Expert Systems With Applications 198 (2022) 116706

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid multidimensional Recommender System for radio programs

Antonio Jesús Fernández-García [b,*], Roberto Rodriguez-Echeverria [a], Juan Carlos Preciado [a], Jorge Perianez [a], Juan D. Gutiérrez [a]

[a] Quercus Research Group, University of Extremadura, Avenida de la Universidad s/n, 10003, Cáceres, Spain
[b] Universidad Internacional de la Rioja, Avenida de la Paz, 137, 26006 Logroño, La Rioja, Spain

ARTICLE INFO

Keywords:
Recommender system
Content-based
Ensemble of recommenders
Hybrid recommender
Radio programs

ABSTRACT

The rise of Recommender Systems has made their presence very common today in many domains. An example is the domain of radio or TV broadcasting content recommendations. The approach proposed here allows radio listeners to receive customized recommendations of radio channels they might listen to based on their specific preferences and/or historical data. Firstly, a *Data Acquisition System* is presented with its main task being to obtain and process data to pass to recommenders. Secondly, a dynamic hybrid Recommender System is developed based on four dimensions reflecting major aspects of radio programs: relative talk/music percentages, music genres, topics covered, and speech tone. Eight recommenders are constructed (two per dimension) using *content-based* or *collaborative filtering* algorithms depending on the nature of the data processed, whether historical data or user preferences. And thirdly, by assigning weights in accordance with the users' preferences, a dynamic ensemble of these recommenders is formed which produces the final recommendations. Experiments were carried out illustrating the usefulness of the recommendations and its acceptance by radio listeners.

1. Introduction

The spectacular growth of data, the increasing ability of companies and institutions to collect it, and the rapid advances in Data Mining, Artificial Intelligence, and Machine Learning have made possible the creation of Intelligent Systems, Virtual Assistants, and Recommender Systems that assist users in making more suitable purchases, in accessing content of their interest, and in general in their decision-making. Recommender Systems have made it possible to provide users with items or content of considerable value or interest for them. Recommender Systems play an undisputed fundamental role in this regard in companies such as Amazon where 30% of its page views are from recommendations, or Netflix where 80% of the content watched comes through recommendations (Gomez-Uribe & Hunt, 2016; Smith & Linden, 2017).

With respect to online radio broadcasting, the case study presented in this communication is aimed at helping radio channels engage with their audiences, given that competition in this sector has grown as the number of broadcasting channels has increased together with access to a wider choice through the Internet or the emergence of new ways of consuming content beyond radio such as podcasts. Given the continuous increment of TV and radio broadcasting channels, recommender systems can play a relevant role to properly lead users to specific content according to their needs and interests (Park, Oh, & Yu, 2017). In this sense, main media corporations might find interesting to implement Recommender Systems to encourage radio listeners' loyalty to their radio stations.

For this reason, it makes sense to create a Recommender System designed as an Assistant that allows radio audiences to receive, based on their likes and dislikes, interests, and other criteria, customized recommendations of programs they might like that are being broadcast among the conglomerate's different radio stations.

To be able to create a Recommender System or Predictive Model, a certain amount of data is required. Normally, Recommender Systems (a) capture the patterns of users' behavior to predict what they might like (*content-based filtering*), or (b) analyze users with similar behaviors or who have similar tastes in their lives in order to make recommendations (*collaborative filtering*). In this communication, we propose an approach to creating a radio program Assistant that aims to work not only with registered users who have a track record of behavior, but also with anonymous or newly registered users.

Given this situation, it is necessary to make use of recommendation approaches that rather than using only historical data, or similarities between users use the radio program content's intrinsic information and

* Corresponding author.
E-mail addresses: antoniojesus.fernandez@unir.net (A.J. Fernández-García), rre@unex.es (R. Rodriguez-Echeverria), jcpreciado@unex.es (J.C. Preciado), jpery@unex.es (J. Perianez), andy@unex.es (J.D. Gutiérrez).

https://doi.org/10.1016/j.eswa.2022.116706
Received 21 December 2020; Received in revised form 3 December 2021; Accepted 20 February 2022
Available online 5 March 2022

the users' preferences to create a Recommender System that produces relevant recommendations for radio audiences. Thus, the proposed approach is to create an Assistant that makes use of (a) historical data if available, (b) pre-processed data about radio stations' programs, and (c) users' preferences in terms of the programs' type, content, tone, and schedule. Additionally, the approach can make use of users' likes or ratings if these are available.

These facts led us to pose the following research questions:

> *Research Question: How can one design and deploy a Data Acquisition System that deals with data repositories, third-party services, and human interaction through user interfaces to create a consistent, reliable, and meaningful dataset?*

In the light of the foregoing, the main innovative contributions of this communication lie in:

- Proposing and deploying a *Data Acquisition System* which captures the users' preferences, significant features from the data sources, and connects to various third-party services to process and transform the data into a consistent and reliable dataset with all the data required to model an accurate Recommender System.
- Designing a *hybrid recommendation strategy* that makes use of historical data (if available), users' preferences at specific given times, and pre-processed data providing invaluable information about radio program broadcasting, and creating such a strategy on the basis of four dimensions: relative voice/music percentages, music genres, topics covered, and speech tone.
- Creating a *four-dimension hybrid* Assistant comprising an *ensemble* (Polley & van der Laan, 2010) of weak content-based recommenders to obtain a joint Recommender System that improves the quality of the simpler learner recommenders by resolving any overfitting, bias, or imbalance issues that might arise (Xia, Chen, & Yang, 2020).

The rest of this communication is organized as follows. Section 2 reviews some related projects that involve working with Recommender Systems. Section 3 provides an overview of the approach, describing the data sources and the methods. Section 4 describes the processes involved in the data acquisition and the processing techniques used to construct the datasets that feed the recommenders. Section 5 describes the recommender algorithms used, and details the two main steps of the recommendation strategy – constructing the recommenders for each dimension, and creating the ensemble of the recommenders output. Section 6 describes the experiments carried to validate the approach and their results. Finally, some conclusions and further considerations are summarized in Section 7.

2. Background and related work

2.1. Background

The considerable increase in the generation and storage of data and the rise of Recommender Systems have made the presence of these systems very common today in many domains. These domains present great heterogeneity: job recommendations (Chamoso, Rivas, Rodríguez, & Bajo, 2018), fashion industry (Dong, Zeng, Koehl, & Zhang, 2020), finance (Bunnell, Osei-Bryson, & Yoon, 2020), education (Fernández-García, Rodriguez-Echeverria, Preciado, Conejero, & Sánchez-Figueroa, 2020; Rodriguez-Marin, Duque-Mendez, Ovalle-Carranza, & Martinez-Vargas, 2020), user interfaces (Fernández-García, Iribarne, Corral, Criado, & Wang, 2019), or science publications (Wang, Liang, Xu, Feng, & Guan, 2018), among many others.

There exist two well-known approaches to creating Recommender Systems. These are *Content-Based Filtering* and *Collaborative Filtering* (Adomavicius & Tuzhilin, 2005). *Content-Based Filtering* Recommender Systems provide recommendations to users based upon the similarity between item descriptions and user interest profiles or item descriptions and items liked by the target user in the past, without directly relying on the preferences of other users. (Barragáns-Martínez, Costa-Montenegro, Burguillo, Rey-López, Mikic-Fonte, & Peleteiro, 2010; Pazzani & Billsus, 2007). Part of the success of these models lies in their designers' ability to properly describe the user profiles and to construct good item representations that capture the essence of those items. A major limitation of models of this kind is that they can only make recommendations on the basis of users' existing interests, and in many cases this limits their applicability. *Collaborative Filtering* Recommender Systems evaluate items on the basis of other people's opinions. Users evaluate items through a rating system which consists of associating two things – user and item – often by means of certain values (Schafer, Frankowski, Herlocker, & Sen, 2007). The advantage of these recommenders is that no domain-specific knowledge is required, and they work well with new users. A major limitation, however, is that they cannot handle new items because they have yet to be rated, and therefore cannot be suggested. There have been many recent works aimed at alleviating these cold-start problems, as they are usually known, by helping to provide suitable recommendations to new users and to introduce new items into their recommendations (Herce-Zelaya, Porcel, Bernabé-Moreno, Tejeda-Lorente, & Herrera-Viedma, 2020; Hernando, Bobadilla, Ortega, & Gutiérrez, 2017; Tang, Qian, & You, 2020). *Collaborative Filtering* may also incorporate *distance-based* techniques as a measure of similarity, as can be seen in Bachrach et al. (2014), or as an alternative latent factor model to implement collaborative filtering (Khoshneshin & Street, 2010). Distance techniques are usually applied by a great number of recommenders in many different areas. In Guo, Deng, Ran, Wang, and Jin (2021) a Hellinger distance-based model is used to measure item similarity between movies (MovieLens dataset) and songs (Yahoo Music dataset). In Wang, Zhang, and Lu (2015) a Manhattan distance-based function is used to measure the relevance between two users between movies (MovieLens dataset) and business partners (SmartBizSeeker dataset). In Hasanzadeh and Forghani (2021) a M-distance based recommender system is used to measure the distances in movies using three different datasets (MovieLens, DouBan, and, Each Movie). In Tran, Liu, Lee, and Kong (2019) a Signed Distance-based Recommender is used to measure the similarities in movies using the MovieLens, Netflix Price and Epinions datasets. In our case, we use a Euclidean distance as a measure for searching the radio program closer to the user preferences.

There have also emerged from the extended use of Recommender Systems and their improvements hybrid approaches which mitigate the problems of *content-based* and *collaborative filtering* algorithms. Hybrid methods combine *content-based*, *collaborative filtering* and other methods to make predictions and recommendations (Kiran, Kumar, & Bhasker, 2020). These hybrid approaches can be categorized according to how they work (Burke, 2002; Walek & Fojtik, 2020). Inter alia, some of these categories are: *weighted*, in which the predictions of several approaches are combined with the possible assignation of different weights to the recommenders; *switching*, in which the recommendation model is chosen on the basis of whether certain criteria are met; *mixed*, in which the outputs of several recommenders are offered to the end users for them to decide; *cascade*, in which the output of a recommender serves as input for other recommenders until the final recommendations are produced; or *ensemble*, in which the predictions generated by weak recommenders are combined to yield a final recommendation. These hybrid approaches, integrating various recommenders beyond the simple integration of their outputs, have been widely applied since the emergence of recommender systems, as shown in Burke (2002) and Çano and Morisio (2017).

2.2. Related work

The domain of content recommendation in radio or TV broadcasting is no stranger to the use of Recommender Systems. The following

paragraphs will present a brief review of some of the related work in the literature.

As an example of digital content recommendation, in Tejeda-Lorente, Porcel, Peis, Sanz, and Herrera-Viedma (2014), the authors propose a new Recommender System that filters and evaluates the vast amount of academic-related information that is available on the Web. The system makes a measure of an item's quality to take into account as a new factor in forming its recommendations. In Roy, Sharma, and Singh (2019) and Walek and Fojtik (2020), the authors develop film Recommender Systems. The former propose a monolithic hybrid Recommender System called `Predictory` which combines a recommender module composed of a *collaborative filtering* system with a *content-based* system and a *fuzzy expert* system to recommend suitable films. The latter focuses on the need to accurately recommend items that are relevant to the user's needs in hostile environments that involve searching through very many results. In Adiyansjah, Gunawan, and Suhartono (2019), the authors present a music Recommender System that automatically searches music libraries and suggests suitable songs to users. They use *convolutional recurrent neural networks* for feature extraction, and similarity distance to look for features that are close to each other.

Several works deal with producing accurate recommendations in the TV domain. This is ostensibly more mainstream than the radio domain since there are more TV viewers than radio listeners. In Barragáns-Martínez et al. (2010), the authors describe the design and development of `queveo.tv`, a program recommendation system that follows a hybrid approach combining *content-based* and *collaborative* filtering techniques, and which is accessible through a Web application. In Oh, Kim, Kim, and Yu (2014), the authors present `showTime`, a Recommender System whose novelty is that it determines the timing as well as the items for recommendation. While this Recommender System showed promising results, the authors note that questions remain which need to be answered before their system can be adopted to form part of a real-world application. In the domain of Internet protocol television (IPTV), the authors of Seo, Lee, and Kim (2020) present a novel method of integrating IPTV and different services to create a video-on-demand Recommender System using probabilistic matrix factorization and dealing with the data sparsity problem. The authors of Véras, Prota, Bispo, Prudêncio, and Ferraz (2015) present an interesting review of publications in this domain, identifying and discussing 282 relevant papers published from 2003 to May 2015 which reflect the proliferation of computational and network capable TVs and the large amount of TV-related content available on the Web.

With regard to broadcasting in general, in Park et al. (2017), the authors propose a real-time Recommender System for online broadcasting called `RecTime` which considers time factors and preferences simultaneously. The work applies 4D-tensor factorization, adding two further dimensions to the customary users and items – the user's current active status and the programs' time conditions – in order to estimate changing preferences. They employ 4d tensor factorization, adding two dimensions to the usual users and times so the tensor model also considers the current status of active users in terms of time condition to estimate changing preferences. In Ignatov, Nikolenko, Abaev, and Poelmans (2016), the authors develop a Recommender System for the Russian interactive radio network `FMhost`. They combine a collaborative user-based approach with personalized listened-to track tags to alleviate such problems as cold start, grey sheep, or the absence of explicit feedback in order to accurately match user and radio station profiles.

In the present work, we propose a hybrid Recommender System that first forms recommenders specialized in particular aspects of radio programs – talk vs music, musical genres, topics, and speech tone. They are then combined following a two-level strategy in which weights are assigned to the output of the recommenders that are able to switch (activate or deactivate) according to whether historical data, direct user preferences, or both are used.

The unique characteristics of the present work with regard to the use of pre-processed data, which contains intrinsic characteristics of radio programs (as it will be described below), and the absence of tagged data for validation, unlike other related works, make its comparison to other approaches really complex. As a result, such comparison can only be performed by means of the ultimate quality of the recommendations and their acceptance by users. Nevertheless, Table 1 presents both similarities and differences between our approach and the related works described herein. As shown, our approach defines an ensemble of weak predictors, each of them focused on a specific dimension, which is a major novelty of the contribution. In addition, our approach can even work when some of the dimensions are not available, so the whole system is resilient to changes on the availability of the involved data or services. Thus, the main contribution of this manuscript lies in the definition of a flexible, multidimensional ensemble of recommenders to create an accurate and dynamically adaptable Recommender System.

3. Approach

In this section, the proposed approach and its main parts will be described in detail. We shall first describe the data sources that we have been provided with. Then, we shall give an overview of our methodological approach which consists of two principal parts: the *Data Acquisition System*, that makes use of the data sources, connects to third-party service and process data, and the *Recommender System Ensemble*. The former gathers data from the data sources and connects to third-party services to help process it. The latter produces the actual program recommendations for the Assistant.

3.1. Data sources

The data that feed into the Recommender System come from three types of sources: historical data (if available), pre-processed data from radio station programs, and user preferences.

3.1.1. Historical data

The historical data is that which has been collected concerning the listeners' preferences. These preferences can be calculated either indirectly, based on the users' likes and dislikes, interests, and criteria, or directly, analyzing the number and frequency of programs a user listened to, and the period of time. Unfortunately, such data are not always fully available, and indeed in our case study there were no previous data at all. As a consequence of having no data on users' historical behavior, no constantly available radio program ratings, and no information about users' characteristics which could facilitate the identification of similar users, in many cases the success of the Recommender System depends on its ability to use appropriate data and to adequately collect the users' preferences.

3.1.2. Pre-processed data

Although historical data would have been relatively easy to collect if it were available, data with intrinsic characteristics of radio programs is quite hard to obtain. Fortunately, we had adequate useful pre-processed data resources provided by MetrikaMedia,[1] a firm that uses proprietary machine learning algorithms to analyze TV and radio broadcasts. They provided us with a pre-processed dataset (`Programs`) created from recordings of the entire broadcasting of radio stations during the previous 6 months as described by the features `Name, Day, Time` [with a total of 94 distinct radio programs]. Their machine learning algorithms allowed them to provide us with two further datasets:

[1] MetrikaMedia: https://www.metrikamedia.com/.

Table 1
Summary of related work. CB: Content-Based Recommender Systems; CF: Collaborative Filtering Recommender Systems; CRNN: Convolutional Recurrent Neural Networks; rTP: Recommendation True Positive; tTP: Timing True Positive; MF: Matrix Factorization; MAE: Mean Absolute Error; MSE: Mean Squared Error; RMSE: Root Mean Squared Error.

Research work	Topic	Date time constrains	Algorithms	Performance metrics	Hybrid approach	Multidimensional ensemble	Adaptable
(Tejeda-Lorente et al., 2014)	University Digital Library	No	CB, CF	Precision, Recall, F1	Yes	No	No
(Walek & Fojtik, 2020)	Movies	No	CB, CF	Precision, Recall, F1	Yes	No	No
(Roy et al., 2019)	Movies	No	Regression	MAE, MSE, RMSE, Percentage Error	No	No	No
(Adiyansjah et al., 2019)	Music Streaming Service	No	CRNN	Precision, Recall	No	No	No
(Barragáns-Martínez et al., 2010)	TV Programs	Yes	CB, CF	MAE	Yes	No	No
(Oh et al., 2014)	TV Shows	Yes	CF	Precision, Recall, F1	No	No	No
(Seo et al., 2020)	IPTV Video-On-Demand	No	CF (MF)	Precision, Recall, F1	No	No	No
(Park et al., 2017)	TV Shows	Yes	CB, CF	rTP, tTP	Yes	No	No
(Ignatov et al., 2016)	Radio Stations	No	CF	Precision, Recall	Yes	No	No
(Our proposal)	Radio Programs	Yes	CB, CF	# Times users follow suggestions, Precision	Yes	Yes	Yes

Table 2
Set of features describing the Programs, VoiceMusic, and Playlist datasets.

Dataset	Feature	Description	Example
Programs	ProgramName	Name of the program	40 RPM
	Day	Day of week broadcasting	Saturday
	Time	Time of broadcasting	10:30:00
VoiceMusic	Program-Day-Time	List of programs grouped by day and time	40 RPM-Saturday-10:30:00
	Total Length	Total seconds of each radio program	1800
	Foreground	Seconds of foreground music	1247
	Background	Seconds of background music	167
	Commercials	Seconds of commercials	247
Playlist	Program-Day-Time	List of programs grouped by day and time	40 RPM-Saturday-10:30:00
	Start Time	Song start time	10:32:07
	End Time	Song end time	10:36:24
	Song	Name of the song	Too Much Love Will Kill You
	Artist	Name of the artist performing the song	Queen
	Album	Album in which the song is contained	The Platinum Collection

- The `VoiceMusic` dataset which contains intrinsic information about the radio programs consisting of the time in seconds of foreground music, of people talking, of people talking with background music, and of commercials.
- The `Playlist` dataset which contains the list of songs played on each radio broadcast, giving the name of the radio program, the name of the song, and the date and time information.

Table 2 lists these three datasets' features together with examples. To the best of our knowledge, there has been no previous work using this kind of pre-processed data to design, develop, and deploy a Recommender Model.

3.1.3. User preferences

With regard to the users' preferences, we depend on specific user interfaces that are publicly available to radio audiences, and from which it is possible to get their preferences consistently. These interfaces can ask listeners directly for their tastes. The *Data Acquisition System* can then process this information according to the needs of the Recommender System in order to produce the recommendations to be fed to the radio program Assistant.

Using data from the sources described above, we set ourselves the goal of constructing the Recommender System based on four aspects or dimensions – relative voice/music percentages, musical genres, topics covered, and speech tone. The form in which the features of these dimensions are obtained will be dealt with in Section 4. It involves processing the data sources, followed by further processing through online third-party services.

3.2. Methodology

We propose the creation of an Assistant on the basis of a *hybrid recommendation strategy* that focuses on four dimensions (relative voice/music percentages, musical genres, topics covered, and speech tones). Two recommenders for each of these dimensions are constructed by taking as input Historical Data (if available) and data about Radio Programs and Users Preferences. The input is acquired and processed by a *Data Acquisition System*, which also connects to third-party services for further processing. The output of the *Data Acquisition System* feeds several independent recommenders as a first step. These deal with the aspect taken into account to produce recommendations in each dimension. As the second step, an ensemble of the recommenders is constructed to produce the Assistant's final recommendations to the users.

The links in the chain of steps that constitute this data-driven modeling pipeline are optional in all cases, and can easily be tailored to any possible specific requirements. The process and connection granularity of the *Data Acquisition System* together with the scalability of the dimensions that make up the Recommender System ensemble mean that the approach is flexible and methodologically adaptable to other application domains.

In synthesis, the approach consists of four main steps – 2 for the *Data Acquisition System* and 2 for the Recommender System ensemble (Fig. 1).

- Step (#1a) consists of acquiring the list of radio programs available and the users' preferences. This is handled by the *Data Acquisition System*. There are different pipelines for each of the

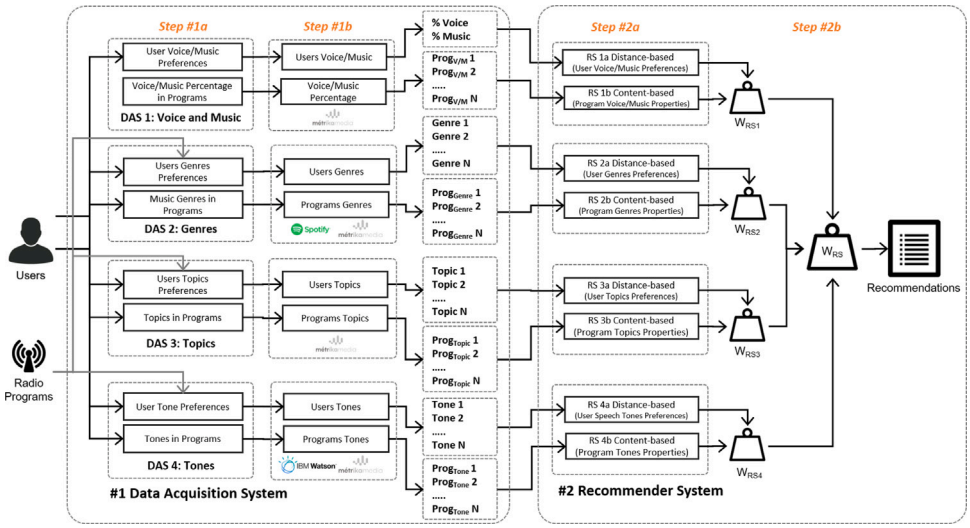

Fig. 1. Recommender System.

four dimensions. Each pipeline is handled according to what data is available, with there being two options (recommenders) which, under the most favorable conditions, can be used simultaneously:

  - Listeners directly introduce their preferences.
  - Listeners preferences are obtained from historical data of the radio programs they have listened to or their likes and dislikes.

  For instance, with regard to the music genres dimension, we would have the kind of music that a user likes, and the list of programs with the songs played in them. The approach is capable of working properly with just one of the two options or with a mix of both together.
- Step (#1b) consists of connecting to third-party services if necessary, and applying basic *Feature Engineering* techniques to facilitate as far as possible preparation of the recommenders' input data. This is also handled by the *Data Acquisition System*.
  For instance, again with regard to the music genres dimension, in this step, we would have the kind of music that a user likes and the list of genres of each program with a score of the weight of each one of them in the program. For that, data from external parties such as Spotify and MetrikaMedia is used. Additionally, transforming *Feature Engineering* processes are applied to create a dataset that could be easily treatable by machine learning algorithms.
- Step (#2a) consists of modeling the recommenders and obtaining their respective recommendations for each individual dimension and type of recommender.
- Step (#2b) consists of combining the individual recommender models in order to produce a single optimal predictive model. The output of the ensemble is more likely to perform better than that of a single model (Zhang & Ma, 2012).

Although the proposed system architecture could have been simpler without losing effectiveness, this work aims to foster the system evolution by simplifying the integration of new recommenders. Therefore, some design patterns of microservice-based architectures have been followed to facilitate the modularity, scalability, maintenance, and evolution of the recommender system. As a result, the proposed architecture provides us with the following advantages:

- The development of the different weak recommenders can be done in parallel, because they are independent of each other. Even different groups of developers may work in each recommender applying their own choice of technologies.
- Recommenders can be easily added, updated or removed in real-time without compromising the service. Hence, new dimensions could be considered, or new types of models involving new type of recommenders for each dimension could be dynamically added without affecting the rest of the components.
- Applying continuous integration (CI) and continuous delivery (CD) techniques and development processes.
- Making the system fault-tolerant to individual recommender failures, i.e. each recommender is running in its own microservice.
- System scalability to manage heavy workloads.

Once finished, the recommender needs to be deployed in order to be consumed by third parties that access it on demand. For this reason, we provide a powerful service to these parties and make use of a robust and agile architecture as the one described above, upon which to deploy the system.

In Section 4, we describe in depth the chain of steps involved in the *Data Acquisition System*: data acquisition, connection to services, and data processing techniques. In Section 5, we describe the chain of steps involved in the Recommender System, and analyze and discuss the models resulting from applying the algorithms with the aim of creating the best possible Recommender Model with which to feed the radio program Assistant. In Section 6, we examine the experiments carried out to validate the approach and its results.

4. Data acquisition system

The main task of the *Data Acquisition System* is to supply the data necessary to create the recommenders. It plays an essential role because it gathers and processes the datasets associated with each dimension and type of recommender, and applies appropriate techniques and methods to improve the datasets' features and structure, providing the recommenders the input data in an appropriate manner to infer knowledge in the form of recommendations. In our approach, the *Data Acquisition System* comprises services with specific purposes, rules, and responsibilities which focus on (a) connecting to online services, libraries, or data sources to obtain data, and (b) processing data by applying *Feature Engineering* techniques and methods to achieve better data representation.

Before the system can act in each of the dimensions, a fundamental transformation is applied to the original `Programs` dataset outlined in Table 2. This transformation is made because radio programs tend to be long, and may in some cases consist of sections with very different music and talk content depending on the day and time of the broadcast.

Table 3
Features describing the VoiceMusic dataset merged with the Programs dataset.

Feature	Type	Description and number of classes or values
Name - Day - Time	Categorical	List of all programs name grouped by day and time. 3665 classes.
Total Length	Numerical	Total seconds of each radio program. Min: 6,598. Max: 133,189.
Foreground	Numerical	Seconds of Foreground music. Min: 6,003. Max: 91,387.
Background	Numerical	Seconds of Background music. Min: 84. Max: 48,198.
Commercials	Numerical	Seconds of Commercials. Min: 0. Max: 17,027.

This may result in users liking some parts of programs but not others. For this reason, we treat each 30-minute fragment as if it were an individual radio program so that the final Assistant recommendations can be given in fractions of 30 min without having to recommend a whole program. The recommendations produced can thereby be sensitive to the wide range of topics, content, and music that any give radio program might focus on. To this end, the transformation consists of merging the Name, WeekDay, and BroadcastTime features into a single feature called Name-Day-Time.

Once we have the complete list of programs in 30-minute fragments the *Data Acquisition System* constructs an optimized dataset for each dimension's recommenders. Note that now, the number of radio programs has increased from 94 to 2133.

Additionally, the *Data Acquisition System* deals with the evolution of the systems in the sense that supporting new radio programs is not trivial. When new radio programs are broadcasted, the process of collecting data associated with them and processing and calculating the features associated with each dimension needs to be prepared to work autonomously and get all this data to feed the recommender system. This needs a high degree of automatization of the data acquisition process to update the recommenders.

The following subsections describe the processes applied for each of the four dimensions.

4.1. DAS1: Voice/Music

DAS1 applies to the Voice/Music dimension. This dimension is responsible for identifying the amounts of time that radio programs dedicate to playing music and to talk, and obtaining the user preferences in this regard.

To measure the times that radio programs dedicate to music and to talk, DAS1 uses the list of programs compiled in the Programs dataset and an additional dataset called VoiceMusic provided by Metrika-Media. This new dataset contains intrinsic information extracted from the analysis of the recordings of the programs using the proprietary machine learning algorithms. This information includes what music has been broadcast, how much time the music played in the foreground, how much time the music played in the background, and the time filled with commercials. The data is grouped by radio program. Table 3 lists the features of the VoiceMusic dataset after it has been merged with the Programs dataset.

To improve the meaningfulness of the features of the dataset listed in Table 3, a basic procedure of *Feature Engineering* processing was applied. In particular, the Total Length, Foreground, Background, and Commercials features are absolute values, and can therefore be misleading. For this reason, they were transformed into more readily interpretable features. The derived interpretable features are:

- Voice Percentage. This is the percentage of time that talk, with or without background music, is broadcast in each program. It is calculated as follows:

$$\texttt{Voice Percentage} = \frac{\texttt{Total Length} - \texttt{Foreground} - \texttt{Commercials}}{\texttt{Total Length}} \times 100 \quad (1)$$

Table 4
Features describing the VoiceMusic dataset after *Feature Engineering* processing.

Feature	Type	Number of classes or values
Name - Day - Time	Categorical	3665 classes.
Voice percentage	Numerical	Min: 0.1376, Max: 100.
Music percentage	Numerical	Min: 0, Max: 99.8624.

- Music Percentage. This is the percentage of time that foreground music is broadcast in each program. It is calculated as follows:

$$\texttt{Music Percentage} = \frac{\texttt{Foreground}}{\texttt{Total Length}} \times 100 \quad (2)$$

These features can now be taken as indicators of each program's content. Those with a high Voice Percentage are usually news, debate, or opinion programs. Those with a high Music Percentage are usually musical programs. The information in the dataset is thus more useful for the creation of a recommender model. Table 4 lists the features of the dataset after applying this *Feature Engineering* process.

To acquire data about the users' preferences in this dimension, DAS1 can receive them directly from listeners if it is possible to ask them through the Assistant, or it can analyze their historical data if available, in particular taking account of their last or most listened to programs.

4.2. DAS2: Genres

DAS2 applies to the music genre dimension. This dimension is responsible for identifying the musical style played on each radio program, as well as registering the type of music that each user prefers. To this end, DAS2 uses a dataset called Playlist which contains the list of songs played by each radio program in the last 6 months, and the Spotify® API to obtain the genres associated with each song. This API does not, however, allow one to obtain the genres associated with each song directly. Instead, it is necessary to request the artists' genres.

Once the genres associated with the artists had been found from Spotify, we realized that they were too numerous and might not be representative. As an example, the music genres associated with "The Beatles" are: {beatlesque, british invasion, classic rock, merseybeat, psychedelic rock, rock}. Of these, only two really seem representative (classic rock and rock) while the others are too particular to be associated with other artists. If all these genres were used as features, the computational cost would be very high, and there could arise issues of convergence associated with high-dimensional spaces (Hastie, Tibshirani, & Friedman, 2009), thus hindering the production of accurate recommendations.

To deal with this issue, DAS2 incorporates 2 mechanisms that drastically reduce the number of features. Spotify® provides music recommendations to its users on the basis of a list of 127 music genres.[2] First, we use the list to filter out the genres obtained from a single artist, keeping only those that appear on that list. In this way, genres such as beatlesque are removed since they are only applicable to a tiny group of artists. It has to be noted that since there is no guarantee that this number will remain constant or that the genres will be the

[2] https://developer.spotify.com/documentation/web-api/reference/browse/get-recommendations/

same over time, the list of genres will have to be updated regularly. Second, the features that do not provide representative information are removed. To this end, we set the following rule: if the classes of a genre feature coincide in more than 90% of the instances or there are no occurrences, that feature is deleted.

4.3. DAS3: Topics

DAS3 applies to the topics dimension. This dimension is responsible for identifying and classifying the topics that are covered in each radio program, as well as obtaining the user preferences in this regard.

The difficulty of labeling each program according to the topics it addresses is due to the programs' heterogeneity. To deal with this problem in a simple and practical way, we manually labeled the programs according to common industry references in consensus with radio broadcasting experts. In particular, the programs were classified according to the topics {`news`, `sport`, `entertainment`, `musical`, `education`}, assigning each of them a score from 0 to 1.

4.4. DAS4: Tones

DAS4 applies to the speech tone dimension. This dimension is responsible for identifying the speech tone of talk in each radio program, as well as obtaining the user preferences in this regard.

An important aspect to take into account in order to determine whether a radio program is suitable for a listener is the tone of the conversation, especially in programs with a large percentage of talk content. To determine the tone, DAS4 uses IBM's `Watson Tone Analyzer` API Service which applies linguistic analysis to detect emotional and language tones in written text. In particular, it classifies a text into the emotions {`anger`, `fear`, `joy`, `sadness`, `analytical`, `confident`, `tentative`}, assigning a score of from 0 to 1 to each of them. Recommendations created with emotions taken into account can be more in line with listeners' expectations.

The process of acquiring the tone is complex, and several external services are involved. To begin with, `IBM Watson` only analyzes texts in English or French, and we have recordings in Spanish. This requires transforming the recordings into text and translating that text from Spanish to English. To do this, we use the `Google Speech-to-Text` and `Google Translate` APIs, respectively.

Given the impossibility of analyzing the complete recordings of each radio program to determine the tone due to the high computational and economic costs of doing so as well as the time that would have been required, a limited number of fragments of each program were analyzed to determine the programs' speech tones. Specifically, we considered ten 10-second fragments of each program, as according to experts it is a representative sample.

5. Hybrid recommender system ensemble

5.1. Types of recommenders

We will consider four different dimensions with respect to which build our recommender systems. For each dimension d ($d = 1, \dots, 4$) we will consider the *Programs Matrix* $M^{(d)}$ of size $n \times m_d$, being n the number of programs and m_d the number of features defining the dimension d. Since recommenders are independent, we shall drop the explicit reference to the dimension d, in order to simplify the notation. Therefore, the program matrix will be defined as

$$M = \left[e_{ij}\right], \qquad i = 1, \dots, n, \quad j = 1, \dots, m.$$

The element e_{ij} will be an indication of the importance of feature j in program i (e.g. if the value of feature 1 in the program 1 is twice as much as feature 2, then $e_{11} = 2e_{12}$.)

A specific user u can be modeled thus by the submatrix U of matrix M containing the rows of M corresponding to the programs that precise user has listened to. In particular we can write

$$U = \left[e_{ij}\right], \qquad i \in I_u \subset \{1, \dots, n\}, \quad j = 1, \dots, m,$$

where I_u denotes the set of indices corresponding to the programs viewed by the user u.

5.1.1. Similarity filtering

In our case, we use a Euclidean distance as a measure for searching the radio program closer to the user preferences.

Through the Euclidean Distance, we measure the distances between the preferences directly entered by users and the characteristics of the radio programs. We use this kind of measure when users express their opinion about what they want to listen to, indicating how much music or conversation they want to hear, what type of music, what tone of conversation, or what type of content. Other previous works, as Bachrach et al. (2014) and Khoshneshin and Street (2010), have used distances to measure the similarities between feature vectors or to implement *collaborative filtering* algorithms.

To model this type of recommender, we model the users' preferences in the form of a vector $\mathbf{u}_p$,

$$\mathbf{u}_p = \left[u_{p1}, \dots, u_{pm}\right].$$

The element u_{pj} will be an indication of the importance of feature j for the user $\mathbf{u}_p$. Moreover, from the Program Matrix M defined above, one can obtain a program vector $\mathbf{M}_p$ (the pth row of M).

For the distances between the users' preferences and the program characteristics, we use the *Euclidean Distance*, widely used in other machine learning algorithms such as *k-nearest neighbors* (classification) or *k-means* (clustering). The *Euclidean Distance* is the length of a straight-line segment between a pair of samples p and q in an n-dimensional space. In our case, we calculate the *Euclidean Distance* between a program $\mathbf{M}_p$ and the preferences of a specific user $\mathbf{u}_p$ as follows:

$$d\left(\mathbf{M}_p, \mathbf{u}_p\right) = \sqrt{\sum_{i=1}^{m} \left(M_{pi} - u_{pi}\right)^2} \tag{3}$$

By sorting in ascending order the distances separating the preferences of a specific user from each of the radio programs, we obtain the most suitable suggestions for that user.

Summarizing, it is just a search for the row in the matrix M closest to the users' preferences vector u_p.

5.1.2. Content-based filtering

Content-based filtering algorithms capture the patterns of users' behavior to predict what they might like. Notice that these recommenders can only be constructed if there is data available about what programs users might like, or such information could be obtained by analyzing historical data.

From the User Matrix U defined above, one can obtain the user's vector $\mathbf{u}_p$,

$$\mathbf{u}_p = \mathbf{1}U,$$

where $\mathbf{1}$ denotes an all-ones row vector.

Normalizing this vector, we get the *user's preference vector* $\mathbf{v}_p$, defined as

$$\mathbf{v}_p = \frac{\mathbf{u}_p}{\sum_{j=1}^{m} u_{pj}},$$

which gives us the proportion of each feature present in the user's vector. Therefore, vector $\mathbf{v}_p$ can be interpreted as the "ideal" program for that user.

Table 5
Features describing the `VoiceMusic` dataset after *Feature Engineering* processing.

Dimension	Classes
RS1a (Voice/Music Dimension)	$\vec{p}$ ={mpp, vpp}, with mpp being the music percentage preference and vpp the voice percentage preference.
RS2a (Genre Dimension)	$\vec{p}$ ={acoustic, afrobeat, alt-rock, alternative, ambient, anime, black-metal, bluegrass, blues, bossanova, brazil, breakbeat, british, cantopop, chicago-house, children, chill, classical, club, comedy, country, dance, dancehall, death-metal, deep-house, detroit-techno, disco, disney, drum-and-bass, dub, dubstep, edm, electro, electronic, emo, folk, forro, french, funk, garage, german, gospel, goth, grindcore, groove, grunge, guitar, happy, hard-rock, hardcore, hardstyle, heavy-metal, hip-hop, holidays, honky-tonk, house, idm, indian, indie, indie-pop, industrial, iranian, j-dance, j-idol, j-pop, j-rock, jazz, k-pop, kids, latin, latino, malay, mandopop, metal, metal-misc, metalcore, minimal-techno, movies, mpb, new-age, new-release, opera, pagode, party, philippines-opm, piano, pop, pop-film, post-dubstep, power-pop, progressive-house, psych-rock, punk, punk-rock, r-n-b, rainy-day, reggae, reggaeton, road-trip, rock, rock-n-roll, rockabilly, romance, sad, salsa, samba, sertanejo, show-tunes, singer-songwriter, ska, sleep, songwriter, soul, soundtracks, spanish, study, summer, swedish, synth-pop, tango, techno, trance, trip-hop, turkish, work-out, world-music}.
RS3a (Topics dimension)	$\vec{p}$ ={news, sport, entertainment, musical, education}.
RS4a (Tone dimension)	$\vec{p}$ ={anger, fear, joy, sadness, analytical, confident, tentative}.

This vector $\mathbf{v}_p$ can be used to obtain a recommendation vector, $\mathbf{R}_u$, for that user by multiplying the rows of the matrix M by this preference vector. That is,

$$\mathbf{R}_u = M\mathbf{v}_p^T$$

Sorting the elements of $\mathbf{R}_u$ in **descending** order we obtain the most suitable suggestions.

Summarizing, it constructs a vector u as sum of the rows of the matrix M corresponding to programs listened by the user. Then the vector is normalized to obtain v, and $R = Mv^T$ (i.e., R is the sum of the rows of the matrix M with weights given by the normalized vector v).

5.2. Recommenders

For each of the 4 dimensions two recommenders are created, one *collaborative filtering* model and one *content-based* model, yielding in total 8 recommender models.

To create the *collaborative filtering* models, we get the users' preferences which consist of the direct measurements indicated by each user of their preferences according to the attributes present in each dimension, shown in Table 5.

With the preferences indicated by the users, the Euclidean distance is used as a metric to create recommenders, following the procedure detailed in Section 5.1.1.

To create the *content-based* models, we get the historical data about users which consist of lists of programs that the users like. Alternatively, if a user's likes or dislikes are unavailable, they can be obtained by analyzing the amount of time that the user has listened to each radio program in the past. With the list of programs as inputs, the recommenders *RS1b, RS2b, RS3b, RS4b* are created following the procedure detailed above in Section 5.1.2.

5.3. Ensemble

Ensembles (Polley & van der Laan, 2010) are constructed to improve the quality, robustness, and accuracy of simple machine learning methods. Usually, ensembles in decision systems take the learners' outputs or predictions to select the most popular among them. In some cases, when the normalized frequency ($\hat{f}$) of each output can be calculated (Dietterich, 2000), these outputs are not counted directly but their normalized frequencies are summed to get the output with greatest 'probability'.

In our approach, instead of selecting the most popular output, a dynamic weighted system processes in real-time the outputs of the simple learners in accordance with the type of prediction aimed for. This process, illustrated graphically in Step (#2b) of Fig. 1, is divided into two phases. The first produces the output (recommendations) for each dimension ($W_{RS1}, W_{RS2}, W_{RS3}, W_{RS4}$), and the second produces the final output (W_{RS}).

In Phase 1, the mean score of the two recommenders of each dimension is calculated. Both recommenders do not have to be available at the same time. If the Assistant only has information about the user preferences, the outputs of recommenders ($RS1a, RS2a, RS3a, RS4a$) are directly assigned to ($W_{RS1}, W_{RS2}, W_{RS3}, W_{RS4}$). And if the Assistant only has information about the user's likes and historical data, the outputs of recommenders ($RS1b, RS2b, RS3b, RS4b$) are directly assigned to ($W_{RS1}, W_{RS2}, W_{RS3}, W_{RS4}$). If the Assistant has information about both the user's preferences and their likes and historical data, the mean scores of the recommenders' outputs are assigned to ($W_{RS1}, W_{RS2}, W_{RS3}, W_{RS4}$).

In Phase 2, the output of each dimension produced in Phase 1 is taken to calculate the final output. This phase is dynamic in the sense that the weights are not set previously but are assigned in real-time according to the output obtained in Phase 1. This is because, in programs where most of the time is spoken content, aspects related to topics or tone will have a stronger influence, and, in programs where most of the time music is being played, the musical genres will have a stronger influence.

It is the case that the proportions of music and talk are considered in the first dimension. For this reason, the weight of each output produced in Phase 1 in each dimension is set according to the vector $\vec{p} = \{mp, vp\}$, which is the first dimension's input, where mp is the music proportion preference and vp the voice proportion preference. Thus, the weights for each dimension are set according to the following formula:

$$W_{RS} = 0.25 \cdot W_{RS1} + 0.75 \cdot \left(mp \cdot W_{RS2} + \frac{vp \cdot W_{RS3}}{2} + \frac{vp \cdot W_{RS4}}{2} \right) \quad (4)$$

The weight of Dimension 1 (voice/music) is set to 0.25. The remaining proportion has the weights adjusted according to the value of $\vec{p} = \{mp, vp\}$, distributing proportionally the weight of mp to Dimension 2 (music genres), and the weight of vp to Dimension 3 (topics) and Dimension 4 (tones), also proportionally. Although in part, the weights are set ad hoc, this was done following the guidance of experts in the field. This is clearly one of the aspects needing improvement by broadening the experiments to obtain the best weights for the most general cases.

6. Validation and results

In this section, we shall describe the process of validation that we carried out to evaluate the Recommender System's suggestions and

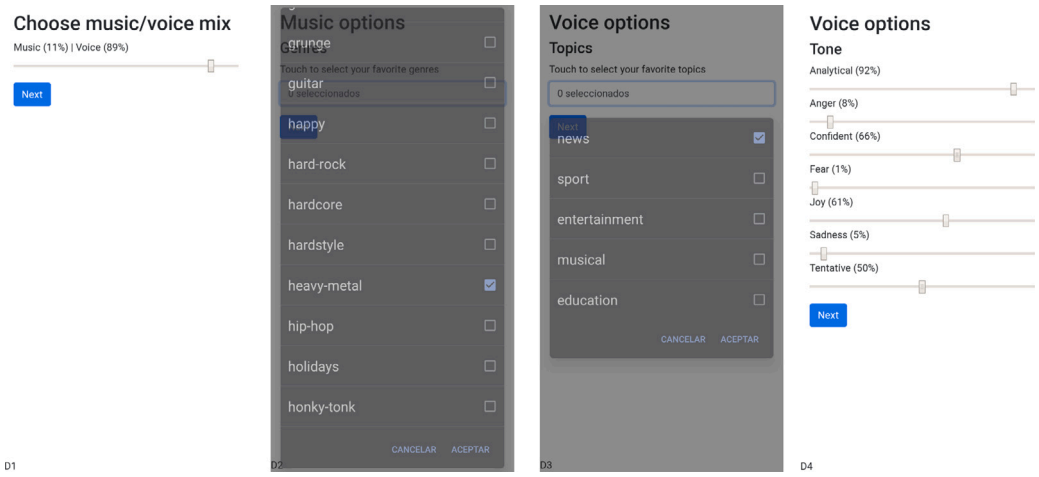

Fig. 2. Web app: User preferences.

to validate our approach. To this end, we developed an experimental Web application that captures the users' preferences and offers them recommendations. The users then provide feedback to the recommendations they received, allowing us, with that feedback, to validate the approach.

6.1. Validation application

To produce recommendations, our approach needs to know the users' preferences, and optionally have access to historical data. For validation, we need to know the users' opinions about the recommendations they were offered. For this, we developed an experimental Web Application[3] that does all the work from acquiring users' preferences and emulating historical data to providing recommendations and handling the users' feedback. This experimental application is not intended for deployment in a real environment. Nonetheless, it has a minimal set of features that allow us to test the Hybrid Recommender System's outputs by collecting feedback from potential users. The application is structured into four main steps.

The first step consists of getting the users' preferences. As has been described throughout this communication, these preferences are organized into four dimensions regarding information about such aspects as voice/music proportion, music genres, topics covered, and speech tone. Fig. 2 shows screens of the Web application where the users can input their preferences concerning each of these dimensions. As can be seen, to extract information about voice/music percentages and speech tone, users indicate their preferences using a range slider control on which they decide the percentage of music or voice they want to listen to and the relative intensity of the spoken contents according to some aspects that influence listeners' perceptions. To gather information about music genres and topics, users select their favorite music genres or the type of content they want to listen to through multi-selection drop-down menus. With the information collected in this step, it is possible to ask the $RS1a$, $RS2a$, $RS3a$, $RS4a$ *collaborative filtering recommenders* for recommendations.

The second step consists of dealing with historical data. Although this is an optional step, it provides interesting information about users' likes. As we do not have historical data at this point, we ask the users directly what programs they like. Fig. 3 shows the screens of the Web application where users can select the programs they like through a multi-selection drop-down menu that contains the radio programs broadcast in the last 6 months by a conglomerate of radio stations. With the information collected in this step, the $RS1b$, $RS2b$, $RS3b$, $RS4b$

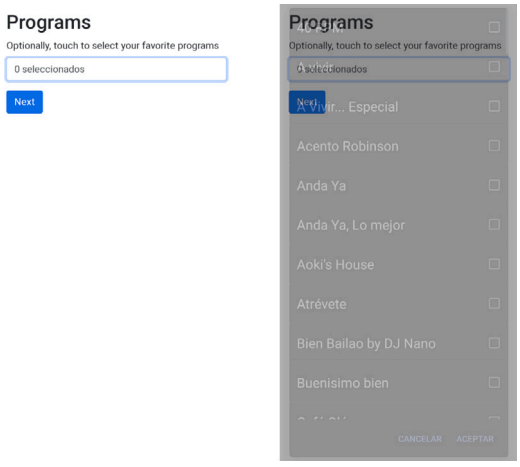

Fig. 3. Web app: Historical data.

content-based recommenders create user preferences vectors for each dimension, and use the programs' intrinsic information contained in the pre-processed data to produce recommendations in this respect. Note that users not selecting any program from the multi-selection drop-down menu are directly considered anonymous users.

The third step, whose screens as shown in Fig. 4, basically shows the users a summary of the data they have entered, and processes that data to produce recommendations. Internally, it connects with the 8 recommenders, obtains their outputs, and dynamically assigns the weights in the ensemble to generate a list of recommendations. Alternatively, those weights can also be manually assigned by the users to better aligned the relevance of each dimension to their own preferences. In this case, users are presented with an additional screen to select the relative relevance of each dimension by means of a slider control (central image in Fig. 4).

Finally, the fourth step presents the users with the recommendations in descending order. At the top of the list are the programs that are most akin to the data entered. Although in this experimental research application, the recommendations are not filtered, it is advisable to do so in accordance with the day of the week and the time-slots selected by the users, with the default being the current day and time. After the recommendations have been presented, the application asks the user for feedback about whether those recommendations are consistent with their likes, and therefore that they have listened to the programs suggested. Specifically, the user can select whether they have

[3] Validation Application URL: https://i3lab.unex.es/radiorecs/.

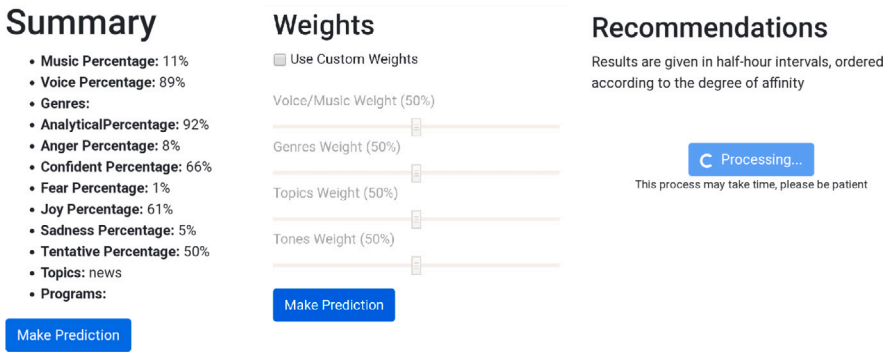

Fig. 4. Web app: Summary-Weights-Processing.

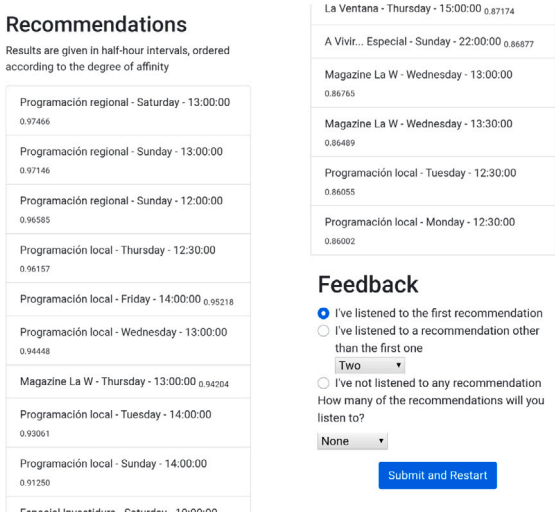

Fig. 5. Web app: Recommendations and Feedback.

listened to the first recommendation, to any other recommendation, or to none of the suggested recommendations. The recommendations and the feedback request are shown in Fig. 5. It should be noted that the recommendations are given in half-hour time slots and for different days. For this reason, the same program can be suggested more than once, giving preference to the day and time that is optimal for the user.

6.2. Results

In this subsection, we use the users' feedback in order to validate the Recommender System's effectiveness. A common strategy applied in evaluating Recommender Systems is to attempt to measure the quality of the suggestions by (a) calculating the number of recommendations produced that are relevant (the *Precision*), and (b) calculating the number of relevant programs that are recommended (the *Recall*). As we do not have supervised data, we examine the adequacy of the recommendations in accordance with the users' perceptions as a measure of the Recommender System's acceptance.

An extensively used approach is to calculate the proportion of radio programs suggested by the Recommender System that users actually listen to. This metric presents some limitations, however, that we wish to avoid. A particularity of this metric is that if a user listens to 1 out of 20 recommendations, the calculated value is 0.05, and if they listen to 19 out of 20 recommendations, the calculated value is 0.95. We want to maximize the likelihood of users following the Recommender System's suggestions even if they listen to just one radio program. In such a context, this metric could be misleading because users who follow most of the suggestions offered can easily lead to this metric having a high value that would not serve to validate our proposal. The situation is further aggravated due to our providing recommendations in half-hour time slots and for different days. Thus, it is relatively frequent that the recommendations include several time slots of different days for the same program.

For these reasons, to validate our proposal we want to calculate the number of times that users follow the Recommender System's suggestions, regardless of the score that the Recommender System gives to the suggestion followed. Also, as a secondary objective, we want to know whether the first suggestion, which is the most suitable according to the Recommender System, is generally considered. Thus, the feedback question put to users at the end of the experimental application offers the following possible responses: (1) I have listened to the first recommendation; (2) I have listened to a recommendation other than the first one; and (3) I have not listened to any recommendation. Moreover, to get a broader evaluation, two additional questions are presented. Firstly, we want to know which recommendation (position) has been followed, in case the first one was dismissed. Given that position, the deviation from the top ranked recommendation can be measured. Secondly, we want to know how many recommendations are useful from the user's point of view, so we can calculate the overall *precision*.

In order to evaluate our proposal, we have carried out three different experiments, i.e. three different setups. For each experiment, a total of 200 executions of the Recommender System with their corresponding users' feedback was acquired by the experimental Web application. The characteristics of each experiment are the following:

1. **Experiment 1**. Recommendations are calculated using ad-hoc weights according to experts' opinions. Both anonymous and recurrent users are considered.
2. **Experiment 2**. Recommendations are calculated according to users' preferences, which determine the weights assigned to each dimension (relative talk/music percentages, music genres, topics covered, and speech tone). Both anonymous and recurrent users are considered.
3. **Experiment 3**. Recommendations are calculated using ad-hoc weights according to experts' opinions. Only anonymous users are considered, i.e. only user preferences recommenders are applied.

A summary of the results of the experiments is shown in Table 6. Concretely, for every experiment, the percentage of users and the total number of users (in brackets) are presented for every item considered. 15 recommendations are offered from a total of 2133 possible recommendations. For example, in the first row the number of users following the top most recommendation is shown.

As a first conclusion, both experiments taking into account historical data (experiments 1 and 2) yields similar results, e.g. they return 82.5% and 81.5% respectively for the ratio of users following any of the provided recommendations. Consequently, it would be hard to argue

Table 6
Experiments results comparison.

Metric	Experiment 1	Experiment 2	Experiment 3
Top recommendation	0.265 (53)	0.210 (42)	0.195 (39)
Other recommendation	0.560 (112)	0.605 (121)	0.485 (97)
Any recommendation (agg.)	0.825 (165)	0.815 (163)	0.680 (136)
No recommendations	0.175 (35)	0.185 (37)	0.320 (64)
Average distance	6.217	7.911	8.435

that one is better than the other. The main difference is that the first experiment gets slightly better results regarding the number of users that follow the first recommendation and, the second one gets slightly better results regarding the number of users that follow other recommendations than the first.

As aforementioned, we define a *distance* metric as a way of measuring the deviation of the user selected recommendation from the top ranked one. It is calculated as the average position of the first recommendation followed by the users when it is not the top one.

Looking at the distance performance metric in the same table and considering that 20 recommendations are given, one can see that, usually, the first recommendation followed by users in the first experiment is in the first third of the recommendations provided by the recommender system. It indicates that the recommender system has a convenient means of ordering, and recommendations with higher scores are most likely to be followed by users. In conclusion, the experiment 1 creates a recommender providing the best ranking of recommendations compared to the other ones. This is indeed another advantage of the recommender created for the first experiment.

As regards the performance of the third experiment, it can be seen that its result is poorer than the first and second experiments. However, it indicates generally satisfactory results bearing in mind that it just makes use of four recommenders and has no information regarding the programs commonly listened to by the users.

Finally, although we believe these are promising results, some efforts are still pending to reduce the bias introduced by the user input. For example, a better way to obtain the users' feedback would be to incorporate the Recommender System into a real application to monitor whether users actually follow the recommendations, so no questions need to be asked. Moreover, it would be better to know how long users follow the recommendations, *i.e.*, the proportion of time that the user listens to a suggested radio program relative to its total time as a measure of the quality of the recommendations.

6.3. Comparison with related works

It is hard to compare the present results with those of related work due to the heterogeneity of the metrics used as we described in Table 1. Most of the related studies use *Precision* and *Recall* metrics (Barragáns-Martínez et al., 2010; Ignatov et al., 2016; Oh et al., 2014; Park et al., 2017), some include *F1-Score* (Tejeda-Lorente et al., 2014; Walek & Fojtik, 2020), and one uses *MAE* and *RMSE* (Seo et al., 2020). Even one of the works use its own metric according to accuracy and timing (Park et al., 2017). Even those with the same metrics are difficult to compare because their domains and objectives are quite different, such as the quality of the recommendations or their timing.

For the reasons described above regarding the heterogeneity of the works, we have to rely on the results that individual experts in the field assess as good and useful, a goal that essentially has been achieved in the validation of our particular case study where more than 4 out of 5 users follow the Recommender System's suggestions.

Nevertheless, Table 7 presents the characteristics and results of the closest related works in order to compare them to our approach. We use the *Precision* metric to compare, which can be defined as follows.

Table 7
Comparison of recommender system metrics.

Research work	Precision
(Tejeda-Lorente et al., 2014)	0.6923
(Walek & Fojtik, 2020)	0.8100
(Adiyansjah et al., 2019)	0.7120
(Oh et al., 2014)	0.1580
(Seo et al., 2020)	0.5274
(Ignatov et al., 2016)	≈ 0.3000
(Our proposal)	0.5785

- **Precision.** Ability of the recommender system to provide useful recommendations. Given a list of k recommendations, the precision is the percentage of them that are useful for the user.

$$Precision = \frac{Number\ of\ Recommendation\ that\ user\ likes}{List\ of\ recommendations\ provided} \quad (5)$$

It would be interesting using the *recall* metric as well but we could not perform this comparison because it is impossible for us to calculate this metric since we do not have our radio programs rated by the users at this stage, so we cannot identify the relevant content on this aspect. We will indeed make use of this metric when possible.

It must be noted that these metrics are directly obtained from the original papers, and some aspects have to be taken into consideration. In Tejeda-Lorente et al. (2014), the metrics values are the mean of the recommender system results for 30 users. In Walek and Fojtik (2020), the metrics are the mean of the recommender system results for 17 users extracted. In Adiyansjah et al. (2019) the metrics values are the mean of the recommender system results for 7 music genres. Oh et al. (2014) provide global results. In Seo et al. (2020), the metrics are calculated over the top 10 recommendations of the recommender system. In Ignatov et al. (2016), the metrics are calculated over the top 100 recommendations of the recommender system. In our recommender system, the metrics are calculated over the top 20 recommendations of the recommender system. If the number of recommendations provided is not specified, it is assumed to be 1. If the atomicity of the metrics is not specified, it is assumed to be a global result.

To perform the comparison, we use the results obtained by our first experiment, which got the best results. To calculate the *precision* we have considered the average value of the *precision* calculated for each of the 200 output of the recommender system by dividing the number of recommendations that the users are willing to listen to by the number of recommendations, which is 20. As a result, the *precision* is 0.5785.

In Table 7, we perform a comparison with the related works discussed. As one can see, there is a great difference between the highest and the lower value. Walek and Fojtik (2020) yields the highest *precision* (0.81). The *precision* metric is greatly influenced by the number of recommendations provided by the recommendation system and the total number of possible suggestions. In this work, a number of 25 movies are suggested from a list of 9724. Follow-up, the second-highest *precision* can be found in the work of Adiyansjah et al. (2019) (0.7120), which suggests 5 songs from a list that has an undetermined number of elements. Initially, there are 25.000 songs but after processing and cleaning data they do not provide the final number of elements considered. Besides, they provide recommendations grouped by genre, which significantly reduces the number of elements to suggest. The

last discussed work with better *precision* than our approach is the work of Tejeda-Lorente et al. (2014) (0.6923), which produces recommendations over 200 research resources related to different areas. The analyzed work with a lower value of *precision* is *tvrs2*. They justifies those results because their model tries a high number of recommendations, so most of them would eventually fail. They do not specify the number of recommendations presented to users.

In the light of the results of the experiments carried out and the results of the discussed work, one can see how complex this comparison could become because the important differences in the number of suggestions provided, the whole list of possible suggestions, the algorithms used to build the models or the available data in each case, among others. Given this diversity, we could conceivably conclude that the results achieved by our approach are satisfactory because they are aligned with the results achieved by the related work discussed.

7. Conclusions and future work

This communication has focused on the creation of a hybrid Recommender System (using *content-based* and *collaborative filtering* recommenders) for radio programs using a dynamic ensemble of these learners depending on whether historical data, direct user preferences, or both exist. The two-phase ensemble assigns dynamically the weights that produce the final recommendations and, additionally, allows radio listeners to assign the weights according to their preferences if they want to. Also, experimentation was carried out to validate the results of the approach.

Thus, the main contribution of this manuscript lies in the definition of a flexible, hybrid multidimensional ensemble of recommenders to create an accurate Recommender System, having its own customizations, can be generalized, which is supported by a software architecture that facilitates the adaptation and evolution over time of the recommender. Thereby, the work has answered the research question by performing the following tasks:

(a) We have designed and validated an approach to create a reliable radio program Recommender System using pre-processed data and users' preferences, and that works with or without historical data.
(b) We have deployed a *Data Acquisition System* that obtains data from data sources, captures user preferences, and connects to third-party services. It then implements *Feature Engineering* processes to get optimized datasets for sending as inputs to recommenders.
(c) We have made use of Pre-processed Data including automatic recognition of the songs and artists being played, the time dedicated to talk and/or music, and the tone of the conversation. Aspects which, to the best of our knowledge, have not previously been used to create a Recommender Model of this kind.
(d) We have designed a hybrid Recommender System that makes use of *content-based* and *collaborative filtering* recommender models to produce specific recommendations for different aspects of radio programs such as voice/music percentage, musical genre, topics covered, and speech tone.
(e) We have created an ensemble of the basic recommenders that is able to detect which recommenders to use depending on the data available, and to dynamically assign weights to the recommenders' outputs in accordance with the user preferences to produce the most suitable recommendations by overcoming any overfitting, bias, or imbalance issues that may arise, and to transfer these recommendations to the Assistant for them to be offered to radio listeners.
(f) We have designed and developed an experimental Web application that allows us to obtain the data needed to evaluate the quality of the recommendations and to validate our approach.
(g) We have created an ensemble of recommenders flexible, in the sense that it is easy to add or delete new type of recommenders or dimensions; fault-tolerant, because it keeps working despite the availability of some recommenders may be restricted; which evolves over time, providing adaptation to new users and radio programs.

As future work, we plan to follow these practices:

(a) Take into account the activity that the user is doing by monitoring their smartphone and wearable devices. We would thus know whether the user is driving, exercising, sitting, or relaxing, and could use this contextual information as new dimensions to produce specific recommendations according to the user's activity.
(b) Gathering the necessary resources that would allow us to carry out further experiments to validate the proposal such as: (a) Get the first production results after the deployment of the Recommender Model to incorporate new validation metrics such as *Recall*, and *F1-Score*; and (b) Better determine the weights of the type of recommenders and dimensions using experimental methods, or assign them in a fully dynamical fashion.
(c) Further study alternative forms of feature representation to analyze whether they may improve the final performance of the recommenders.
(d) Implement our own libraries to detect the tone of the voice to avoid bias or possible misunderstandings that Spanish to English translation and Voice to Text transformation may introduce. Also, experiments with new methods and libraries that already deal with this problem obtain the voice's tone directly from recordings without further transformation that may include inaccuracies.
(e) At the moment, we just know whether a user have listened to a program. In the future we plan to let users rate programs, so we can use those ratings to know how much a user likes or dislikes a program.

CRediT authorship contribution statement

Antonio Jesús Fernández-García: Conceptualization, Supervision, Software, Investigation, Writing - Original Draft, Writing – review & editing. **Roberto Rodriguez-Echeverria:** Conceptualization, Supervision, Data curation, Investigation, Writing – original draft, Writing – review & editing, Project administration, Funding acquisition. **Juan Carlos Preciado:** Methodology, Investigation, Software, Investigation, Writing – original draft, Project administration, Funding acquisition. **Jorge Perianez:** Methodology, Writing – review & editing, Investigation, Visualization, Data curation, Validation. **Juan D. Gutiérrez:** Methodology, Validation, Software, Writing – review & editing, Investigation, Visualization, Data curation, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was carried out with the support of (i) Ministerio de Ciencia, Innovación *y* Universidades (MCIU), Spain, Agencia Estatal de Investigación (AEI), Spain, and European Regional Development Fund (ERDF): RTI2018-098652-B-I00 project, and (ii) European Regional Development Fund (ERDF) and Junta Extremadura: projects IB16055, IB18034, GR18112.

References

Adiyansjah, Gunawan, A. A. S., & Suhartono, D. (2019). Music recommender system based on genre using convolutional recurrent neural networks. *Procedia Computer Science*, *157*, 99–109. http://dx.doi.org/10.1016/j.procs.2019.08.146, URL: https://www.sciencedirect.com/science/article/pii/S1877050919310646. The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749.

Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., et al. (2014). Speeding up the Xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys '14*, *Proceedings of the 8th ACM conference on recommender systems* (pp. 257–264). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/2645710.2645741.

Barragáns-Martínez, A. B., Costa-Montenegro, E., Burguillo, J. C., Rey-López, M., Mikic-Fonte, F. A., & Peleteiro, A. (2010). A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, *180*(22), 4290–4311.

Bunnell, L., Osei-Bryson, K.-M., & Yoon, V. Y. (2020). FinPathlight: Framework for an multiagent recommender system designed to increase consumer financial capability. *Decision Support Systems*, *134*, Article 113306.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*, 331–370. http://dx.doi.org/10.1023/A:1021240730564.

Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*, 1487–1524. http://dx.doi.org/10.3233/IDA-163209.

Chamoso, P., Rivas, A., Rodríguez, S., & Bajo, J. (2018). Relationship recommender system in a business and employment-oriented social network. *Information Sciences*, *433–434*, 204–220.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *MCS '00*, *Proceedings of the first international workshop on multiple classifier systems* (pp. 1–15). Berlin, Heidelberg: Springer-Verlag.

Dong, M., Zeng, X., Koehl, L., & Zhang, J. (2020). An interactive knowledge-based recommender system for fashion product design in the big data environment. *Information Sciences*, *540*, 469–488.

Fernández-García, A. J., Iribarne, L., Corral, A., Criado, J., & Wang, J. Z. (2019). A recommender system for component-based applications using machine learning techniques. *Knowledge-Based Systems*, *164*, 68–84.

Fernández-García, A. J., Rodriguez-Echeverria, R., Preciado, J. C., Conejero, J. M., & Sánchez-Figueroa, F. (2020). Creating a recommender system to support higher education students in the subject enrollment decision. *IEEE Access*, 1.

Gomez-Uribe, C. A., & Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*, *6*(4).

Guo, J., Deng, J., Ran, X., Wang, Y., & Jin, H. (2021). An efficient and accurate recommendation strategy using degree classification criteria for item-based collaborative filtering. *Expert Systems with Applications*, *164*, Article 113756. http://dx.doi.org/10.1016/j.eswa.2020.113756, URL: https://www.sciencedirect.com/science/article/pii/S0957417420305807.

Hasanzadeh, N., & Forghani, Y. (2021). Improving the test time of M-distance based recommendation system. *Journal of the Institution of Engineers (India): Series B*, http://dx.doi.org/10.1007/s40031-021-00626-1.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction. *International Statistical Review*, *77*(3), 482–482.

Herce-Zelaya, J., Porcel, C., Bernabé-Moreno, J., Tejeda-Lorente, A., & Herrera-Viedma, E. (2020). New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests. *Information Sciences*, *536*, 156–170.

Hernando, A., Bobadilla, J., Ortega, F., & Gutiérrez, A. (2017). A probabilistic model for recommending to new cold-start non-registered users. *Information Sciences*, *376*, 216–232.

Ignatov, D. I., Nikolenko, S. I., Abaev, T., & Poelmans, J. (2016). Online recommender system for radio station hosting based on information fusion and adaptive tag-aware profiling. *Expert Systems with Applications*, *55*, 546–558.

Khoshneshin, M., & Street, W. N. (2010). Collaborative filtering via euclidean embedding. In *RecSys '10*, *Proceedings of the fourth ACM conference on recommender systems* (pp. 87–94). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/1864708.1864728.

Kiran, R., Kumar, P., & Bhasker, B. (2020). DNNRec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, *144*, Article 113054. http://dx.doi.org/10.1016/j.eswa.2019.113054, URL: https://www.sciencedirect.com/science/article/pii/S0957417419307717.

Oh, J., Kim, S., Kim, J., & Yu, H. (2014). When to recommend: A new issue on TV show recommendation. *Information Sciences*, *280*, 261–274.

Park, Y., Oh, J., & Yu, H. (2017). RecTime: Real-time recommender system for online broadcasting. *Information Sciences*, *409–410*, 1–16.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: methods and strategies of web personalization* (pp. 325–341). Berlin, Heidelberg: Springer Berlin Heidelberg.

Polley, E. C., & van der Laan, M. J. (2010). Super learner in prediction.

Rodriguez-Marin, P., Duque-Mendez, N., Ovalle-Carranza, D., & Martinez-Vargas, J. (2020). Personalized hybrid educational recommender system using matrix factorization with user and item information. http://dx.doi.org/10.20944/preprints202008.0700.v1.

Roy, S., Sharma, M., & Singh, S. K. (2019). Movie recommendation system using semi-supervised learning. In *2019 Global conference for advancement in technology* (pp. 1–5).

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web: methods and strategies of web personalization* (pp. 291–324). Berlin, Heidelberg: Springer Berlin Heidelberg.

Seo, Y.-D., Lee, E., & Kim, Y.-G. (2020). Video on demand recommender system for internet protocol television service based on explicit information fusion. *Expert Systems with Applications*, *143*, Article 113045.

Smith, B., & Linden, G. (2017). Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, *21*(3), 12–18.

Tang, X., Qian, T., & You, Z. (2020). Generating behavior features for cold-start spam review detection with adversarial learning. *Information Sciences*, *526*, 274–288.

Tejeda-Lorente, A., Porcel, C., Peis, E., Sanz, R., & Herrera-Viedma, E. (2014). A quality based recommender system to disseminate information in a university digital library. *Information Sciences*, *261*, 52–69.

Tran, T., Liu, X., Lee, K., & Kong, X. (2019). Signed distance-based deep memory recommender. In *WWW '19*, *The world wide web conference* (pp. 1841–1852). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3308558.3313460.

Véras, D., Prota, T., Bispo, A., Prudêncio, R., & Ferraz, C. (2015). A literature review of recommender systems in the television domain. *Expert Systems with Applications*, *42*(22), 9046–9076.

Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems with Applications*, *158*, Article 113452.

Wang, D., Liang, Y., Xu, D., Feng, X., & Guan, R. (2018). A content-based recommender system for computer science publications. *Knowledge-Based Systems*, *157*, 1–9.

Wang, W., Zhang, G., & Lu, J. (2015). Collaborative filtering with entropy-driven user similarity in recommender systems. *International Journal of Intelligent Systems*, *30*(8), 854–870. http://dx.doi.org/10.1002/int.21735, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/int.21735.

Xia, Y., Chen, K., & Yang, Y. (2020). Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*.

Zhang, C., & Ma, Y. (2012). *Ensemble machine learning: methods and applications* (pp. 1–329).