds</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td>Retrieved</td>
<td>16</td>
<td>159</td>
<td>317</td>
<td>528</td>
<td>1016</td>
<td>12</td>
<td>85</td>
<td>269</td>
<td>608</td>
<td>1153</td>
<td>11</td>
<td>73</td>
<td>166</td>
<td>349</td>
<td>752</td>
</tr>
<tr>
<td></td>
<td>Relevant</td>
<td>443</td>
<td>3978</td>
<td>5359</td>
<td>5755</td>
<td>6414</td>
<td>1417</td>
<td>2690</td>
<td>3076</td>
<td>5878</td>
<td>6606</td>
<td>406</td>
<td>1173</td>
<td>2793</td>
<td>4708</td>
<td>6306</td>
</tr>
<tr>
<td></td>
<td>Relevant-Retrieved</td>
<td>13</td>
<td>123</td>
<td>273</td>
<td>475</td>
<td>839</td>
<td>8</td>
<td>67</td>
<td>170</td>
<td>422</td>
<td>844</td>
<td>9</td>
<td>65</td>
<td>152</td>
<td>283</td>
<td>568</td>
</tr>
<tr>
<td>Mean Average Precision (MAP)</td>
<td>0.0048</td>
<td>0.0604</td>
<td>0.0965</td>
<td>0.1299</td>
<td>0.1834</td>
<td>0.0019</td>
<td>0.0211</td>
<td>0.0573</td>
<td>0.1372</td>
<td>0.2224</td>
<td>0.0119</td>
<td>0.0213</td>
<td>0.05</td>
<td>0.1075</td>
<td>0.1863</td>
<td></td>
</tr>
<tr>
<td>Average Precision, Geometric Mean</td>
<td>0.0</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0009</td>
<td>0.0042</td>
<td>0.0</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0019</td>
<td>0.0095</td>
<td>0.0</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0007</td>
<td>0.0041</td>
<td></td>
</tr>
<tr>
<td>R-Precision</td>
<td>0.0048</td>
<td>0.0604</td>
<td>0.0973</td>
<td>0.1356</td>
<td>0.1939</td>
<td>0.0019</td>
<td>0.0213</td>
<td>0.0579</td>
<td>0.1386</td>
<td>0.2292</td>
<td>0.0149</td>
<td>0.0383</td>
<td>0.0476</td>
<td>0.1067</td>
<td>0.1882</td>
<td></td>
</tr>
<tr>
<td>Binary Preference (bpref)</td>
<td>0.0048</td>
<td>0.0604</td>
<td>0.0973</td>
<td>0.1356</td>
<td>0.1925</td>
<td>0.0019</td>
<td>0.0213</td>
<td>0.0579</td>
<td>0.1384</td>
<td>0.2277</td>
<td>0.0112</td>
<td>0.0183</td>
<td>0.0476</td>
<td>0.1059</td>
<td>0.1864</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Image-level summary statistics of all runs and seconds for the total number of images retrieved over all image queries.

<table>
<thead>
<tr>
<th>Runs</th>
<th>Seconds</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
<th>10</th>
<th>30</th>
<th>60</th>
<th>120</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run 1</td>
<td>Retrieved</td>
<td>6</td>
<td>18</td>
<td>29</td>
<td>46</td>
<td>104</td>
<td>11</td>
<td>30</td>
<td>49</td>
<td>79</td>
<td>140</td>
<td>5</td>
<td>13</td>
<td>28</td>
<td>67</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>Relevant</td>
<td>23</td>
<td>105</td>
<td>215</td>
<td>259</td>
<td>295</td>
<td>85</td>
<td>136</td>
<td>197</td>
<td>272</td>
<td>310</td>
<td>10</td>
<td>35</td>
<td>124</td>
<td>213</td>
<td>305</td>
</tr>
<tr>
<td></td>
<td>Relevant-Retrieved</td>
<td>4</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>73</td>
<td>8</td>
<td>20</td>
<td>32</td>
<td>56</td>
<td>94</td>
<td>4</td>
<td>12</td>
<td>22</td>
<td>48</td>
<td>82</td>
</tr>
<tr>
<td>Mean Average Precision (MAP)</td>
<td>0.0484</td>
<td>0.1215</td>
<td>0.1567</td>
<td>0.1895</td>
<td>0.3175</td>
<td>0.0284</td>
<td>0.0955</td>
<td>0.1626</td>
<td>0.3145</td>
<td>0.2164</td>
<td>0.0112</td>
<td>0.1054</td>
<td>0.1767</td>
<td>0.2562</td>
<td>0.3857</td>
<td></td>
</tr>
<tr>
<td>Average Precision, Geometric Mean</td>
<td>0.0</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0013</td>
<td>0.0078</td>
<td>0.0</td>
<td>0.0001</td>
<td>0.0003</td>
<td>0.0036</td>
<td>0.0184</td>
<td>0.0</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0016</td>
<td>0.0162</td>
<td></td>
</tr>
<tr>
<td>R-Precision</td>
<td>0.0484</td>
<td>0.1215</td>
<td>0.1466</td>
<td>0.1832</td>
<td>0.3159</td>
<td>0.0281</td>
<td>0.0002</td>
<td>0.1667</td>
<td>0.3052</td>
<td>0.4213</td>
<td>0.0208</td>
<td>0.095</td>
<td>0.1685</td>
<td>0.2586</td>
<td>0.3949</td>
<td></td>
</tr>
<tr>
<td>Binary Preference (bpref)</td>
<td>0.0484</td>
<td>0.1215</td>
<td>0.1466</td>
<td>0.1831</td>
<td>0.3365</td>
<td>0.0281</td>
<td>0.0001</td>
<td>0.1658</td>
<td>0.3041</td>
<td>0.4164</td>
<td>0.0208</td>
<td>0.095</td>
<td>0.1682</td>
<td>0.2496</td>
<td>0.3832</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Event-level summary statistics of all runs and seconds for the total number of images retrieved over all event queries.

---