Event Recommendation Engine

Jorge Oliveira
jorge.de.oliveira@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2016

Abstract

Nowadays, with the growing ammount of information related to existing events on the Internet, it has become increasingly difficult for users to find those that best fit their preferences and personal tastes. In this regard, recommender systems came to help users in this task by reducing the overload of information felt by the users, by recommending events that users may like. However, differently from the classical recommender problem, which envolves items such as movies or books, events suffer from the so called *new item cold-start problem*. Since events often take place in the future, there's an absence of *feedback* from the users, as well as a lack of records about the user's attendance in the events. In such situations, it's necessary to consider, not only information about the events and the users, but also contextual information. Having in mind the presented problem, an event recommendation engine was implemented, which has, for primary objective, to predict the events in which the users may be, eventualy, interested. The recommendation is made by considering information regarding previously attended events by the users, demographic information about them, and what events theyve seen and interacted with, e.g., through clicks in an application. In order to get the best possible recommendation performance, several *features* were created, being the event recommendation made with a *Random Forest* classifier. Several tests carried out have certified the good efficacy and performance of the recommender solution developed.
Keywords: Recommender Systems, Event Recommender Systems, Context, Personalization.

1. Introduction

Recommender Systems are a particularly important tool in the actual panorama of the Internet, both for the users and for the enterprises that makes use of them. In fact, huge ammounts of information are posted daily on the Internet, making difficult for users to find items that best fullfil their preferences and personal tastes. In this sense, recommender systems help the users in this task, by reducing the information overload, recommending items that they will, with high probability, like, e.g., books and movies. On the other hand, by recommending items of possible interest of the users, these systems are also a profit source for the commerce enterprises that own them, offering, in addition, an addicional and personalized service to the users that allows to conquer their trust and loyalty [8].

However, the recommendation task does not apply solely to items such as books or movies. Indeed, huge ammounts of information regarding the most varied types of events, e.g., concerts, music festivals, scientific conferences, etc., are also posted daily on the Internet. The resulting information overload makes the task of finding events that best addresses the user's preferences difficult. In that sense, the use of recommender systems supply a precious help, recommending to the users events that can fullfil their personal preferences and tastes.

The task of recommending events is, however, a different problem from the classical recommender problem. In the classical recommender problems, the items to be recommended were previously consumed and rated by many users, except those added recently to the system. This problem is called the *new item cold-start problem*. On the other hand, in the event recommendation problem, the items to be recommended (i.e., events) are also called *one-and-only items* and have, tipically, a short lifetime and allways take place in the future [16]. Consequently, users cannot participate in events or rate them before their occurence. Thus, the event recommendation task has to deal constantly with the *new item cold-start problem*, making the event recommender task more complicated that the classical recommendation task. However, users can express their intention of attending or not events, thus providing useful information in order to mitigate this problem [16].

In order to provide to the users accurate recommendations, event recommender systems take

into account a wide range of information, which includes information regarding the users (e.g., birthdate, gender, location), the events (e.g., the event's type, the event's description, the event's location), and context-aware information (e.g., distance between a user and an event, co-attendance of users in events, the time preferences of the users in terms of event attendance, etc.).

Taking into account the event recommender problem described above, an event recommender engine was implemented, which has, as primary objective, to predict the events in which the users may be, eventualy, interested. The recommendation is made by considering information regarding previously attended events by the users, demographic information about them, what events they have seen and interacted with (e.g., through clicks in an application), etc. This information was modeled through a set of features. For each user considered, the system returns a list of recommended events, ordered by decreasing order of relevance. These recommendation lists are generated through the application of the implemented features in a Random Forest classifier. In order to increase even more the recommendation efectiveness and performance, a feature selection procedure based on the Sequential Forward Selection (SFS) algorithm was applied.

The evaluation of the developed event recommender engine was made through the use of the dataset from *Kaggle's Event Recommendation Engine Challenge*. The obtained results were evaluated through the MAP@200 evaluation metric. The following main results were obtained:

- The application of all the implemented features in the event recommender task results in a value of MAP@200 = 0.7059;
- The application of a feature selection procedure based on the SFS algorithm resulted in the selection of a small subset of relevant features from the original feature set. These ones were applied in the event recommender task, resulting in a value of MAP@200 = 0.7475.

Based on the results above, the following main conclusions were obtained:

- The developed event recommender engine presents better recommendation performance and effectiveness against the solutions submited by the *Kaggle's Event Recommendation Engine Challenge* participants;
- The best recommendation results can be obtained through the use of context-aware information;
- By selecting the relevant features for an event recommender problem, the use of feature selection procedures allows the obtention of even better event recommendation results.

The remaining of this paper is organized as follows: in the Section 2 the background is presented, which includes some basic concepts and the presentation and discussion of related work regarding the event recommender problem. In the Section 3 the implemented event recommender engine is explained in detail. The Section 4 describes the dataset used and the evaluation metric considered. The obtained results are also presented and discussed in this section. Finally, the main conclusions and some future work in the context on the implemented event recommender engine are presented in the Section 5.

2. Background

This section aims to present the basic concepts related with the recommender systems which are applicable to the event recommender task. At the same time, the related work in the area of the event recommendation problem will be presented and discussed.

2.1. Basic Concepts

Some basic concepts related to recommender systems are presented below.

2.1.1 Collaborative Filtering

Collaborative Filtering (CF) is a filtering method in which the recommendations, for each user, are made taking into account information (ratings for a given set of items such as movies and books) provided by other users who have a high similarity with the target user of recommendation [3]. The key ideia is that, if two users had similar preferences in the past, they will have similar preferences in the future [8]. We can distinguish between two main types of collaborative filtering:

- User-based Collaborative Filtering, wherein the rating prediction of a given item by a given user is done through the aggregation of the ratings assigned to the same item by users similar to the user target of recommendation [22];
- Item-based Collaborative Filtering, wherein the items are recommended based on information regarding other items previously rated by the user. in this type of CF, the items recommended to a given user are rated thorugh the aggregation of the similarities between each candidate item and the items the user rated [22].

In CF, we have a users versus items matrix, wherein, for each user, we have the ratings that it has assigned to a given set of items. Based on this matrix, we can, for example, determine the similarity between two users through the Pearson Correlation:

$$\text{sim}(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \overline{r}_a)(r_{b,p} - \overline{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \overline{r}_a)^2}\sqrt{\sum_{p \in P}(r_{b,p} - \overline{r}_b)^2}} \quad (1)$$

In Equation 1, a and b are users, $r_{a,p}$ is the rating of the user a for the item p, P is the set of items rated both for a and b, and $\overline{r_a}$ e $\overline{r_b}$ correspond to the averages of the ratings of users a and b.

The rating prediction for an item not rated yet by a given user is given by:

$$\text{Pred}(a,p) = \overline{r_a} + \frac{\sum_{b \in N} \text{sim}(a,b)(r_{b,p} - \overline{r}_b)}{\sum_{b \in N} \text{sim}(a,b)} \quad (2)$$

The most used algorithm in this type of approach is based on the search for the k Nearest Neighbors (kNN) [3].

2.1.2 Content-based Filtering

Content-based Filtering (CBF) is a filtering method wherein the recommendations for each user are done based on the similarity between items that the user has consumed and rated positively in the past and new items, not yet seen by the user [3]. The key ideia of this method consists in the recommendation of new items to a given user, based on his past choices (e.g., if a given user read and liked crime fiction books in the past, the system will try to recommend to him other crime fiction books not yet read by the user). In this approach, the content of the items is analyzed (e.g., the items description, keywords, genre, etc.), being the result of that analysis used in the estabilishment of similarities between items [3]. A simple way of determining the similarity between an item not yet seen by the user and his profile is obtained through the use of the Dice Coefficient [8], in which the similarity calculation is done based on keyword overlapping:

$$\text{sim}(b_i, b_j) = \frac{2 * \mid \text{keywords}(b_i) \cap \text{keywords}(b_j) \mid}{\mid \text{keywords}(b_i) \mid + \mid \text{keywords}(b_j) \mid} \quad (3)$$

However, representation in simple keywords presents some problems such as the fact that not all the words extracted from the items contents have equal importance. Thus, the standard measure used in this type of recommendation methodology is the TF-IDF measure (Term Frequency-Inverse Document Frequency), in which documents (in this case, the items contents) are encoded as a vector of weighted terms:

$$\text{TF}(i,j) = \frac{\text{freq}(i,j)}{\text{maxOthers}(i,j)} \quad (4)$$

$$\text{IDF}(i) = \log\left(\frac{N}{n(i)}\right) \quad (5)$$

$$\text{TF} - \text{IDF}(i,j) = \text{TF}(i,j) \times \text{IDF}(i) \quad (6)$$

Let $\overrightarrow{\theta}_{d_x}$ be the vector of TF-IDF weights for a document d_x. Thus, the similarity between any two documents d_1 and d_2 can be determined through its cosine similarity:

$$\text{sim}(d_1, d_2) = \cos(\overrightarrow{\theta}_{d_1}, \overrightarrow{\theta}_{d_2}) = \frac{\overrightarrow{\theta}_{d_1} . \overrightarrow{\theta}_{d_2}}{\| \overrightarrow{\theta}_{d_1} \| \| \overrightarrow{\theta}_{d_2} \|} \quad (7)$$

2.1.3 Supervised Classification

Supervised Classification consists in a Machine Learning (ML) technique in which the main goal consists in the construction of a concise class/label distribution model through the use of a set of features [12]. In order to generate a supervised classification model, a set of training examples are supplied. For each training example, the feature values are known, as well as their respective labels. The resultant supervised classification model is then applied to a set of test examples. For each test example, the feature values are known, but their respective labels are unknown. The supervised classification model generated previously should be capable to predict the right class/label for each test example supplied. This process is also explained in Figure 1 [12].

2.2. Related Work

There is several works in the literature which focus on the event recommendation problem. The works presented and discussed in this paper can be divided into three main groups:

1. Simple approaches to the event recommendation problem;
2. Hybrid approaches to the event recommendation problem;
3. Comparison of approaches to the event recommendation problem;

Besides these ones, there are also other works which provide an overview of recommender systems [8, 22, 3]. This paper will focus only on the event recommendation problem.

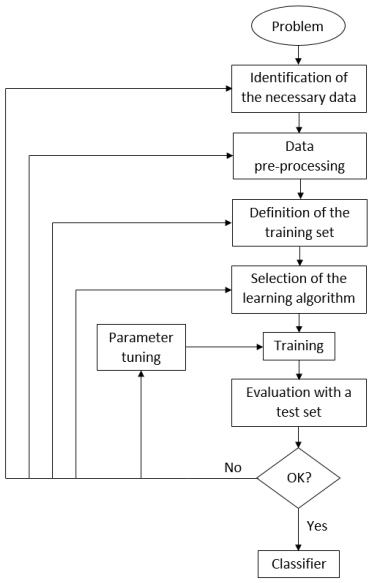

Figure 1: Diagram of the supervised classification process [12].

2.2.1 Simple approaches to the Event Recommendation Problem

There are in the literature solutions for the event recommendation problem that make use of approaches based on a single recommendation technique. The work of Kang et al. [9] describes a real-time event recommender solution, called EVENTERA[1], wherein the recommendation is done through the use of a CF algorithm (similar to the one presented in Section 2.1.1) over the event's keyword lists. The presented solution also allows the aggregation of massive quantities of online media from heterogeneous channels, the summarization of these into events, the discovery of meaningful associations by linking the events, and the generation of a sequence map of the events in order to provide an image on how the events interact with each other over time. The experimental results obtained show that EVENTERA is able to accomplish the above purposes. However, EVENTERA does not use any kind of context-aware information, which can lead the system to present a lower recommendation performance.

2.2.2 Hybrid approaches to the Event Recommendation Problem

An hybrid approach consists in the combination of several input data and/or several individual recommender approaches. The resultant recommender approach allows the combination of the advantages of the various individual recommender approaches used. The use of different input data is also another advantage. There are three tipes of hybrid approaches:

1. **Monolithic hybridization**, which consists in the use of a single recommender component which combines the different recommendation characteristics and strategies of the various employed recommender approaches;
2. **Parallel hybridization**, wherein the final recommendation score is obtained through the recommendation scores from various recommender approaches, through a linear combination or a voting scheme;
3. **Pipelining hybridization**, wherein the final recommendation score is obtained through the use of several recommender approaches mounted in series. Each one performs some kind of pre-processing on the input data it received, being the result of the pre-processing passed to the next recommender block until the obtention of the final recommendation score.

Several authors have proposed solutions based on monolithic hybridization [17, 16, 20]. Minkov et al. [17] discuss the event recommendation problem, taking the particular case of scientific talks. In order to solve this problem, the authors have presented a content-based approach based on a RANKSVM formulation, in order to recommend events to users having into account their past attended events, as well as the events' descriptions. Simultaneously, the authors have proposed a collaborative extension called LOWRANK, by decomposing the users' parameters into individual and shared components. Their results show that the LOWRANK approach presents better recommendation performance when compared to a pure content-based approach. However, the solution proposed by the authors presents some disadvantages, such as the fact that it does not make use of context-aware information, as well as the fact that it requires explicit feedback from the users.

Macedo et al. [16] proposed a context-aware approach for event recommendation in event-based social networks (EBSNs). Their solution considers, not only the sets of users and events, but also several contextual signals, such as the user's time preferences in terms of event attendance, the groups the user belongs to, the user's preferences in terms of geographical distances, and the textual content of the events. The authors then developed context-aware models for each one of the contextual signals considered previously, which were then used as input features in a learning to rank approach. The obtained results show that the proposed solution

[1] http://www.cs.cmu.edu/~dongyeok/project/Eventera/

presents better recommendation performance when compared to other state-of-the-art recommendation approaches, particularly, *Most Popular*, *Bayesian Personalized Ranking-Matrix Factorization* (BPR-MF), and BPR-NET [19]. They also concluded that the use of contextual signals can lead to a better recommendation performance, as well as to mitigate the new user and new item cold-start problems in EBSNs.

Rendle et al. [20] proposed the use of Factorization Machines (FMs) in recommendation problems, in order to model contextual information, as well to provide context-aware recommendations. They also addressed the importance of using context-aware information in recommendation problems. Their method was compared to *Multiverse Recommendation* [10], one of the best methods for context-aware recommendation. Their results show that FMs present better performance than *Multiverse Recommendation* in terms of computational complexity, and a prediction quality comparable or better than the one presented by *Multiverse Recommendation*.

Other authors also proposed solutions based on parallel hybridization [11, 23]. Khrouf and Troncy [11] proposed an hybrid approach built over the Semantic Web. Their approach combines a content based filtering system enriched with Linked Data [1] and a collaborative filtering system. Given the fact that the events' similarity in a CB approach can be influenced by the topic diversity of the events the users attended in the past, a user diversity model based on Latent Dirichlet Allocation (LDA) [2] was also added. Finnaly, the authors proposed an hybridization strategy based on the use of a linear combination of the recommendation scores given by each one of the two approaches above. The obtained results show that their approach outperforms the tradicional user-based CF and UBExtended [18] approaches.

Zhang et al. [23] proposed three event recommendation approaches based on the semantic similarity, the relationships between users, and the users' event attendance history. Like Khrouf and Troncy [11], they also proposed an hybridization strategy based on the use of a linear combination of the recommendation scores given by each one of the three approaches proposed by them, in order to obtain the similarity between a user and an event. Their results show that each one of the three individual approaches outperforms a random event recommendation strategy. They also concluded that the hybrid strategy they proposed outperforms both each one of the individual recommendation approaches, as well as the random event recommendation strategy.

Finnaly, other authors also proposed solutions based on pipelining hybridization [14]. Li et al. [14] discussed the importance of personalized and location-aware services, in the context of mobile event recommendation. Thus, they presented a Multi-Stage Collaborative Filtering process to generate event recommendations, which is divided in two stages. In the first one, a User-to-User CF is performed, in order to agglomerate neighbor users with similar profiles and preferences, using an ART network. In the second one, an Item-to-Item CF is performed, in order to discover the event participation patterns of the users, which culminates in the generation of sequential rules. This rules, which allow to predict the next possible locations of the mobile users, are then crossed with information regarding events, which results in the obtention of matches between the future locations of the mobile users and the events. The final step in the recommendation process consists in the sorting of the top-N events to be recommended to a given user according to its preferences.The obtained results show that their approach outperforms the classical user-based CF and item-based CF approaches.

2.2.3 Comparison of approaches to the event recommendation problem

Due to the appearance of many event recommender approaches, it is also important to evaluate them in order to estabilish which are the best ones. Several authors did studies regarding this topic [15, 6].

Macedo and Marinho [15] investigate and discuss different characteristics of the EBSNs, such as the number of positive RSVPs[2], the events' lifetime, if the RSVPs are given closer or farther to the event's occurence date, the distribution of the co-participation in events by two distinct users, and the data regarding geographical distances between the users and the events. The following recommendation approaches were also studied: random, most-popular, location-aware, Bayesian personalized ranking-matrix factorization (BPR-MF), user-KNN, item-KNN, and logistic regression. Their results show that the user-KNN algorithm presented the best performance when applied to the event recommendation task in EBSNs. They also concluded that the item-KNN and logistic regression algorithms presented performances comparable to the one presented by the user-KNN algorithm.

Dooms et al. [6] focused on a user-based evaluation of several event recommender approaches, in order to find the best one. The following algorithms were tested: Random, User-based Nearest Neighbor Collaborative Filtering (UBCF), Singular

[2]From the french *répondez s'il vous plâit*. RSVPs contain information on the intention of a given user attend or not a given event

Value Decomposition (SVD), Content-based Filtering (CB), and an hybrid approach (UBCF + CB). For each user considered in the study, a randomly choosed recommendation algorithm were used to generate the event recommendations , after which it was asked them to answer several questions regarding the recommendation precision, novelty, diversity, satisfaction, and thrust in the system. The retrieved data show that the hybrid approach UBCF + CB outperforms all the remaining algorithms, except in terms of diversity. On the other hand, the SVD and random recommendation strategies obtained the worst classification in all the aspects considered, except in terms of diversity. Finally, they also concluded that the aspects that mostly correlate with the user satisfaction were the recommendation precision and transparency.

3. Implementation

In this section, I will present the various aspects regarding the event recommender engine implemented, particurlarly: (1) the features used, (2) the classification model used, (3) the architecture of the event recommender engine developed, and (4) the importance of feature selection applied to the event recommendation problem.

3.1. Features

Feature creation is one of the most relevant steps that one must take into account when working with event recommendation. The main goal of feature creation is to model a user-event pair in terms of a set of attributes/variables. These ones can then be used in a classifier, in order to generate recommendations. The process of feature creation is also typically an incremental and time consuming process since that it involves an exhaustive analysis of the datasets that will be used later.

For the event recommender engine developed, a set of 45 features was created. This ones can be classified into one of 6 distinct groups:

1. **Demographic and location data**: Features that explore demographic and location data related to the users and the events, e.g., the geographical distance, in quilometers, between the user's address and the event's location;
2. **Users' data**: Features that explore user-related information, e.g., the user's age or gender;
3. **Temporal data**: Features that explore temporal information, e.g., the time difference between the timestamp when the event will take place in the future, and the timestamp when the user saw the event in the system;
4. **Social-aware data**: Features that explore information regarding the social aspect, e.g., the number of friends of the user target of recommendation that will attend the event to be recommended;
5. **Collaborative data**: Features that explore collaborative data, e.g., the number of users that will attend the event to be recommended;
6. **Similarity data**: Features that explore information regarding the similarity between events, e.g., the similarity between the event to be recommended and the events the user target of recommendation attended in the past.

In some situations, the determination of the distance values between a user's address and an event's location is impossible. In such situations, the problem is solved by determining the mean distance between a user's address and an event's location, being this value applied to all the user-event instances without it.

Also, the determination of any of the cosine similarity values considered is sometimes impossible. In such situations, the problem is solved by determining the mean cosine similarity values for each similarity feature considered. This values are applied to all the user-event instances without it only if the number of attendances, attendance intentions, attendance invitations, or attendance refuses are null.

3.2. Classification Model

After the creation of all the necessary features, we need to apply them in a classification method in order to generate recommendations. For the event recommender engine developed, a Random Forest classifier was applied. A Random Forest classifier is a supervised classification method which consists in an ensemble of decision trees. The classification is done through a voting scheme. Each decision tree in the Random Forest classifier outputs a given classification, being the final classification obtained from the most voted class among all the decision trees [5].

The Random Forest classifier used in the event recommender engine developed was the one implemented using the *scikit-learn* library[3]. In this Random Forest classifier, each decision tree is created according to the following procedure:

1. From the original training set, choose a random subset of training examples, with replacement (bagging [4]);
2. From the random subset of training examples previously drawn, choose a random subset of features;

[3] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

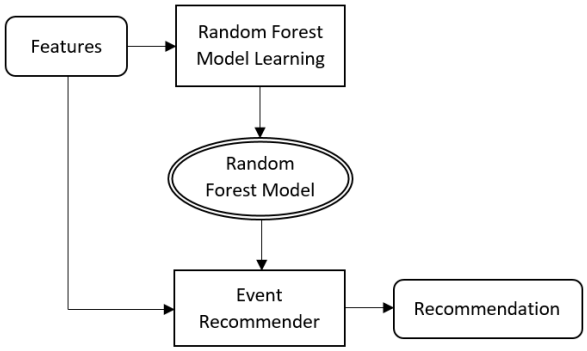

Figure 2: Architecture of the event recommender system developed.

3. Use this random subset of training examples with the random subset of features to create each decision tree.

Random Forest classifiers present some advantages. In addition to be a simple classification method, they also do not overfit, due to the Law of Large Numbers [5]. Also, the use of bagging jointly with random feature selection allows the creation of Random Forest classifiers with enhanced accuracy [5].

3.3. System Architecture

The general architecture of the event recommender system developed is presented in Figure 2.
Initially, a Random Forest classifier is trained based on a set S of training examples in the form $(\mathbf{x}, y)$, where $\mathbf{x} \in \Re^n$ is a vector with 45 features which represents a user and a candidate item (event), and where y denotes the prediction target. As a result of this training process, an event recommender model is then generated, being this one applied within the recommender in order to generate event recommendation predictions for each user considered.

These predictions are obtained from a set of test examples in the form of a feature vector $\mathbf{x}$, as presented previously. For each training example given to the Random Forest classifier, a probability value $P \in [0, 1]$ is returned, which denotes the likelihood of a given user come to attend a given event to be recommended. The final step in the recommendation process involves the sorting, for each user considered, of its recommendation lists by decreasing order of relevance. This is done through the probability values returned by the Random Forest classifier.

3.4. Feature selection

As seen previously in Section 3.1, the feature creation process presents itself as one of the most important aspects in machine learning. In general, features can be characterized according to the following groups [13]:

1. **Relevant features**: Features that have influence on the classification result. It's role in the classification process cannot be assumed by other features;
2. **Irrelevant features**: Features that have no influence on the classification result;
3. **Redundant features**: Features that can take the role of other ones already existing.

Thus, in order to obtain the best classification performance, we must select all the relevant features for a given classification problem. The use of irrelevant or redundant features in a classification problem leads to a poor performance, since this features have a low or null predictive power, do not contribute to the classification results, and can even introduce noise in the classification process.

The process of selecting relevant features instead of irrelevant or redundant ones is called feature selection. In this process, we try to select a small subset of relevant features from the original feature set, which, when applied in a machine learning process, can lead to the creation of classification systems with improved performance [13].

This process presents some advantages [13]:

1. Dimensionality reduction of the space of features, making algorithmic execution faster and facilitating the data visualization;
2. Removal of irrelevant, redundant or noisy information;
3. Reduction of the execution time of the learning algorithms used;
4. Increasing of the data quality;
5. Increasing of the accuracy of the resulting models;
6. Increasing in performance, with the corresponding gain in terms of predictive accuracy.

There are several feature selection methodologies available. The feature selection procedure applied to the event recommender engine developed is based on a feature selection methodology called Sequential Forward Selection (SFS) [13]. It is also the simplest one among them. In this algorithm, initially, the relevant feature set is empty. Then, for each iteration of the SFS algorithm, it is added to the relevant feature set the feature $\mathrm{F_x}$ that, when combined with all the previously selected features, maximizes a given evaluation metric. This procedure ends when (1) the number of selected features is equal to the total number of features implemented,

or (2) when the maximum score obtained in the current iteration is lower than the maximun score obtained in the previous one.

4. Evaluation

This section aims to present the various aspects regarding the evaluation of the event recommender engine developed, particularly: (1) the dataset used, (2) the evaluation metric employed, and (3) the obtained results. A discussion of the obtained results will be done at the end of this section.

4.1. Dataset

In order to proceed to the evaluation of the implemented event recommender engine, a dataset is needed. The dataset chosen was from the Kaggle Event Recommendation Engine Challenge[4], a challenge hosted by Kaggle in 2013. This dataset has 38209 users and 3137972 events.

4.2. Evaluation Metric

The selection of the appropriate evaluation metric (or metrics) for a given problem is a crucial step in any evaluation procedure. Various authors addressed the evaluation of recommender systems [7, 21]. In Section 2.2.3 several works regarding the evaluation and comparison of recommender systems were also presented [6, 15].

For the particurlar case of the event recommender system developed, the metric MAP@n, with $n = 200$, was used. For the choice of this evaluation metric, two aspects were taken into account. On one hand, the event recommender engine implemented aims to give, for each user considered, a list of recommended events ordered by decreasing order of relevance, making this evaluation metric suitable for this case. On the other hand, the dataset used to support experience making was the one from the Kaggle Event Recommendation Engine Challenge. In this competition, the evaluation metric used was MAP@200. This allows the comparison of results between the ones obtained with the event recommender engine developed and the ones reported by the participants of the Kaggle Event Recommendation Engine Challenge.

To determine the value of MAP@n, we need first to obtain the various values for AP@n, with $n = 200$, according to Equation 8:

$$\text{AP@n} = \frac{\sum_{k=1}^{n}[\text{Precision}(k) \times \text{Relevance}(k)]}{\min(m, n)} \quad (8)$$

In the equation, Precision(k) is the precision value at k-th position (also called P@k), m is the total number of relevant events, and Relevance(k) is a function that can take the following values:

$$\text{Relevance}(k) = \begin{cases} 1 & \text{, if the k–th event is relevant} \\ 0 & \text{, if the k–th event is not relevant} \end{cases} \quad (9)$$

Finally, the value of MAP@n is calculated using the values of AP@n, according to Equation 10, wherein AP@n_u is the value of AP@n for each user u, according to Equation 8, and N is the total number of users.

$$\text{MAP@n} = \frac{\sum_{u=1}^{N} \text{AP@n}_u}{N} \quad (10)$$

4.3. Results

This section aims to present the evaluation results obtained with the event recommender engine implemented. The evaluation procedure applied is divided in two distinct stages:

- **Stage 1**: Results obtained through the use of all the implemented features;
- **Stage 2**: Results obtained through the use of a feature selection procedure.

The MAP@200 values presented in this paper were obtained through the use of trec_eval, an evaluation tool from the Text Retrieval Conference (TREC)[5]. Each stage of the evaluation procedure is described below.

4.3.1 Stage 1: Using all the Implemented Features

First and foremost, it is important to assess the performance of the event recommender engine developed when all the 45 features implemented are applied to it. Taking into account the obtention of the best possible results, a Random Forest classifier with 1400 trees was applied in the event recommender engine developed.

The execution of this experiment resulted in a MAP@200 = 0.7059.

4.3.2 Stage 2: Using a Feature Selection Procedure

The next stage of the evaluation procedure involves the application of a feature selection procedure, in order to understand: (1) what are the relevant features for the event recommendation problem treated, and (2) what is the performance of the event recommender engine developed when this features are applied to it.

The feature selection procedure applied is the one described bellow:

[4] https://www.kaggle.com/c/event-recommendation-engine-challenge

[5] http://trec.nist.gov/

1. The first step involves the creation of 45 subsets of features from the original feature set. The first feature subset contains the first feature from the original feature set, the second feature subset contains the first two features from the original feature set, etc.;

2. Next, an SFS feature selection algorithm is applied to each one of the 45 feature subsets previously created. In this step, a Random Forest classifier with 1200 trees was employed by the SFS algorithm. This allows to retrieve all the relevant features for each tested subset, as well as its corresponding MAP@200 values;

3. Choose the subset of selected features (from the previous step) that leads to the best MAP@200 value;

4. Rerun the SFS feature selection algorithm over all the implemented features. Instead of the usual empty set, the SFS algorithm applied in this step considers the initial set of selected features as the one obtained in the previous step. The main goal is to try to select and add to the relevant feature set other features potentially relevant for the event recommendation problem, but not selected by the SFS algorithm applied in the previous step. In order to find the best possible MAP@200 value, the SFS algorithm was executed several times, being the number of trees of the Random Forest classifier different from run to run, ranging between 600 to 1600 trees, in increments of 100 trees;

5. Once again, choose the subset of selected features (from the previous step) that leads to the best MAP@200 value;

6. Repeat steps 4 and 5 until the maximum MAP@200 value obtained in the current iteration is smaller that the MAP@200 value obtained in the previous iteration.

As a result of this feature selection procedure, a small subset of 15 features were selected from the original set of 45 features. The number of trees of the Random Forest classifier used by the event recommender engine was also tuned to the 700 trees. The selected features were then applied into the event recommender engine developed, resulting in a value of MAP@200 = 0.7475.

4.4. Discussion

The final step in the evaluation process consists in the discussion of the obtained results. Three aspects will be considered in this discussion: (1) the features used in the event recommendation process, (2) the influence of feature selection procedures in the construction of better event recommender systems, and (3) the comparison of results to those obtained by the participants of the Kaggle Event Recommendation Engine Challenge.

As presented previously in this paper, several authors emphasized the use of contextual information as a way to obtain recommender systems with increased performance [16, 20, 14, 15]. Also, the maximum value of MAP@200 was obtained through the use of a subset of 15 features, as described previously in Section 4.3.2. As expected, this ones express contextual information, such as the geographical distance between the user target of recommendation and the event to be recommended, leading the implemented event recommender engine to present a high recommendation performance.

Another aspect to be considered is the influence of feature selection procedures in the construction of better event recommender systems. In the Section 3.4 of this paper, it was explained how feature selection methodologies can increase the performance of recommender systems through the selection of relevant features, instead of irrelevant and redundant ones. The various advantages of using such methodologies were also explained. Thus, as expected, the use of the feature selection methodology explained previously (see Section 4.3.2 of this paper for details) not only led to the selection of a small subset of features from the original feature set, but also led to a better recommendation performance, as shown in Section 4.3.2.

Finally, and since the dataset used in the experiences above was the one from the Kaggle Event Recommendation Engine Challenge, a comparison of results to those obtained by the participants of his competition must be done. Since the Kaggle platform does not provide the complete solutions for the callenges hosted by them, the results expressed in the public leaderboard of this competition were used for comparison, instead of the ones expressed in the private leaderboard. The analysis of the public leaderboard shows that the participant DataLab was the one with the highest score (MAP@200 = 0.72876). Through the use of the event recommender engine developed, a value of MAP@200 = 0.7475 was obtained. The comparison of scores shows that the event recommender engine developed and explained in this paper outperforms the solution presented by DataLab.

5. Conclusions and Future Work

This paper addressed the thematic of recommender systems and how they are important in the current panorama of the Internet, both for users and for entities that make use of them. It has also addressed the problem of event recommendation and how this new recommendation problem is different

from the traditional recommendation. The main articles from the literature in this area were also presented, in order to understand, in detail, the problems inherent to the event recommendation, as well as the solutions adopted by several authors in order to solve this problem. Having in consideration the related work presented, an event recommender engine was developed. This system provides, for each user, a list of recommended events ordered by decreasing order of relevance. In order to achieve the best recommendation performance possible, the system developed presents an hybrid architecture and makes use of several types of information, particularly: (1) information about the users, (2) information about the events, and (3) contextual information. Through an extensive analysis of the dataset used, a set of 45 features were implemented, being this ones applied in a Random Forest classifier, in order to generate the recommendation lists for each user. A feature selection procedure based on the SFS algorithm was also applied, in order to increase even more the performance of the event recommender engine developed. The application of this procedure resulted in an even higher recommendation performance than the one obtained trough the use of all the features implemented. A comparison between the implemented solution and the ones implementd by the participants of the Kaggle Event Recommendation Engine Challenge was also made. This comparison shows that the implemented solution outperformed the ones presented by the participants of that Kaggle competition.

Last, but not the least, there are also some hypothetic future work planned for this event recommender engine, particularly:

1. The creation/implementation of other features, in addition to those already implemented and used by the event recommender engine;

2. The test of other machine learning methodologies, in addition to the ones already tested duting the implementation of the event recommender engine;

3. The use of other feature selection methodologies, in addition to the one applied in the context of this dissertation.

Acknowledgements
The author would like to thank to Pavel Calado and Bruno Martins, supervisor and co-supervisor of this master thesys, respectively, and part of the Information and Decision Support Systems team at INESC-ID, for all the help, support and supervisation of this work.

References

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3), 2009.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0), 2013.

[4] L. Breiman. Bagging predictors. In *Machine Learning*, 1996.

[5] L. Breiman. Random forests. In *Machine Learning*, 2001.

[6] S. Dooms, T. De Pessemier, and L. Martens. A user-centric evaluation of recommender algorithms for an event recommendation system. In *Proceedings of the RecSys Workshop on Human Decision Making in Recommender Systems and User-Centric Evaluation of Recommender Systems and Their Interfaces*, 2011.

[7] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 2004.

[8] D. Jannach and G. Friedrich. Tutorial: Recommender systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2013.

[9] D. Kang, D. Han, N. Park, S. Kim, U. Kang, and S. Lee. EVENTERA: Real-time event recommendation system from massive heterogeneous online media. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, 2014.

[10] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems*, 2010.

[11] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *Proceedings of the ACM Conference on Recommender Systems*, 2013.

[12] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, 2007.

[13] L. Ladha and T. Deepa. Feature selection methods and algorithms. *International Journal of Advanced Trends in Computer Science and Engineering*, 3(5), 2011.

[14] L.-H. Li, F.-M. Lee, Y.-C. Chen, and C.-Y. Cheng. A multi-stage collaborative filtering approach for mobile recommendation. In *Proceedings of the International Conference on Ubiquitous Information Management and Communication*, 2009.

[15] A. Q. Macedo and L. B. Marinho. Event recommendation in event-based social networks. In *Proceedings of the International Workshop on Social Personalization*, 2014.

[16] A. Q. Macedo, L. B. Marinho, and R. L. Santos. Context-aware event recommendation in event-based social networks. In *Proceedings of the ACM Conference on Recommender Systems*, 2015.

[17] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. Collaborative future event recommendation. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2010.

[18] T. D. Pessemier, S. Coppens, K. Geebelen, C. Vleugels, S. Bannier, E. Mannens, K. Vanhecke, and L. Martens. Collaborative recommendations with content-based filters for cultural activities via a scalable event distribution platform. *Multimedia Tools and Applications*, 58(1), 2012.

[19] Z. Qiao, P. Zhang, C. Zhou, Y. Cao, L. Guo, and Y. Zhang. Event recommendation in event-based social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014.

[20] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

[21] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*. 2011.

[22] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys*, 47(1), 2014.

[23] Y. Zhang, H. Wu, V. S. Sorathia, and V. K. Prasanna. Event recommendation in social networks with linked data enablement. In *Proceedings of the International Conference on Enterprise Information Systems*, 2013.