Live It - Recommendation System based on Emotion Detection

Diana Isabela Crainic
Faculty of Computer Science,
"Alexandru Ioan Cuza" University
General Berthelot, 16,
Iasi, Romania
diana.crainic@info.uaic.ro

Adrian Iftene
Faculty of Computer Science,
"Alexandru Ioan Cuza" University
General Berthelot, 16,
Iasi, Romania
adiftene@info.uaic.ro

ABSTRACT
This paper presents the development of a web application that integrates a system of movie recommendations using the collaborative filtering algorithm with a component of real-time recording and detection of emotions. So far, there were no implementations that combine recommendations and emotions, so this application proposes these two to work together to make our lives better regarding the movie-watching experience. To recognize emotions, we created a component that examines facial expressions, which offers, as a result, one of the emotion types: *happy, sad, neutral, anger, surprise.* This component was later integrated into a movie recommendation application, which analyzes the user's emotions in real-time while watching the presentation video. In the second part of the paper, we presented how we performed usability tests in order to improve the quality of the application. The results were promising, with a high degree of accuracy and usefulness coming from end-users, showing the future potential of this application, for instance adding new functionalities or recommendation algorithms.

Author Keywords
Face recognition; emotions identification; recommendation system.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces. H.3.2. Information Storage and Retrieval: Information Storage.

General Terms
Human Factors; Design.

DOI: 10.37789/rochi.2021.1.1.7

INTRODUCTION
The motivation to build a movie recommendation system comes from the fact that choosing a movie that suits one's preferences is a time-consuming process. One solution would be to use a prediction algorithm that simplifies this process, and that takes into account the watched movies and how they were evaluated.

In our case, we considered both the direct evaluation (through the movie ratings), but also the indirect one (by identifying the feelings of the user when watching the movie trailer).

The process of emotion recognition is composed of the following steps (1) identifying the area of the face in the image, (2) extracting features, and (3) classifying in one of the available classes, represented by one of the emotions.

The first real-time emotion recognition paper, which is also a source of inspiration for the component of the application responsible for recording, is *Facial Emotion Recognition in Real-Time*, a Stanford University piece of work [6].

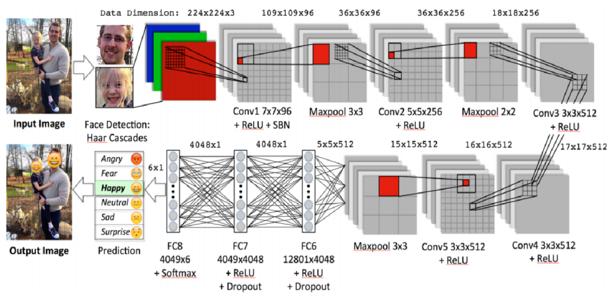

Figure 1. The neural network architecture of the Facial Emotion Recognition application in Real-Time [6].

Figure 1 illustrates the architecture of the neural network used in this application. Having an image that contains one or more faces, the face area will be identified, it will be involved in a series of operations in the convolutional neural network. In the end, once the emotion is obtained, an emoticon will be applied over the defined area in the first step. Considering this paper as a state of art, we optimized this process of emotion recognition, choosing from a wide range of activation functions the one with the best result. For the six classes used (*anger, fear, happiness, neutral, sadness,* and *surprise*), the accuracy obtained is 57.1%.

Movie recommendation systems

Netflix is a streaming service that offers a wide variety of award-winning movies, series, and documentaries [14]. But an important topic is the process of recommendation to users, especially when there is no data on the viewing history (cold start problem). Netflix has organized a competition called *Netflix Prize* to develop the best collaborative filtering algorithm to predict users' ratings for movies based on other users' previous ratings [15]. Thus, the generated movie recommendations will be suitable for the user's profile, depending on his appreciation. The competition ended in 2009 when BellKor's team achieved the best score for RMSE (Root Mean Square Error), of 0.8567 and thus winning the grand prize worth $ 1 million [16]. Since then, based on this prediction algorithm, millions of users daily receive recommendations. Considering this type of algorithm, which is used for the Netflix platform, we have chosen the same one, collaborative filtering to make recommendations, providing the most five suitable movies for the user. Moreover, we have integrated with an emotion detection module which shows in real-time the predominant experienced emotion of the user while watching the trailer of the movie.

TMDB (The Movie Database) is another movie recommendation system, which also provides to developers an API service for collecting and using data about existing movies [20]. The algorithm used to generate recommendations is based on collaborative filtering, which analyzes the list of favorite movies, their ratings, and compares it with lists of users with similar preferences. As regards the API that provides information about movies, actors, genres, and users, through endpoints and an authentication key, offers the possibility to extract the data you want to use in your application. As part of the database of the application, we have operated with a reduced set of movies choosing those which have the highest ratings, to improve the reliability of these suggestions.

Recognition of Emotions

Emotions can be considered a mechanism that facilitates interactions between human beings. Therefore, understanding them can be a complex process that requires a lot of resources. An emotion can be identified by a variety of methods such as tone of voice, body language, or even through an electroencephalogram. But the simplest and most practical method is to examine facial expressions [8].

Over time, it has been proven that there are seven types of emotions that can be identified in all nationalities: *happiness, sadness, surprise, anger, neutral, disgust, fear* [9].

Recognizing emotions based on images can be considered a difficult process for several reasons: it is tedious to collect enough images to train the model, and the classification of emotions will depend on input data received: static or dynamic images - transition to a new emotion, brightness, clarity, position of the subject, etc.

MODEL CONSTRUCTION

In the context of emotion recognition, our model was developed through a convolutional neural network. The motivation for choosing a model based on deep learning is supported by the results provided by this type of network in Computer Vision problems, which use large collections of two-dimensional data, such as images or even videos. A video, as a sequence of images, provides new information about the action, therefore allows for deeper situational understanding. For example, we can track an obstacle through a sequence of images and understand its behavior to predict the next move. We can track a human pose, and understand the action taken with action classification [15].

In the scenario in which we want to identify the facial expression, we will use the classification algorithm, as it will produce a probability of belonging to a certain type of emotion (*happiness, sadness, surprise, anger,* or *neutral*).

Having at our disposal a database with images depicting people who transmit a certain emotion, among those mentioned above, we aim to build a classification filter to obtain the percentage that a certain image fits into one of the classes. To achieve this, the filter must be able to predict what is unique to the emoticon, take from the image and make a correlation with the images in the collection [10] (see Figure 2).

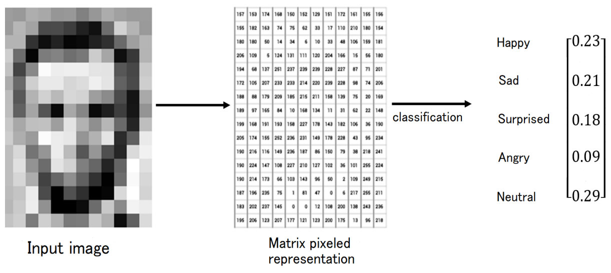

Figure 2. Image classification [1].

Difficulties in identifying objects can occur for various reasons: lighting conditions, variation in space of the region of interest, scaling, deformation, uneven background, or occlusions. In the context of learning problems, the essential goal is to minimize the cost of loss function by optimizing the weights of the network, in order to finally obtain an expected accuracy.

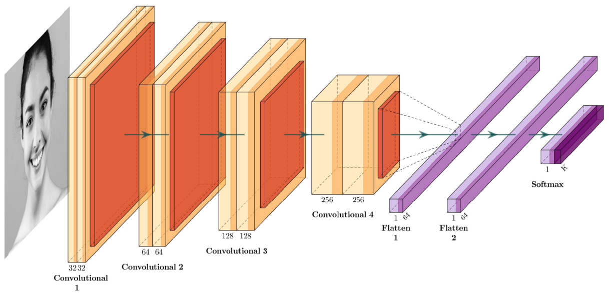

Figure 3. Diagram of the used convolutional neural network.

Fixing the right hyperparameters plays an important role in neural network architectures, having a direct impact on training the model. This can be a difficult process, as there are numerous possibilities to change these configuration parameters.

Figure 3 illustrates the architecture of the convolutional neural network we used. The network consists of seven components, of which the first four are responsible for the convolution operation, the next two for flattening, and the last one is represented by the Softmax function.

The first four blocks of the model are composed of two convolutional layers, between them being made Batch Normalization, and at the end, the Max Pooling and Dropout operations are applied. The activation function used in the case of convolutional layers is ELU (Exponential Linear Unit) [5].

Batch normalization is a technique for deep learning neural networks that standardizes input data in a single layer for each subset. This approach results in stabilizing the learning process and substantially reducing the number of epochs required to train the network. The Max Pooling process aims to extract the highest value from the area of interest, more precisely the portion that contains the information with the highest degree of importance. Dropout operation reduces the importance of a certain set of neurons in the process of training the neural network.

The purpose of the process is to prevent overfitting (learning by heart), i.e., overloading the network with additional connections between neurons. Thus, omitting certain randomly selected neurons can help generalize the model, namely the ability to adapt to new data that has not been processed.

Blocks five and six are responsible for the Flatten, a Dense layer with the function of ELU activation, Batch normalization, and Dropout operations. Flattening transforms a two-dimensional array of features into a vector which in turn represents input data for the next layer. Dense adds to the neural network this layer that is fully connected, which means that each neuron receives input data from the previous layer. The last block is responsible for performing the Dense operation on the model using the Softmax activation function, also specifying the number of classes that are used, in the number of five.

The model thus created is also configured with hyperparameters, the next procedure being training, and validation based on the input data specific to each operation.

Image collection

In the process of creating and training the model, five classes were used, more exactly five types of emotions: *happiness, sadness, surprise, anger,* and *neutral*. Emotions of disgust and fear have been omitted, as these 2 emotions are very similar in terms of facial mimicry to existing ones: *angry*, respectively *surprise*.

Each of the five emotions mentioned had a collection of black and white photos, with reduced size of 48 x 48 pixels, men or women, of various ages and nationalities. This set of data was taken from *Kaggle* and consists of two parts, the first for the training process and the second for the validation process [12].

Figure 4 shows some examples of photos from the data collection for *happiness* and *anger*.

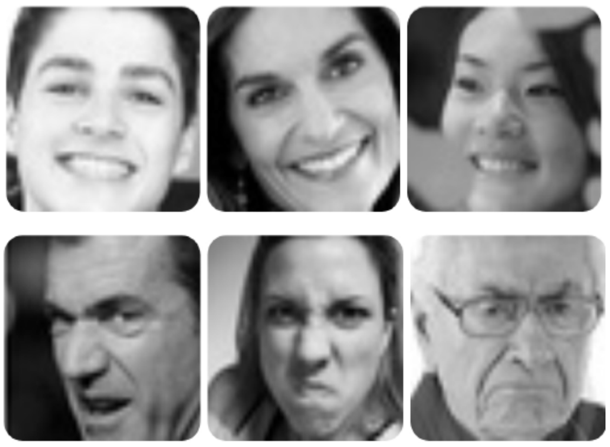

Figure 4. Kaggle images examples of happiness and anger.

Figure 5 shows the distribution of the number of photos in the data collection for training, the predominant type of emotion being happiness, followed by a relatively equal number for sadness and neutral, and emotions of anger and surprise having a smaller number of photos. The total number of images in the collection is 24,282.

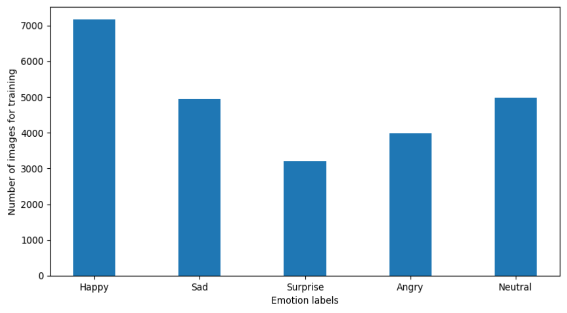

Figure 5. Distribution by class of data used for training.

The distribution of photos in the test data collection was similar to that in the training image collection, but this time the total was significantly lower.

RESULTS

According to Table 1, it is observed that for *batch size* = 16, from the point of view of the accuracy metric, we obtained a higher value in comparison to 8 or 32. These experiments were performed for the ELU activation function and the number of epochs was set to 10.

Batch size	Accuracy	Error	Duration
8	45.66 %	1.2824	~ 3h
16	**58.28 %**	1.0503	~ 3h
32	50.78 %	1.2078	~ 3h

Table 1. Results for different batch sizes (10 epochs).

Once the values of the *batch size* = 16 and *number of epochs* = 10 parameters were fixed, we performed a series of experiments whose results are summarized in the following table for five types of activation functions, to compare their performance.

Function	Accuracy	Error	Duration
ELU	**58.28 %**	1.0503	~ 3h
RELU	53.73 %	1.1261	~ 3h
Leaky RELU	41.52 %	1.3468	~ 3h
Swish	45.81 %	1.2942	~ 3h
Mish	32.78 %	1.5278	~ 3h

Table 2. Results for different activation functions.

In Table 3 the ELU activation function that obtained the best score is tested for a larger *number of epochs* (25) and variable *batch size*: 16, 32, respectively 64.

Batch size	Accuracy	Error	Duration
16	64.27 %	0.8968	~5h
32	**64.70 %**	0.8958	~ 5.5h
64	64.14 %	0.8965	~ 5h

Table 3. Results for different batch sizes (25 epochs).

According to table 3, we deduce that the ELU function for a *batch size* = 32 and the *number of epochs* = 25 obtains the highest percentage for the accuracy metric, i.e., 64.7%.

Discussion

Training the model with different configurations starting with epochs number, activation function, ending with batch size, has led to the best result regarding classification accuracy. Compared to other similar scientific papers, the resulting accuracy percentage is comparable.

Thus, in the paper [6], the authors mention the obtaining of an accuracy percentage of 57.1%, using 6 classes of feelings (with one class more than we used). And in the work [7] the authors obtained an accuracy of 65%, in the context of using the same number of classes of emotion.

Haar-Cascade classifier

In this process of recognizing emotions, we also used a Haar-Cascade classifier [21] filter provided by the OpenCV library, through which people's faces were identified. More specifically, the real-time recording of people is decomposed into frames, for each image is applied a black and white filter, over which several operations are then applied through the *detectMultiScale* function and thus the face area is detected and cut.

If the face has been identified, the function will result in a set of rectangular coordinates (*x, y, w, h*) that will represent the searched area and thus the region of interest of the face can be found. Having the model that was created based on the data collection and through the region of interest just calculated, a series of operations are performed, and a list of probabilities is obtained for each class. The defining emotion for the current frame will be the class with the highest value.

Once the emotion has been calculated and obtained for the image received at the input, the application will display one of the emoticons from Figure 6 to describe the user's facial expression when viewing the video presentation of the chosen movie.

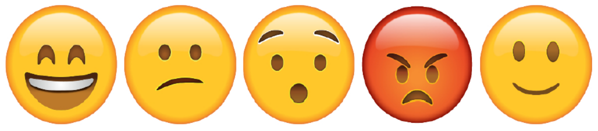

Figure 6. Representative emoticons for Happiness, Sadness, Surprise, Anger, and Neutral [2].

Integration with the recommendation application

The usefulness of this component in the recommendation system appears when watching the trailer of the movie both to recognize and show the emotions experienced by the user, and to identify the predominant emotion experienced by him.

The emotion recognition component is a Django API application that provides an endpoint to the external so that it can be used later. The second application is represented by the Angular client, the visual interface of the recommendation system is the one that will call the endpoint of the Django application, thus communicating through HTTP requests. Once the client sends a request, it starts recording and identifying the emotions, and the currently identified emotion will be sent in response.

One of the five emoticons from Figure 6 will be displayed on the frontend, depending on the identified emotion. Also, the predominant emotion will be calculated in real-time, to identify the user's interest in the trailer of the movie.

RECOMMENDATION APPLICATION

The movie recommendation application is made in the model of a microservices architecture, orchestrated by an API Gateway [3], [11]. The microservice-based architectural style is a way to develop an application as a suite of small, autonomous services that communicate through various mechanisms, often using an HTTP resource API. These services are built around a single functionality of the system and can be delivered independently. They can also be developed with different technology infrastructures and have their own database [9].

The application consists of four microservices:

(1) User management microservice;

(2) The microservice responsible for movies;

(3) Movie evaluation microservice;

(4) Prediction microservice.

They communicate with each other and are managed by the API Gateway, which serves as an intermediate level between the client application and each microservice.

User management microservice

The user management microservice is responsible for creating an account for a new user and authenticating it. The database of this microservice contains information related to the user entity, which is identified by the properties: *id, account name, email, password*, as well as a *role*, predefined as a generic user.

In a user registration scenario, the expected data is the *username, email address,* and *password.* Those mentioned above were verified and validated so that the account is successfully created. The chosen password is passed through a cryptographic function (SHA256) and the result is saved in the database.

Movie microservice

The microservice responsible for *movies* and the details about them, such as the *actors* and the *genres*, has as a database scheme several tables, both for specific information and for creating relationships. The movie-specific table contains the properties: *id, title, director, the country* where the movie was produced*, release year, ratings* from user scores on the TMDB movie page*, description*, *YouTube link* to the movie presentation, *the link to the poster,* and the *ids of the movies* similar to the current one. The source of all this data (*movies, actors, genres*) comes from the API provided by TMDB [20].

Evaluation microservice

The movie evaluation microservice is designated for authenticated users in the application who want to give a rating from 1-10 to the watched movies. An authenticated user can *rate movies, edit a rating*, or *delete it.* Based on these ratings and depending on the preferences of other users, further movie recommendations will be provided.

The database of this microservice was populated with data from Kaggle [13], with movies, users, and the rating values which were translated in the 1-10 range. Thus, the set of training data necessary for the machine learning model was created.

The movie evaluation microservice communicates with all the other microservices: the user management service for displaying the ratings, the one responsible for movies with the aim of showing them, and the one for predictions that provide recommendations based on the ratings.

Prediction microservice

The Prediction microservice is responsible for providing movie recommendations to users who have rated at least one movie. The algorithm chosen to create the model is collaborative filtering, which is based on other users' common movie preferences.

The first step was to collect the data, which are in fact those used in the movie rating microservice, the columns of the table being represented by the evaluated *movie id, the user id,* and the *rating* given.

The column chosen to be predicted is the *value of the rating*, which will represent the score in this movie recommendation context. The next step is the training one, at which time must be specified depending on the size of the given file, in this case, the period being 600 seconds. The evaluation of the model consists of a test for predicting the score (*rating*) according to the specified parameters (*user id* and *movie id*). And in the final step, of generating the code, an output file will be generated, which represents the model, as well as two projects, which will be used later in the application.

API Gateway

API Gateway is a concept and also a model developed in parallel with microservices. It started from the architectural design model Facade and thus allows the aggregation of functionalities from several microservices. This API Gateway has access to all the microservices endpoints and serves as a bridge between the client and the microservices. The client will send requests in a certain format, API Gateway will interpret them, make calls to microservices, retrieve information, aggregate it and return it to the client.

Recommendation system

A recommendation system provides suggestions to users through a filtering process based on their preferences and history. User information, which is perceived as input data, reflects previous uses, along with assigned ratings. Thus, the system analyzes preferences and provides predictions based on user choices. These systems are used in a variety of fields such as movies, music, news, books, or scientific articles [18], [19].

The algorithm used for our recommendation system for providing predictions is *matrix factorization* [17]. Through it, can be identified the relationships between entities: *user* and *movie*. Having the ratings given by users, it is possible to predict how a user will evaluate a certain movie.

The recommendation methods used by the matrix factorization algorithm fall into two broad categories: (1) collaborative filtering and (2) content-based filtering. The type used in this application is collaborative filtering.

Collaborative filtering uses the similarity between users' preferred options to make subsequent predictions based on similar preferences of other users. Thus, this resulting model offers recommendations to user A based on the interests of a similar user B. If two users have watched the same movie, a similar relationship is established between them. Thus, if one of them watched and appreciated a certain movie, it will be recommended to the second user if this movie has not already been viewed and appreciated.

The recommendation system has as training data a feedback matrix in which each row is represented by a user, and each column a movie. Movie feedback is divided into two types: (1) explicit and (2) implicit.

The method used by the ML .NET library is the explicit one, in which the users specify the degree of appreciation for a certain movie, giving specific ratings.

Once the model is generated, movie recommendations for a specific user are generated as follows: scroll through the list of movies that have not already been rated by the user, consume the model to extract the predicted score for the current movie, and add it to a dictionary, <*Movie, Movie Rating*>. This dictionary is then sorted in descending order by the value of the rating to have at the beginning of the movies most likely to be appreciated by the user. Finally, the first 5 movies whose prediction values have the highest score will be displayed in the visual interface.

Among the advantages of using this method is the high possibility to discover new movies of interest to users, a wide variety of movies and in addition, it is not necessary a large amount of information about the user, but only a minimal selection of popular movies. Regarding the disadvantages, a problem would be the fact that when a new movie appears, it will not be recommended to users, until after subsequent retraining of the model.

On the other hand, content-based filtering uses the characteristics of a product to recommend products similar to what the user liked, based on his previous evaluations. Once a user has watched a movie, and there is a movie like it in the database (in terms of *genres, actors*, and *release year*), it will be recommended. An advantage of this approach is that movies from the same area of interest as the user will be recommended. As for disadvantages, this method requires a lot of knowledge in the field, as well as limiting the recommendations to the evaluated movies.

USABILITY TESTS

In the context of testing the application, we created a Google Forms form with several questions addressed to real users of the *Live It* application, to use the opinions to introduce further improvements.

Methodology

The test of interaction with the application contains an introduction in which its role is presented, and it is composed of seven questions, with a free answer or single choice. Before answering this test, the subject followed a series of scenarios in the application, each session lasted an average of ten minutes, and the steps were explained as follows:

Step	Description
1	Registration page: creating an account
2	Login page: user authentication
3	Movie page: View movie information
4	Movie page: give ratings
5	Movie page: search for a movie or browse the page list
6	Trailer page: watching it
7	Trailer page: recording and detecting emotions
8	Trailer page: view similar movies

Table 4. Usability tests – steps description.

Participants

For evaluation, we collaborated with 8 people, aged between 18 and 55 years, remotely in their homes. The group was diverse, consisting of both technical and non-technical people, experienced in computer use, young people, adults, with different frequencies of watching movies to simulate a real use audience.

As part of the study, the first two questions asked were related to the participants' experience in watching movies. The first refers to the frequency of watching movies, 50% answered that they watch 2-3 times a week, the rest being either in the group of four times a week (25%), or once a week (25%). The second question is related to the length of the process of searching for a movie that is to the liking of the subject. In proportion of 25% answered that it takes them a lot of time, and 62.5% answered that it takes a medium amount of time.

Results

The next section of the questionnaire included questions about each person's experience with the *Live It* application. According to the observations, the percentage of participants who considered it to be useful is high (see Figure 7).

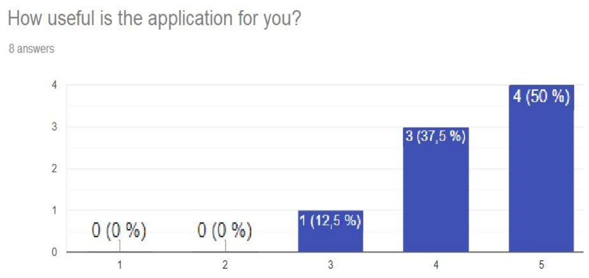

Figure 7. Results regarding the usability of the application.

Therefore, the degree of usefulness of the recommender algorithm is high, so they received suggestions of movies to their liking, without spending much time in this search process.

In the next section, the users of the application had to give a grade from 1 to 5, where 5 was the maximum grade, in connection with various aspects related to the application: (1) ease of use, (2) response time of application, (3) framing the elements on the screen, (4) design. These results are shown in Figure 8.

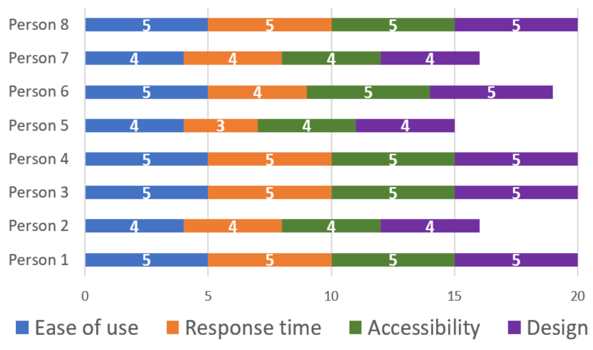

Figure 8. Results regarding various aspects of the application.

In the end, we asked the participants to offer some suggestions for improving the application. The first suggestion mentioned is to provide a link for watching the entire movie. Adding this option would create a relationship between the movie and a list of streaming services, which usually require paying a subscription. Therefore, the user will be able to choose the desired provider. A second proposal was to offer the possibility to add a new movie to the application. This can be implemented in a future version of the application but will require specifying all the details related to the desired movie, this information is extracted using the API provided by TMDB, only if it already exists in the API database.

CONCLUSION

The *Live It* application aims to simplify the search process for a movie that will be to the user's liking. Through this interactive and friendly mode, the application provides features such as: view available movies along with details about them, give ratings, watch the trailer along with recording and recognizing real-time emotions of the viewer, offer recommendations based on evaluated movies, but also similar movies.

The development potential for this application is represented by the recommendation system, which can be extended by adding new functionalities or recommendation algorithms. Some examples would be the creation of a forum where users can initiate discussions about movies, the ability to add comments along with the given rating, comments that can be seen later by other users. Also, another alternative is the evaluation of movies based on several criteria such as actors, genres, or director and thus the design of new algorithms, which can bring value to recommendations.

ACKNOWLEDGMENTS

This work was supported by project REVERT (taRgeted thErapy for adVanced colorEctal canceR paTients), Grant Agreement number: 848098, H2020-SC1-BHC-2018-2020/ H2020-SC1-2019-Two-Stage-RTD.

REFERENCES

1. Amini, A., and Soleimany, A. Introduction to Deep Learning. *MIT's official introductory course on deep learning methods with applications*, Massachusetts Institute of Technology, (2020).
2. Apple Emoji Faces, Emoji Pictures, https://emojiisland.com/pages/free-download-emoji-icons-png (2021)
3. Baboi, M., Iftene, A., and Gîfu, D. Dynamic Microservices to Create Scalable and Fault Tolerance Architecture. In *23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Procedia Computer Science, 159* (2019), 1035-1044.

4. Ciubotariu, C.C., Hrișca, M.V., Gliga, M., Darabană, D., Trandabăț, D., and Iftene, A. Minions at SemEval-2016 Task 4: or how to build a sentiment analyzer using off-the-shelf resources? *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, ACL, (2016), 247-250.
5. Clevert, D.A., Unterthiner, T. and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *4th International Conference on Learning Representations ICLR (Poster)* (2016).
6. Duncan, D., Shine, G., and English, C. Facial Emotion Recognition in Real Time. *Report Stanford University* (2016), 1-7.
7. El Ali, A., Wallbaum, T., Wasmann, M., Heuten, W., and Boll, S. Face2Emoji: Using Facial Emotional, Expressions to Filter Emojis, In *2017 CHI Conference Extended Abstracts* (2017).
8. Ekman, P. Universals and cultural differences in facial expressions of emotion. Nebraska, USA: Lincoln *University of Nebraska Press* (1971).
9. Fowler, M. Microservices. https://martinfowler.com/articles/microservices.html (2014)
10. Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning, *MIT Press*, (2016).
11. Iftene, A., Gîfu, D., Miron, A. R., and Dudu, M. Ș. A Real-Time System for Credibility on Twitter. In *Proceedings of The 12th Language Resources and Evaluation Conference* (LREC 2020), Marseille, France, (2020) 6168-6175.
12. Kaggle - Face expression recognition dataset https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset, (2018).
13. Kaggle - Large Movie Dataset https://www.kaggle.com/chaitanyahivlekar/large-movie-dataset (2021).
14. Machine Learning & Deep Learning, https://www.thinkautonomous.ai/blog/?p=computer-vision-from-image-to-video-analysis (2021)
15. Netflix, https://www.netflix.com/ro-en/ (2021).
16. Netflix Prize, https://www.netflixprize.com/rules.html (2021).
17. Rendle, S., Krichene, W., Zhang, L., and Anderson, J. Neural Collaborative Filtering vs. Matrix Factorization Revisited. *Fourteenth ACM Conference on Recommender Systems* (2020), 240–248. doi:10.1145/3383313.3412488.
18. Ricci, F., Rokach, L., and Shapira, R. Introduction to Recommender Systems Handbook. *Recommender Systems Handbook, Springer*, (2011), 1-35.
19. Șerban, C., Alboaie, L., and Iftene, A. Image and user profile-based recommendation system. In *Workshop on Social Media and the Web of Linked Data (RUMOUR 2015) at EUROLAN 2015 Summer School on Linguistic Linked Open Data*. Springer International Publishing Switzerland. EUROLAN 2015, CCIS 588, (2016), 1-16, doi: 10.1007/978-3-319-32942-0_5.
20. TMBD, https://www.themoviedb.org/ (2021).
21. Viola, P., and Jones, M. J. Rapid Object Detection using a Boosted Cascade of Simple Features. Proceedings of the *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR* (2001), 511- 518.