Extraction and analysis of territorial emotions: GeoMotion, an approach to identify predominant emotions in an area through audio recordings

Davide Bruni, Daniel Deiana, Marco Galante, Lorenzo Guidotti
d.bruni6@studenti.unipi.it, d.deiana1@studenti.unipi.it, m.galante1@studenti.unipi.it, l.guidotti6@studenti.unipi.it

ABSTRACT

Emotions are integral to human interactions and experiences, often tied to specific locations. For example, amusement parks evoke joy, while hospitals may induce anxiety. However, the study of emotions linked to places through audio recordings is relatively new. To address this gap, we present GeoMotion, an innovative Android application that captures and associates emotions with geographic locations using audio recordings. GeoMotion employs a Neural Network to extract emotions directly from audio, bypassing the traditional Speech-to-Text process. This allows users to explore the emotional landscape of different areas through sound. GeoMotion has potential applications in various fields. In social media, it enables users to share emotional experiences of places. For sociological and psychological research, it can provide insights into behavioral patterns and environmental influences. Tourists can use it to identify vibrant or safer areas in a city. The app's architecture integrates Firebase for user authentication, Google Maps API for location services, and Google Cloud for audio storage. A TFLite model processes audio data, achieving 73% accuracy in emotion recognition. Power consumption tests reveal efficient energy usage, crucial for mobile application sustainability. Future developments could include model customization, multimodal classifiers, and enhanced privacy features, further expanding GeoMotion's utility.

1 Introduction

Emotions play a significant role in various aspects of our lives, influencing how we interact with others and our surroundings. Specific emotions often become linked to locations. For instance, hospitals commonly evoke feelings of anxiety, fear, or pain, while amusement parks tend to elicit happiness and joy. Furthermore, emotions aren't exclusively determined by location: they're also influenced by the immediate circumstances individuals find themselves in. Take, for example, people stuck in traffic: if they're running late for work, they might experience anger and frustration, but their emotions can shift dramatically to joy if they're still in traffic but celebrating their favorite team's victory. However, the association of emotions with specific places and audio recordings is a relatively unexplored territory. This presents a gap in how we can understand and interact with our surrounding environment. To address this challenge, we introduce GeoMotion an Android application that allows users to record audio and associate it with the current location of the users on the map. What distinguishes GeoMotion is its capacity to not only capture sounds but also discern emotions tied to places and audio recordings, achieved through a Neural Network extracting emotions directly from audio rather than employing a Speech-to-Text process followed by emotion recognition from text. This cutting-edge approach enables users to explore not only physical locations but also the emotional experience tied to those places and the audio recordings present.

GeoMotion has potential usages in different domains. We describe a few usages, to illustrate this.

Social media purposes: People can exchange information and sensations about the places they visit.

Social and humanistic studies: It could help explain various sociological and psychological behaviors of populations, such as voting patterns in democratic countries or describe how the environment influences people.

Touristic search: People arriving in a location might want to know which areas of the city are most exciting to visit, as well as which ones are less safe and should be avoided.

1.1 Related works

Similar approaches are presented by Y. Doytshe et al. [1], who utilize thematic maps generated from a dataset of social media posts. In their study, emotions are extracted through emotion analysis applied to the posts rather than using audio recordings. As their aim wasn't to develop a standalone application but rather to create thematic maps, the data used to construct these maps was collected in real-time from Twitter (the current X).

Other tests [2], on the other hand, have tried to collect different data gathered from the keyboard, gyroscope, and accelerometer to compose a dataset for emotion recognition, but without associating these predicted emotions with a map, which is the cornerstone of our application.

2 Architecture

As shown in *Figure 1*, the mobile application communicates with: *Firebase, Google Maps API, Firestore* and *Google*

Cloude Service.

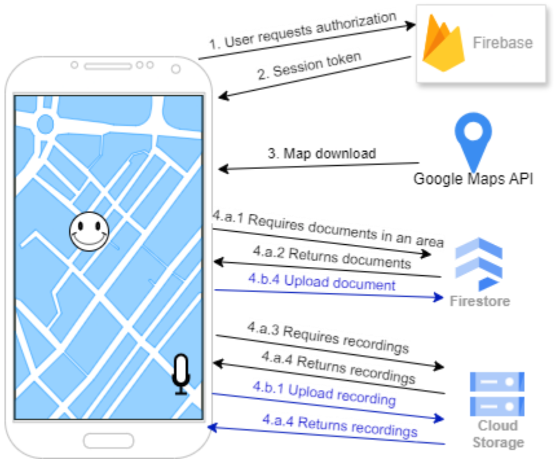


Figure 1: Architecture scheme and communication paradigm

Once the application is opened, it will prompt the user to authenticate.

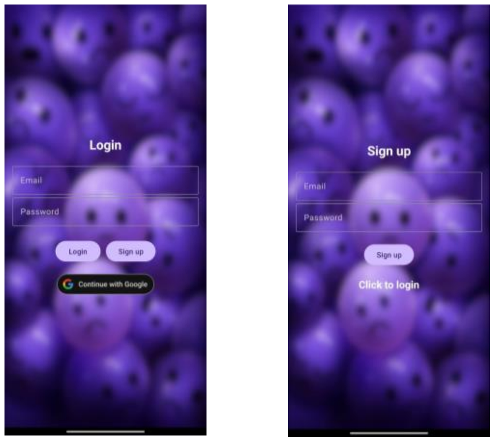


Figure 2. Login Activity and SignUp Activity

The authentication process is handled by the Firebase service, which allows the user to sign in to the application using their Google account. Firebase was chosen due to its ease of interaction and its easy scalability since is a cloud solution. Upon receiving the authenticated session from Firebase, the device will proceed to invoke a call to the Google Maps API to display the map on the smartphone, which serves as the core functionality of the application.

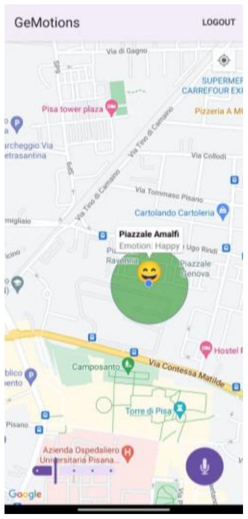


Figure 3. ***Main Activity***

All subsequent interactions occur with Firestore and Google Cloud Service. Firestore [3] it's a solution proposed by Firebase, which allows easy access given that the latter has already been adopted for the authentication phase. Firestore is a Document database that allows the management of TTLs easily, therefore allowing the automatic deletion of elements after a certain period of time. This is great for our solution, as we don't want to keep all the audio from (potentially) the whole world forever in the database, but only those relating to the last 24 hours being the most significant ones for determining the emotion in an area. Furthermore, Firestore allows sorting and composite filtering, something that RealTime database, its counterpart proposed by Firebase, does not allow. Finally, with a view to scalability, it hosts data in multiple data centers in distinct regions, ensuring global scalability and strong reliability. Google Cloud Service [4], on the other hand, is used to store audio. GCS was chosen as it is a cloud solution well integrated with FireBase, allows you to manage TTLs and, for each upload, allows you to retrieve a URL that can be used for download. The fact of having a URL for the download is very useful for our purposes, in fact it allows us to store only the URL of the audio in each Firestore document, bringing the following advantages: (1) save space in Firestore Database; (2) Firestore's maximum document size is not reached; (3) it allows you not to have to download all the audio every time the map is clicked to view the dominant emotion in the area, but to download them only if the user decides to listen to them.

2.1 Document and recordings download

In order to avoid to download all the recordings once the user click on map, we decide to require only a certain amount of document to Firebase (in our application we set up this threshold *document_threshold* = 30). When the user wants to play audios associated to a certain area, they need to be downloaded: the audios are downloaded one at a time in batches of 3 from GCS, after recovering the url from each document. After downloading, they are then displayed in the interface and are ready to be played.

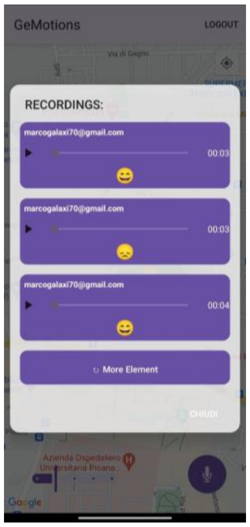
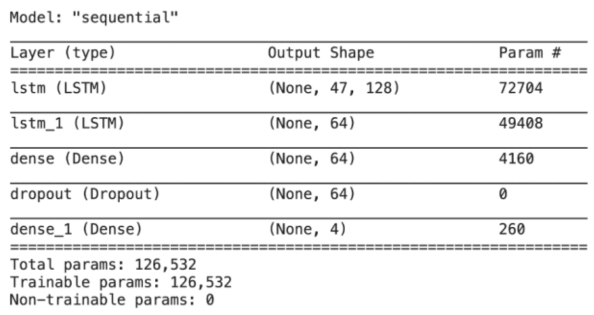


Figure 4. ***Audio playback interface***

When the user wants to record and upload an audio, microphone and user location access permissions must be active, and GPS must also be enabled. After recording the audio and once the neural network has predicted the associated emotion, if the user decides to upload the audio, it is uploaded to GCS, and the associated URL is retrieved. Subsequently, the user's current location coordinates, user email, and download URL are uploaded to Firestore.

2.2 Neural Network architecture

Two different deep learning models were utilized for emotion recognition: a quantized *TFLite* model and the *OpenVokaturi API*. Regarding the *TFLite* model, it was downloaded from the repository [5] *https://github.com/Hannibal0420/Speech-Emotion-Recognition-TinyML*, which occupies 150 KB of memory. It was obtained following quantization from a float32 model (512 KB) to an int8 model, with a negligible loss of model accuracy but an improvement in response times up to **0.299** seconds. This model requires audio spectrograms as input, as the features used by the model are extracted from them. Therefore, audio preprocessing was performed using the *TarsosDSP* library to sample the audio and obtain the so-called (Mel-Frequency Cepstrum coefficient) *MFCC* features. These features are then fed as input to the neural network, which has the following structure:


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 47, 128)           72704
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 126,532
Trainable params: 126,532
Non-trainable params: 0
```

Further tuning of the model's output values was conducted to enhance its performance, resulting in an accuracy of approximately **73**%.

As for the *Vokaturi* API [6], the *'.aar'* extension file was downloaded from the link https://vokaturi.com/downloads/download-the-sdk and occupies 150 KB of memory. The neural network is formed by 3 linear layer, each one followed by a ReLu activation function and the output layer presents a softmax. The Neural Network has an accuracy of **67**% for the free version that we downloaded.
In practical terms, the *TFLite* model proves to be significantly superior in terms of emotion recognition and was therefore chosen, despite its higher energy consumption as shown in the following section.

3 Power consumption tests

Three power consumption tests were conducted: the first one compared the energy usage of two machine learning models, the second one evaluated various configurations for the number of documents downloaded at a time; lastly the third one analysed the GPS incidence during the application usage. The common test methodology for all experiments involved running the application solely on the system without the device being connected to a charger. For each measurement, an average was calculated based on approximately twelve independent repetitions. The tools utilized included the *Android Debug Bridge (adb)* for computing metric dumps and *Battery Historian* for visualizing the metrics on a dashboard.

3.1 Machine learning models consumption test

The test involved employing deep learning classifiers for a duration of 90 seconds, during which both models performed an equal number of classifications on recordings of identical sizes. The obtained results are as follows:

model	**Vokaturi model**	**TFLITE model**
average battery usage [%]	0.13	0.41

These low percentages are attributed to the short duration of the test compared to the device's battery life. Additionally, there's a notable difference between the two models: the *TFLITE* model exhibited higher average energy consumption compared to the *Vokaturi SDK*, which might seem counterintuitive given that the *TFLITE* model is expected to be more efficient. However, it's crucial to note that the test also considers the preprocessing of the audio data, which differs between the two scenarios. The *Vokaturi SDK* requires a *WAV* file, while the *TFLITE* model necessitates a spectrogram computed in real-time using the *TarsosDSP* library. Consequently, conducting more fine-grained tests excluding the preprocessing part and focusing solely on the inference time of the models may be necessary.

3.2 Different number of requested documents consumption test

The purpose of this test was to evaluate potential variations in battery usage when downloading different numbers of documents per request. In the test scenario, 60 documents were downloaded without playback, and the test duration was fixed at 50 seconds of usage. Three distinct values were utilized for the *document_threshold* parameter.

document_threshold	**5**	**15**	**30**
Average battery usage [%]	0.25	0.19	0.16

Despite the proximity of the values, our initial expectation holds true: loading more documents at once can indeed optimize battery usage.
On the other hand, not all documents in a certain area must be downloaded, for two reasons: it would increase memory usage on the device and increase latency, making the application less responsive. Empirically, it has been decided to set the parameter document*_threshold = 30.*

3.3 GPS usage analysis

The application requires the use of GPS to send a recorded audio associated with the user's current location. This test aims to quantify the impact of GPS usage in terms of energy consumption. From the conducted tests, it is evident that GPS usage increases power consumption by approximately **0.1%** for a two-minute usage of the application. Since having the GPS active or not does not significantly affect battery consumption, keeping it active makes the application more user-friendly since having the GPS active is necessary to record audio.

4 Future developments

Possible future developments could include:

- **Model customization**: Since people express their emotions differently, this would enhance the model's performance. Note that now, the audio recording interface shown in Figure 4 allows to specify the emotion and using that instead of the one identified by the model. This interface is currently used to populate the database with different emotions (since it's challenging to fake emotions), but it could be utilized for model customization.
- **Implementation of a multimodal classifier:** To improve classifier performance, data from a smartwatch, such as heart rate, could also be used.
- **Introducing a concept of followers/following:** This implementation is necessary for two reasons: (1) privacy, as a user may not want everyone to listen to their audio; (2) when a user downloads audio in a densely populated area, not having to download documents related to recordings from all users helps reduce latency and memory usage.
- **Searching for areas with a specific emotion in a city:** This feature could be particularly useful for tourism purposes, as mentioned in the introduction.

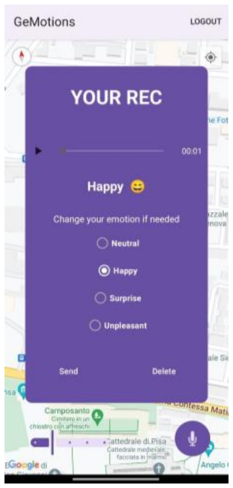


Figure 1. ***Audio recording interface***

Riferimenti

[1] Y. Doytsher, B. Galon e Y. Kanza, «Emotion Maps based on Geotagged Posts in the Social Media,» *GeoHumanities'17,* p. 8, 2017.

[2] R. Wampfer, S. Klingler, B. Solenthaler, V. R. Schinazi, M. Gross e C. Holz, «Affective State Prediction from Smartphone Touch and Sensor,» p. 14, 2022.

[3] «Cloud Fire Store,» [Online]. Available: https://firebase.google.com/docs/firestore?hl=it.

[4] «Inizia con Cloud Storage su Android,» [Online]. Available: https://firebase.google.com/docs/storage/android/start?hl=it.

[5] «Speech Emotion Recognition using TinyML,» [Online]. Available: https://github.com/Hannibal0420/Speech-Emotion-Recognition-TinyML?tab=readme-ov-file.

[6] Vokaturi. [Online]. Available: https://github.com/alshell7/vokaturi-android?tab=readme-ov-file.