RépubliqueAlgérienneDémocratiqueetPopulaire
Ministère de l'EnseignementSupérieuret de la RechercheScientifique
UniversitéChadliBendjedid
Faculté des Sciences et de la Technologie
Départementd'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

Master Thesis in Computer Science

Presented by

ATHMANI HoussemEddin

Specialty: Intelligent Computer Systems

THEME

Hotel Context-Aware Recommender System

Defence on: 15 / 09 / 2022

Member of the jury:

Quantity	First And Last Name	Grade	Université
President	Mr.Betouil A.A	MCB	ChadliBendjedid El-Tarf
Supervisor	**Mme Gasmi I.**	**MCA**	**ChadliBendjedid El-Tarf**
Examiner	Mme Ziani A.	MCB	ChadliBendjedid El-Tarf

University Year: : 2021/2022

Acknowledgment

My Thanks Go First To Allah For The Will, Health, He Gave Me Patience During All These Years Of Study.

Special Thanks Goes To My Supervisor Gasmi I, Who Inspired Me To Step Up And Face Life Challenges And Admire Your Work No Matter What Happens

A Huge Thanks To My Parents For The Encouragement And Support During My Career In My Best And Worst

A Big Shoutout And Huge Thanks To My Besties, Ines And Aya , We Faced All Life Challenges And Finally We Did it Together , Without Forgetting Bilel The Right Hand When You Need It The Most

From Bottom Of The Heart Thank You

Table of Contents

Figures table

General Introduction

The human brain has developed to deal with our complex world, which is incredible. However, technology has just been added to this world, and humans have become reliant on it to complete daily duties. As a result of this vicious cycle, people increasingly require outside assistance to make sense of the world, particularly input from others based on their prior experience. Some existing recommender system approaches focus on recommending the most relevant items to individual users without taking into account any contextual information, such as time, place, or the presence of other people. Especially when it comes to planning a vacation on hotels in winter, for example, a travel recommender system might provide a recommendation that is significantly different from one made in the summer. Context-aware recommender systems (CARS), answer this and several other related problems, and show that at least some contextual information can be valuable for generating improved recommendations. We also offer three approaches for incorporating contextual information into recommender systems, evaluate each strategy separately, and explore how these three methods might be merged into a single cohesive approach. Finally, by including contextual information into the suggestion process, end-users and recommender systems can engage in richer and more diverse interactions. This thesis has three chapters which are divided as follows:

- **Chapter 1:**

The first chapter presents an overview of recommender systems. Followed different recommendation approaches, and diving deep into the world of context-aware recommenders as well as the advantages and disadvantages of the system.

- **Chapter 2:**

In the second chapter, we describe the supervised machine learning approach using neural network integrating the context, we ended by predicting the top n items recommended for the user

- **Chapter 3:**

The last chapter covers the experimentation of our approach :tools, software and hardware, dataset, execution sets and evaluation of the proposal as well as the implementation of our approach.

Chapter 1: Stateof Art

I. Recommender Systems (RS)

I.1. History

The concept of using computers to select the best product for a user has existed since the dawn of computing. The first application of the RS concept arrived in 1979, in the form of Grundy, a computer-based librarian who recommended books to the user. This was followed by the release of Tapestry, the first commercial RS, in the early 1990s. In the early 1990s, GroupLens, a research lab at the University of Minnesota, introduced another RS implementation to assist users in finding their chosen articles. GroupLens Recommender System was the name given to the system after the band.[1]

In the late 1990s, Amazon Collaborative Filtering, one of the most well-known RS technologies, was implemented, furthering the development of RSs. RSs based on Collaborative Filtering have been highly popular in recent years, and many e-commerce and internet systems have integrated them. Many RS toolboxes have also been created. Amazon's success led to the creation of a number of RS algorithms known as hybrid methods, which integrate different methodologies.

After a successful period at the end of the 1990s, the industry granted considerable support to put RSs strategy into action. Netflix, a supplier of internet streaming media, hosted the most popular competition in RSs. In 2006, they established the Netflix Prize, which awards a million dollars to the person who provides the best RS movie recommendation. In 2009, they declared the winning team. [2]

I.2. Definition

Recommender Systems (RS) are software tools and approaches that offer users with customized item suggestions based on a huge collection of items.Information retrieval, machine learning, decision support systems, human computer interface, marketing, and other domains are all used in RS,There has been a lot of research into the various elements and challenges of RS, notably because of their utility in situations where users are overwhelmed by information.They have a wide range of applications since they may

improve core business KPIs including customer pleasure, loyalty, and revenue. In fact, Netflix has used RS to make movie recommendations, Amazon products, YouTube videos, Spotify music, and social media platforms such as Facebook and Twitter.

In all of their forms, RS examines individual users' historical behavior, which exposes their preferences for specific things and is documented through various behaviors such as clicks, ratings, and purchases. These activities, along with the patterns discovered, are then utilized to anticipate item relevancy and generate suggestions that are customized to the user profiles.[3]

This chapter gives a broad review of the field of RS, with a focus on the definitions, concepts, and approaches. We give a comprehensive classification of recommendation techniques, focusing on context-aware approaches, after examining the methods used to evaluate RS.

I.3. Purpose of recommender systems

A recommendation system aims to provide a user with relevant resources based on their preferences. The latter thus sees his search time reduced but also receives suggestions from the system to which he would not have spontaneously paid attention. The rise of the Web and its popularity have notably contributed to the establishment of such systems as in the field of e-commerce. Recommender systems can initially be seen as a response given to users who have difficulty making a decision when using a “classic” information retrieval system.

A recommendation system generally requires three (03) steps, which are illustrated in the following figure:

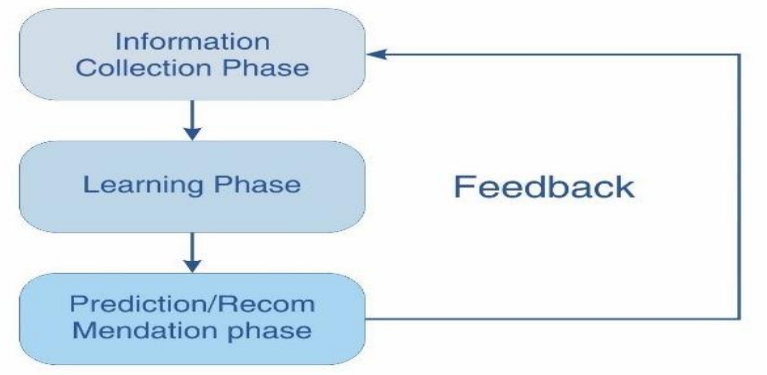

Figure 1: Recommendation system phases

I.4. Recommender Systems Approaches

A recommendation method, also known as a recommendation algorithm, is designed to estimate the utility of products for a group of target users and provide appropriate recommendations. A wide range of approaches to giving actual recommendations have been proposed, and they can be classified based on a number of factors, such as the type of recommendation problem they address, such as rating prediction or top-N recommendation, or the type of feedback they use, such as explicit or implicit feedback.

The most frequent classification in the literature relates to how information about users and items is used for the recommendation task, and it divides users and items into the following categories:

I.4.1. CBF (Content-Based Filtering) techniques.

These methods make explicit use of user or item-specific domain knowledge.For content-based recommendations [4], the task is to determine which catalog items best match the user's preferences. Such an approach does not require a large user community or a large system usage history. Figure 2 illustrates this process

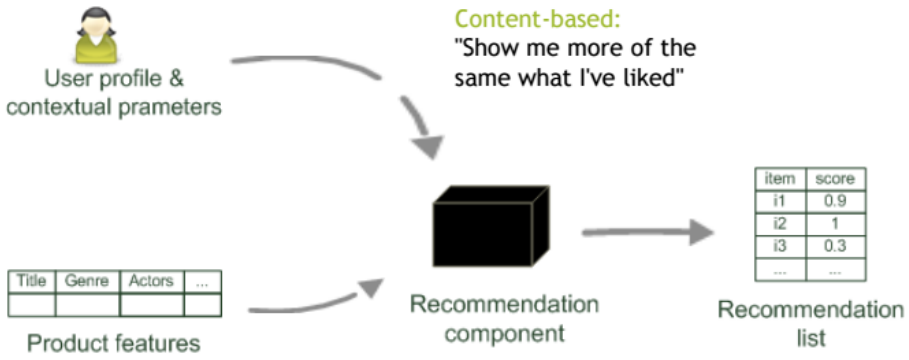

Figure 2: A content-based recommender system [4]

The easiest way to describe an item catalog is to have an explicit list of each item's characteristics (also known as attributes, item profile, etc.). For a book, for example, one can use the genre, the name of the authors, the publisher or any other information relating to the book, then store these characteristics (in a database for example).

The user profile is expressed in the form of a list of interests based on the same characteristics. The coincidence between the characteristics of the elements and the profile of the user can be measured in different ways:

- Dice's index or other measures of similarity;
- TF-IDF (Term Frequency-Inverse Document Frequency);
- Techniques based on the similarity of vector spaces (Bayesian approaches [2], decision trees, etc.) coupled with statistical techniques, when there are too many keywords.

I.4.2. Collaborative Filtering (CF) approaches.

Systems based on collaborative filtering [5] produce recommendations based on calculating the similarity between a user's preferences and those of other users. Such systems do not attempt to analyze or understand the content of the items to be recommended. The method is to make automatic predictions about a user's interests by collecting reviews from many users. The underlying assumption of this approach is that those who liked a specific item in the past will tend to like that specific item, or another very "close" one, again in the future. Figure 3 illustrates this process

Figure 3: A Collaborative-based recommender system [6]

The idea of collaborative approaches is to try to predict a user's opinion on the different elements. The recommendation is based on previous tastes and opinions of the user and on a measure of similarity with other users. The main steps in this approach are:

- many user preferences are stored;

- a subgroup of users is identified whose preferences are similar to those of the user seeking the recommendation;
- an average of the preferences for this group is calculated;
- the resulting preference function is used to recommend items to the user seeking the recommendation.

It is possible to distinguish three types of approaches to define similarity or similarity:

- Item-to-Item approaches based on the similarity between items (items). Note that this approach adapts to a very large number of users or elements.
- User-to-User approaches based on the similarity between users. Note that this approach is not suitable for a very large number of users.

Collaborative recommender systems, by their diversity, are therefore based on many techniques, whether:

- similarity between users (Pearson correlation coefficient [7], etc.) or neighborhood selection (algorithms based on neighborhood search) for User-to-User approaches;
- similarity between items (like cosine similarity measure) for Item-to-Item approaches;
- score prediction techniques (principal component analysis or PCA, matrix factorization, latent semantic analysis, association rules, Bayesian approaches, etc.) for other approaches.

I.4.3. Hybrid approaches.

A hybrid recommender system uses components of different types of recommendation approaches or relies on their logic [3]. For example, such a system can use both external knowledge and the characteristics of the elements, thus combining collaborative and content-based approaches [6].

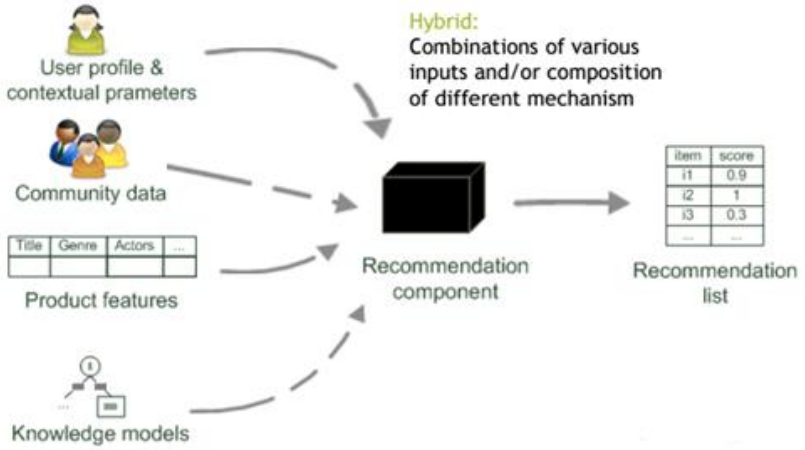

Figure 4: The hybrid recommender system [6]

It should be noted that the term "hybrid" is an artefact of the historical evolution of recommender systems where certain sources of knowledge were exploited in the first place, leading to well-established techniques which were then combined. The objective is then to rely on multiple sources of knowledge, choosing the most appropriate to a given task in order to use them as efficiently as possible, Figure 4 illustrates this process.

There are three main categories of recommender system combinations for designing a hybrid recommender system [3][6]: the monolithic combination (monolithic hybridization design), the parallel combination (parallelized hybridization design) and the tubular combination (pipelined hybridization design).

“Monolithic” describes a hybridization design that integrates aspects of different recommendation strategies into a single algorithm. As illustrated in figure 5 below, different recommender systems contribute to this since the hybrid approach uses additional input data that is specific to another recommender algorithm, or the input data is supplemented by a technique and exploited by another. For example, a content-based recommendation system that also leverages community data to determine similarities between items falls into this category [3][6].

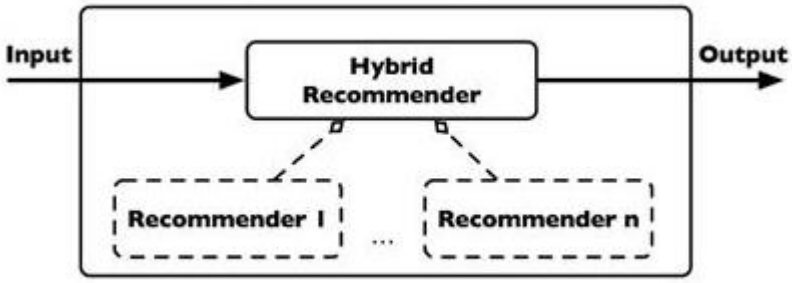

Figure 5: Monolithic hybridization design [6]

The other two hybrid approaches require at least two implementations of separate recommendations which are combined accordingly. Based on their input data, the parallel hybrid recommender systems operate independently of each other and produce distinct recommendation lists, as shown in Figure 6. In a later hybridization step, their outputs are combined into a final set of recommendations [3][6].

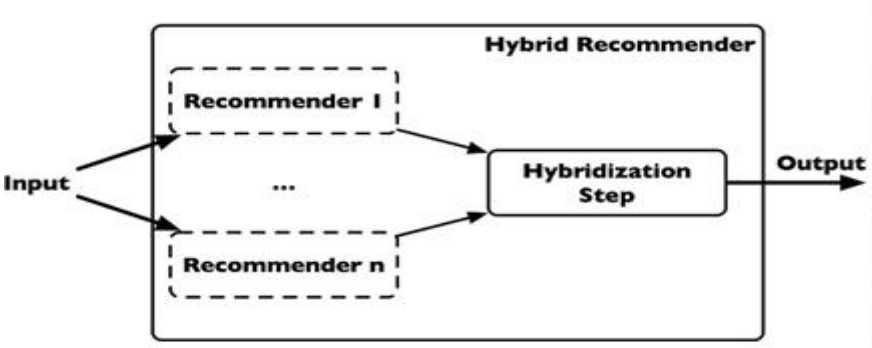

Figure 6: parallelized hybridization design[6]

When multiple recommender systems are joined in a tubular architecture, as shown in Figure 7, the output of one of the recommender systems becomes part of the input data of the next system [3][6]

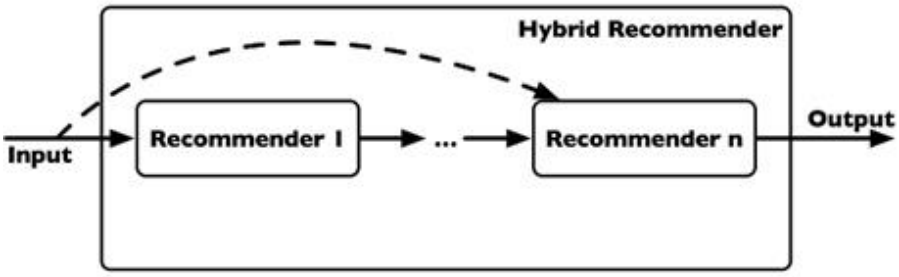

Figure 7: Pipelined hybridization design[6]

II. Context-Aware Recommender Systems (CARS)

II.1. Introduction

The objective of recommender systems is essentially to improve the interactions between users and systems for searching and accessing information. However, the recommender systems discussed above operate in a two-dimensional space (user, content). However, in mobile environments, other aspects must be considered in the recommendation. The user profile is insufficient to produce recommendations. There are several context-dependent items that should be considered (when?, where?, why?, etc.). Indeed, the context of the user influences the choice of relevant information.

A new family of recommender systems called context-aware recommender systems have been increasingly developed in the literature in recent years, which are based on the user's context by modeling data such as his location, time and people around. Filtering then consists of analyzing the current context of the user and selecting, among the available content, those that best correspond to the situation described.

In the first part, we will present the different visions of the definition of the context given by some researchers. Then we present the basic concepts to these systems, as well as the different paradigms for its integration into recommender systems.

The second part of the chapter will be devoted to context-aware recommender systems. We will give some definitions of the latter, then we will expose their architecture. We will end with some examples on context-aware SRs.

II.2. Definition of context

Context is a broad concept for which it is particularly difficult to give a general and operational definition. In computer science, several definitions have emerged by several researchers. In our work, we have selected a few that have been marked over the years.

First, the researchers started by defining the context by listing all the elements (these elements constitute non-functional information of the application, but which can have an impact on the latter) which can be considered as information of the context. Among these definitions we find:

Schilit and Theimer[8] who define the context as being “the location and identity of nearby people and objects as well as the modifications that may occur on these objects”. In the field of recommender systems, the definition used is that of Anind K. Dey and Gregory D. Abowd[e]. Their study aims to give the most precise definition possible.

For them, “the context is the set of all the information that can be used to characterize the situation of an entity”. To sum up, information is part of the context if it influences an interaction between two entities, an entity which can be an actor, a place, an object of the environment considered useful for the interaction between the person and the system.

II.3. Context-Aware Systems

The importance of contextual information in predicting item utility was realized as the RS field progressed. User decisions are influenced by context, present needs, and the scenario they are in, rather than by a fixed set of preferences tied to item qualities identified by CBF and CF techniques. When contextual information is taken into account, it can help generate better recommendations. The following are some examples:

• Location information. Location information might be valuable in a variety of recommendation scenarios. The user's geographic location can have a big impact on their taste, culture, and habits. It can also reveal the current user's behavior, such as work at the office or leisure at a pleasant location, which influences the set of relevant choices. User location is critical in some applications, such as restaurant recommendations, to filter out physically unreachable possibilities.

• Temporal information. Users' tastes different depending on the time of day, the day of the week, or the season in a variety of areas, such as clothing and tourism. As a result, temporal data has been used to increase recommendation quality.

• Social information. The user's behaviour is influenced by social variables. For example, depending on whether he is with friends or relatives, a user may make various decisions. Furthermore, and particularly in the context of online social networks, close user behaviours are likely to influence the target user and should be taken into account when making suggestions.

Integrating context into RS necessitates taking into account a dimension other than the user and item dimensions recorded in the feedback matrix. The representation space of

feedback becomes multidimensional as a result of the context being a complex variable. CARS covers the space U I C, where C is the set of contextual elements that are significant. Context values are frequently considered to be atomic and nominal, and that the context entity is the Cartesian product of numerous contextual dimensions.

is a subset of the Cartesian product of a collection of characteristics, each of which defines a set of values. Three popular paradigms were identified for addressing this multidimensional recommendation problem:

II.3.1. contextual pre-filtering

In the pre-filtering approach, contextual information is used as a label allowing to filter the evaluations which do not correspond to the contextual information specified. This is done before the main recommendation method is launched on therest of the selected data.

That is, if a particular context of interest is k, then this method selects from the initial series all the evaluations relating to the specified context k, and it generates the“Item x user” matrix containing only the data relating to context k.

After that, the recommender systems (SR) method, such as collaborative filtering,is launched on the reduced database in order to obtain the recommendations related to the contextk. Figure 8 shows the integration of context in a two-way recommender system.dimensions (user x product).

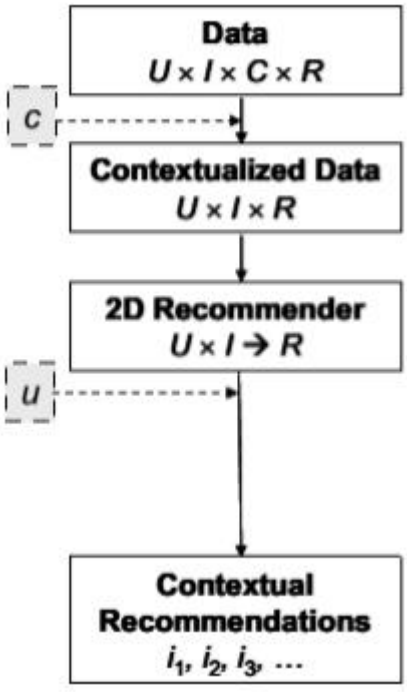

Figure 8 : Approche Pré-filtrage Contextuel [9]

II.3.2. Contextual post-filtering

According to the post-filtering approach, the contextual information is used after the launch of the main 2D (two-dimensional) recommendation method. Once the assessments unknowns are estimated and recommendations are produced, the system analyzes the data for a given user in a specific context to find patterns of use for specific articles, and uses these patterns to "contextualize" the recommendations obtained from the classic 2D recommendation method, such as collaborative filtering. The figure below shows how to use the context after the recommendation (2D)

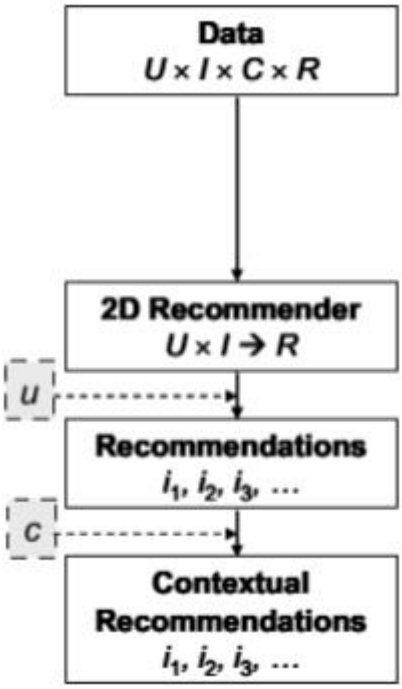

Figure 9 : Approche Post-filtrage contextuel [9]

II.3.3. contextual modeling

In addition to the two methods mentioned above, there is a third method, namely the contextual modeling in which contextual information is used directly to inside the

recommendation generation algorithms. Figure 10 shows integrating context into the recommendation process

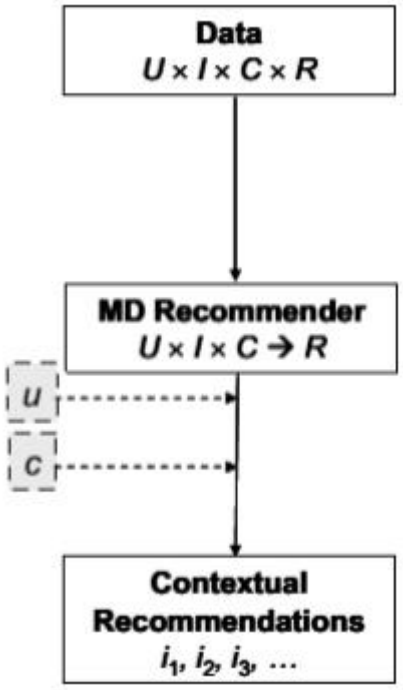

Figure 10 :ApprocheModélisationcontextuelle[9]

II.4. Micro-profiling

The micro-profiling approach [10] aims to recommend unknown songs/artists to a user. The type of recommendation system is pre-filtering because the data is filtered according to the context (time: time of day, day, month or year).

In this approach, assuming that the tastes of users change over a period and can be similar in the same period.

For example, a user prefers to listen to a certain kind of song when working, and another kind of song before sleeping.

We have a user profile in which we will assign the songs/artists listened to by this user. This profile will be broken down so that it adapts to each context: in this approach, time

represents the moment of the day, it will be divided into several segments according to the songs that the user listens to the most during each moment (segment of time) of the day.

The use of these micro-profiles improves the accuracy of the recommendation. One of the problems with this method is that the time partitioning differs from one user to another (instead of being fixed for all users), which risks decreasing the accuracy of the system.

II.5. Splitting approach to contextual pre-filtering

Among the algorithms proposed in the CARS, we are interested in the algorithms of the slicing approach. This approach belongs to the contextual pre-filtering approach of CARS.

This approach consists of not filtering the data but rather cutting (dividing) the data set into two subsets with respect to the value of one (of) the contextual dimension(s).

This splitting is only done if users evaluate an item differently under two contextual conditions, using statistical tests (impurity criteria). This approach will be detailed in the next chapter.

II.6. Context Features

In what follows, we will give the technical characteristics of the context information:

- **Continuity**: As the user moves from one location to another which implies a change in the value of the context, or as the user obtains new resources, the context evolves. This development should be accompanied by the continuous functioning of the systems despite the incessant changes of the context.
- **Heterogeneity**: A large heterogeneity in terms of modeling, processing and quality comes from the fact that the context is captured from a variety of sources [11]. For example, in computing, laptops, smartphones, tablets are different at all levels (hardware, software, communications, etc.).
- **Unpredictability**: One of the questions that arises is how a system should take into account the unpredictability of changes in the environment and user interactions. Because some elements of the context may not be known in advance in the system.
- **Interdependence**: Information and/or change in value of context information may be dependent on other information and/or change in value of context information [5].

- **Imperfect context**: this characteristic is determined according to the source of the information. The context may therefore be imprecise or else the context may be erroneous or even unknown.

III. Context in the Hotel Domain

in the hotel industry the RS promotes travel planning and helps enhance hotel experience and company profitability besides customer loyalty

Accommodation is an important part of the process, and for most individuals, choice of the best option is a difficult and time-consuming process. As in any industry sector where RS is used, it assists in trip planning and profitability, but it also introduces unique difficulties that cannot be addressed with typical suggestion methodologies.

A contextual component is offered in [12], which introduces four major categories, which we will discuss below:

- **Physical context:**The user's physical or environmental context includes their physical location, which has a significant impact on their decisions and behaviors. In applications where these characteristics are important, it also includes time, activity, weather, and temperature. Booking a hotel is determined by the traveler's intended destination, which is influenced by factors such as the traveler's country of residency, the season, and the weather.
- **Social context:**The presence and role of other people around the user are represented by the social context. During the decision-making process, users can be heavily influenced by others. This happens when a user books a vacation for a group of people, including himself. It also happens when a decision is influenced by other people's past experiences, such as social media reviews.
- **Interaction media context:**This context category describes the device used to connect to the system and the channel used to deliver personalized recommendations. Users' behaviors may differ depending on whether they are browsing on their mobile device or their laptop. Different recommendation strategies may then be used to address the various user attitudes. This issue is not addressed in our work, primarily because this type of information is either missing or incomplete across the data we handle.

- **Modal context:**The modal context encompasses the user's psychological condition, goals, and mood. It is very important in the selection phase. A single user, for example, behaves differently when planning a vacation trip.

IV.Advantages and disadvantages of recommender systems

Despite their growing popularity, recommender systems have suffered a few hiccups.

We identify a few problems they suffer from [13]:

- **Adaptability:** As the database of ratings grows, the recommendation becomes more accurate.
- **New user:** A new user who has not yet accumulated enough ratings cannot have relevant recommendations.
- **New item**: An item must have enough ratings for it to be considered in the recommendation process.
- **Cold start:** Often, we are faced with the problem that a user is not comparable with any other. This issue is caused by few or no users rating a given article, or a given user rating very few or no articles. Typically, this issue occurs when a new user or resource is added to the recommendation database.

- **The case of the beginner system:** comes from the launch of a new recommendation service. The system then has no information on the users and on the items. Collaborative filtering methods cannot operate on an empty usage matrix. The solution generally consists in finding descriptive information of the items in order to organize the catalog and encourage users to browse it until the matrix of uses is filled enough and allows to switch to collaborative mode. Table 2. summarizes the strengths and weaknesses of the traditional methods used by recommender systems, in this case Collaborative Filtering (CF), Content-Based Filtering (BCF).

V.Evaluation of Recommender Systems

Evaluating a recommender system allows us to measure its performance with respect to its objectives. As a result, the dimensions to be assessed differ according to the objectives set (prediction of user ratings for items, increase in sales, etc.)

There is a classification of methods for evaluating recommender systems.

As explained in [14], these evaluation methods are classified into three types: offline experiments, studies with users (user studies) and real tests (real life testing).

This last type is called online experiments by [8].

V.1. Online Evaluation

According to Herlocker[15] this type of evaluation can show the uses and habits usage patterns, problems and unmet needs, and issues that researchers may not have considered in a user study. With these real-world tests, most user-centric goals can be effectively assessed, such as user experience evaluation, user satisfaction, or user retention [15][8].

V.2. Offline Evaluation

Much of the work evaluating the algorithms of recommender systems has focused on offline analysis of the accuracy of the predictions these systems can make [15].

Offline assessments use datasets made up of user actions. Offline evaluations simulate the recommendation process where a subset of user actions in the dataset are hidden and the recommender system predicts these hidden actions. The recommender system is evaluated on its ability to predict these hidden interactions. The results of these predictions are analyzed using one or more metrics. [16]

V.3. Evaluation Metrics

Evaluation metrics measure the ability of the algorithm to recommendation .its last uses correct evaluations of a test dataset and compares them with the results predictions proposed by the recommendation algorithm based on a training dataset

The majority of these metrics come from the field of information retrieval. Precision and recall [16] are among these metrics.

- **Precision**: is a measure of accuracy. It determines the proportion of relevant items recommended among all items recommended: in other words, the proportion of recommendations that have proven to be relevant.
- **Recall**: is a measure of completeness that determines the proportion of relevant items recommended among all relevant items

Other metrics are used to evaluate recommender systems such as Mean Absolute Errors (MAE) and RootMean Square Errors (RMSE).

- **MAE**: calculates the difference between the predicted scores and the real scores.
- **RMSE**: it is based on the same principle as the MAE, while emphasizing significant deviations. [17].

VI. Conclusion

We present an overview of RS in this chapter, we address evaluation procedures and gave a various recommendation techniques. We discuss the compromises that each recommendation strategy entails, as well as the benefits and drawbacks of using each one for suggestion. Real-world applications were always a source of inspiration for RS research, as they constantly reframed the problem and the focus of current improvements. This chapter also tries to emphasize this point by charting the evolution of the problem and the implications from a business standpoint. Developing RS, on the other hand, necessitates a comprehension of user behaviour in the relevant area as well as the role the system is intended to play in order to determine which features should be included.

Chapter 2: System Study

I. Introduction

This chapter is devoted to the conceptual details of our algorithm. We present the general architecture of the proposed recommender system and we give a detailed design for each functionality.

II. General system architecture

The goal of this project is to build a hotel recommendation system using neural networks with Keras and TensorFlow library.

First, we created an Embedding layer for both the user and the item. Then, we combined their embedding using a dot product. An embedding model is a type of machine learning model that uses embedding as the weights that are learned during training stage.

The dataset splits into a training set to build the algorithm and a testing set for evaluate the learned model of the engine build. The input layer has an input about user, hotel id and the context.The output layer is a prediction of a recommended hotel to a specific user.

Embedding layer uses the weights that are learned during training with a dense layer for the testing set.

The recommendation engine input user's id to start using the system for the final predictions. The final predictions of the output get rearranged from the biggest value into the smallest and generate it into top n recommendations after checking the ID's of the hotels

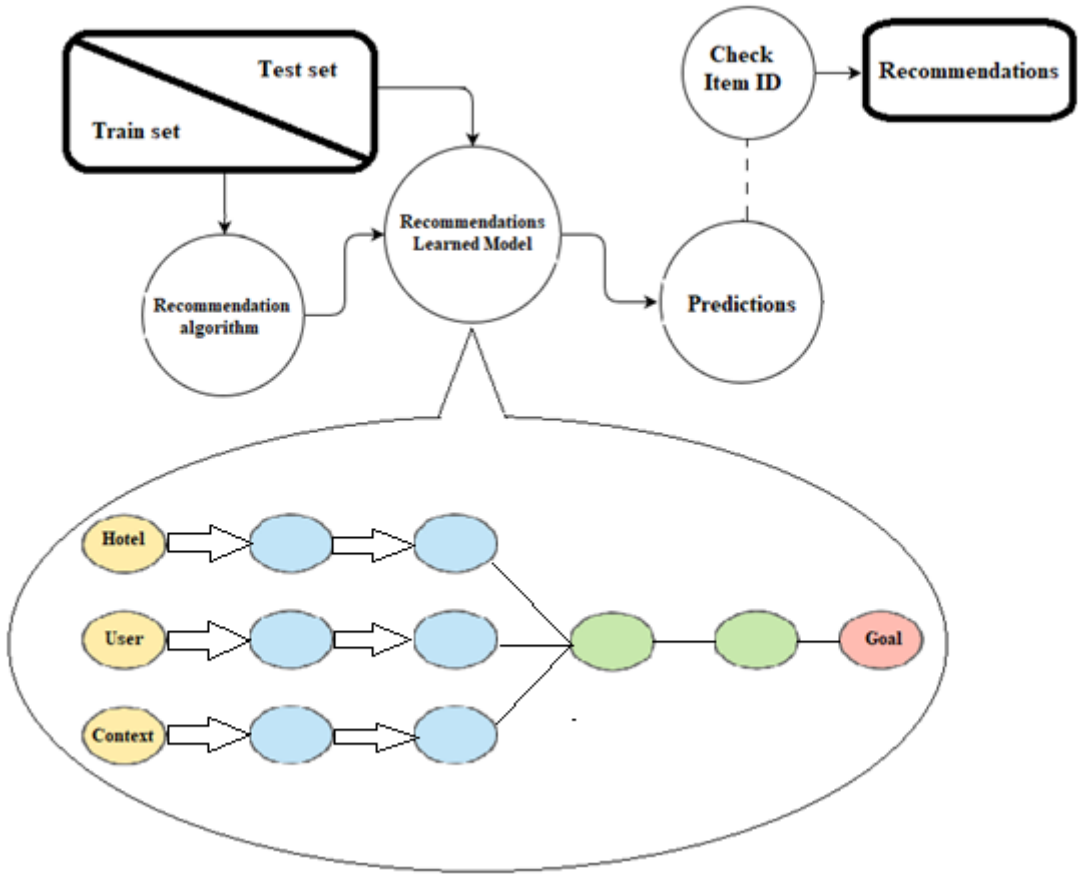

Figure11 : Supervised machine learning model

III. Dataset

We used Kaggle's Expedia Hotel Recommendation dataset, which was collected in recent years, It has a variety of features that could provide us with valuable insights into the process that users go through when selecting hotels.

The train csv set has 37,670,293 entries but cut into 100,000, while the test set has 10% of the entries. In addition, the dataset contains some latent features for each of the destinations recorded in the train and test sets. The information has been anonymized, and almost all of the fields are in numerical format for neural network usage.

The goal is to predict 5 hotel id's where a user is more likely to stay out of a total of 100 hotel clusters. The problem has been modeled as a ranked multi-class classification task. This dataset's main challenges are missing data, ranking requirements, and the curse of dimensionality.

IV. Data preparation

First,we clean and pre-process the data that we need to use. The first thing we noticed was that many users only searched for hotels and did not make any reservations.

As a result, we used the booking as a result output of the system because these entries do not provide any indication of which hotel clusters those users prefer and may have interfered with making predictions.

So we used the number of similar events in the context of the same user session as a context input of the dataset

Below we listed all the data types of the whole dataset :

Column name	Description	Data type
date_time	Timestamp	String
site_name	ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)	Int
posa_continent	ID of continent associated with site_name	Int
user_location_country	The ID of the country the customer is located	Int
user_location_region	The ID of the region the customer is located	Int
user_location_city	The ID of the city the customer is located	Int
orig_destination_distance	Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated	Double
user_id	ID of user	Int
is_mobile	1 when a user connected from a mobile device, 0 otherwise	Tinyint
is_package	1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise	Int
channel	ID of a marketing channel	Int
srch_ci	Checkin date	String

Column name	Description	Data type
srch_co	Checkout date	String
srch_adults_cnt	The number of adults specified in the hotel room	Int
srch_children_cnt	The number of (extra occupancy) children specified in the hotel room	Int
srch_rm_cnt	The number of hotel rooms specified in the search	Int
srch_destination_id	ID of the destination where the hotel search was performed	Int
srch_destination_type_id	Type of destination	Int
hotel_continent	Hotel continent	Int
hotel_country	Hotel country	Int
hotel_market	Hotel market	Int
is_booking	1 if a booking, 0 if a click	Tinyint
cnt	Number of similar events in the context of the same user session	Bigint
hotel_cluster	ID of a hotel cluster	Int

Table1 :Data fields of the csv file

Secondly we removed the extra columns and limited the rows into 100,000 to save wasted space and smooth input to gain time while Processing data.

So we left with 3 rows as input which is (user_id , hotel_cluster , cnt) and an extra row for the goal which is (is_booking)

As shown below

	User_id	Hotel_id	context	Goal
0	12	1	3	0
1	12	1	1	1
2	12	1	1	0
3	93	80	1	0
4	93	21	1	0

Table2 :Dataset after cleaning

V. Encoding

We used datasets that have multiple labels in one or more columns. Labels can take the form of words or numbers. The training data is frequently labeled in words to make it understandable or human-readable.

Encoding changes that by converting the labels to a numeric representation in order to make them machine-readable. Machine learning algorithms can then make better decisions about how those labels should be used. In supervised learning, it is a crucial pre-processing step for the structured dataset.[18]

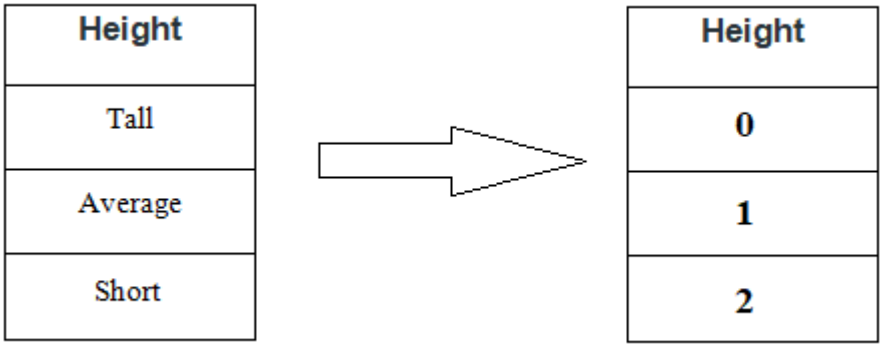

Figure 12 :Example of label encoding

We applied the label encoding on all the columns of our dataset to assure precision in calculation which are :item_id , hotel_id , context , goal

VI. Architecture of the neural network

A neural network can take different forms depending on the object of the data that it processes and according to its complexity and the method of processing the data. Architectures have their strengths and weaknesses, which makes them more in line for a given task, they can be combined to optimize the results. The choice of the architecture is therefore crucial, determining the objective to be achieved is the best way to choose the most appropriate approach according to its advantages and especially its constraints

VI.1. Autoencoder models

Auto-encoders are unsupervised learning algorithms based on artificial neural networks, which make it possible to build a new representation of a data set. Generally, this one is more compact, and presents less descriptors, which makes it possible to reduce the dimensionality of the data set. The architecture of an auto-encoder consists of two parts: the encoder and the decoder [19]

The encoder is made up of a set of layers of neurons, which process the data in order to build new so-called “encoded” representations. In turn, the layers of neurons in the decoder receive these representations and process them in an attempt to reconstruct the original data. The differences between the reconstructed data and the initial data make it possible to measure the error committed by the autoencoder. The training consists in modifying the parameters of the auto-encoder in order to reduce the reconstruction error measured on the different examples of the dataset.

The simplest form of an autoencoder is a non-recurrent forward-spreading neural network, very similar to the multilayer perception having an input layer, an output layer, as well as one, or more layers. Connecting them, but with however an output layer having the same number of nodes as the input layer, its objective being to reconstruct its inputs

The following figure schematizes a simple auto-encoder, whose encoder (encoder) processes images (inputs), in order to represent them as points in a two-dimensional space (encoded representation). then decodes this representation (decoder), in order to find the starting data (output).

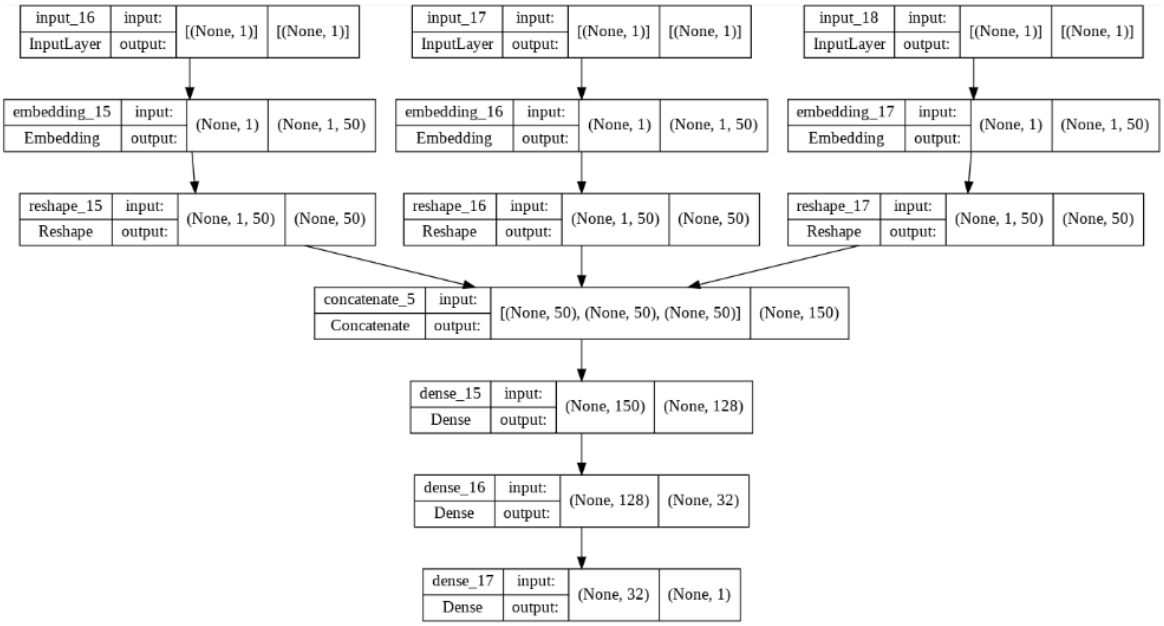

Figure 13 : Our Autoencoder models

VI.2. Activation function

Our activation function by choice is ReLU (Rectufued Linear Units)

Rectified Linear Units are a type of activation function that is often employed in deep learning models. In essence, the function returns 0 if the input is negative, and 1 if the input is positive.

The function will always return a positive number. The function is defined as follows:

$$f(x) = x^+ = \max(0, x).$$

The rectified linear unit, or ReLU, enables the deep learning model to take into account non-linearities and particular interaction effects.

VII. Machine learning

We adapted the supervised learning technique because of the existent data of user's previous interactions. Additionally, historical data can be linked to the actual state of the system. Since not all collected data records can be used as training examples, the data must be filtered. Each training data record is labeled according to its expected outcome. The resulting data set is called training data. It is processed by a machine learning technique that examines the relationship between a data record and the labeled output.

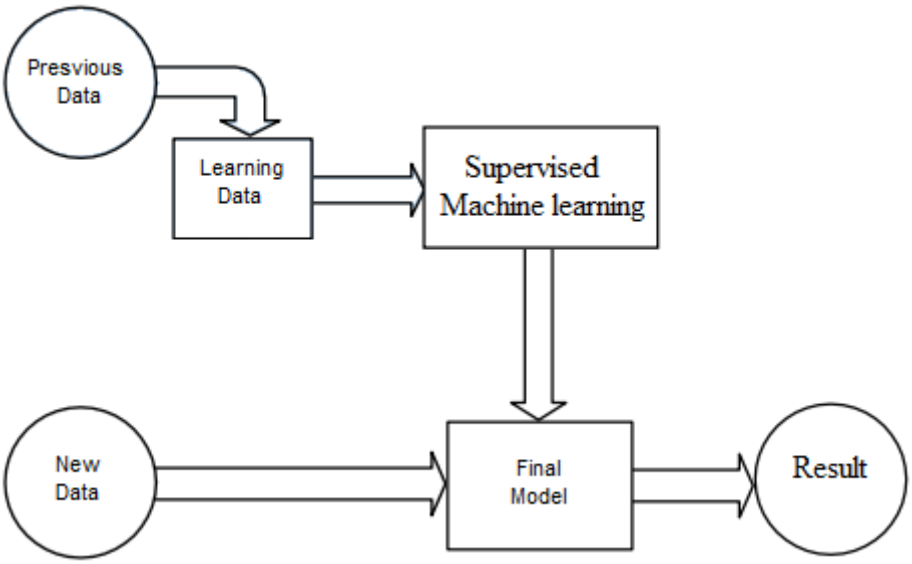

Figure 14 :Supervised machine learning model

an input configuration is associated with an output configuration. The backpropagation algorithm is an iterative gradient algorithm designed to minimize a quadratic criterionerror between the obtained output of a multilayer network and the desired output. This minimization is achieved by an adequate configuration of the weights. The error (e) is the difference between the desired value (d) for the output neuron and its value calculated by propagation (x). This error signal makes it possible to define a cost function :

$$C(W) = M[C_l(W)] = M[\sum_j e_{lj}^2(W)] \text{ avec } e_{lj} = (d_{lj} - x_{lj})$$

VIII. Recommendations

Once the model is trained, we can use it to make the predictions in a list after dropping duplicates.

After rearranging the list we get a list from the biggest value of prediction to the lowest,the algorithm generate in this step the list of the Top N hotels as a list of contextual recommendations to the target user.

IX. Conclusion

In this chapter, we have talked about hotels context and detailed the dataset that used in this project as well asthe machine learning used to construct the neural network beside the General system architecture, finally we described the recommendation part that predicted the top n list of recommendations

In the next chapter we will break down the evaluation of traditional recommender systems than evaluate our context-aware

Chapter 3: Experimentation

I. Introduction

After having detailed the design part of our system, we devote this chapter to its evaluation. First, we describe the hardware and software tools used in developing the proposed system. Then we explain the implementation of our system and we verify its effectiveness by detailing the experimentations on a hotels dataset with context items.

II. Hardware environment

The algorithm described in this chapter is developed on :

- Device name DESKTOP-JMRO5LJ
- Processor Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz 1.80 GHz
- Installed RAM 4.00 GB
- Device ID BFC513B1-0C97-4D13-9FF4-7EAE7ACD9343
- Product ID 00330-80000-00000-AA338
- System type 64-bit operating system, x64-based processor
- Pen and touch No pen or touch input is available for this display

III. Software environment

III.1.Python

Python Created by Guido van Rossum in 1991 is an interpreter, open source, multi-paradigm and multi-platform programming language most used in the field of Machine Learning, Big Data and Data Science.

And here are the libraries that we used in our system:

- **Pandas:** is a data manipulation and analysis library written for the Python programming language. In particular, it provides data structures and operations for manipulating tables values and time series.

  It is downloadable from the terminal with the command ” pip install pandas ”

- **NumPy:** is a Python library for manipulating arrays. It also includes functions for working with linear algebra, the Fourier transform, and matrices. It is an open source project that is free to use.

  It is downloadable from the terminal with the command ” pip install numpy ”

- **Scikit-learn:**is a free machine learning software library for the Python programming language. is an essential component of JPMorgan's Python machine learning toolkit. It is widely used in algorithms such as classification, predictive analytics, regression, and clustering, as well as support-vector machines.

  It is downloadable from the terminal with the command ” pip install sklearn ”

- **Matplotlib:**is a Python programming language package for graph plotting and visualization5. It can be used in conjunction with the Python scientific computing modules NumPy and SciPy. Matplotlib is freely released under a BSD-style license. Its most recent stable version (2.0.1 in 2017) is Python 3 compliant.

  It is downloadable from the terminal with the command ” pip install Matplotlib”

III.2 Keras

Keras is an open-source library of neural network components written in Python. Keras may run on top of TensorFlow, Theano, PlaidML, and other frameworks. The library was designed to be modular and social, but it began as part of a research project for the open intelligent neuro-electronic open system, or ONEIROS. The main author of Keras is François Chollet, a Google engineer who also wrote XCeption, a deep neural network model. Despite the fact that Keras was officially launched, it was not included in Google's primary TensorFlow library until 2017. Additional support for Keras integration with Microsoft Cognitive Toolkit has also been included.

III.3TensorFlow

TensorFlow is an open source library created by Google researchers to support machine learning, deep learning, and other statistical and predictive analysis workloads. It, like comparable platforms, is intended to simplify the development and deployment of advanced analytical applications for users such as data scientists, statisticians, and predictive modelers.

IV. Implementation

In this part, we will show our system coding part besides showing itin action in both coding part and their results:

IV.1.Dataset:

In this work that we presenting , we used “Expedia Hotel Data Set” the train.csv file specifically to splitting it into testing set and training set as shown below:

srch_children_cnt	srch_rm_cnt	srch_destination_id	srch_destination_type_id	is_booking	cnt	hotel_continent	hotel_country	hotel_market	hotel_cluster
0	1	8250	1	0	3	2	50	628	1
0	1	8250	1	1	1	2	50	628	1

Figure 15 :Dataset.

IV.2Processing Data:

We created an array named training to hold the dataset before removing the extra columns from the set and renaming it for the project

```
[ ]  training = pd.read_csv("train.csv", sep=',', nrows=100000)

[ ]  df = training[['user_id','hotel_cluster','cnt','is_booking']]

[ ]  df.columns =['User_id', 'Hotel_id','context', 'Goal']
     df.head(5)
```

	User_id	Hotel_id	context	Goal
0	12	1	3	0
1	12	1	1	1
2	12	1	1	0
3	93	80	1	0
4	93	21	1	0

Figure 16 :The code bloc of exploring the data.

In the second phase we encoded every column using the label encoder function and treating the goal as a float from 0.0 to 0.1

And the add x100 for the percentage possibility for ease reading

```
[ ]  user_enc = LabelEncoder()
     df['user'] = user_enc.fit_transform(df['User_id'].values)
     n_users = df['user'].nunique()

     item_enc = LabelEncoder()
     df['item'] = item_enc.fit_transform(df['Hotel_id'].values)
     n_items = df['item'].nunique()

     context_enc = LabelEncoder()
     df['cnt'] = context_enc.fit_transform(df['context'].values)
     n_cnt= df['cnt'].nunique()

     df['Goal'] = df['Goal'].values.astype(np.float32)
     minr = min(df['Goal'])
     maxr = max(df['Goal'])
     n_users, n_items

     print("minimum rating :",minr ," - with possible booking of:",minr*100,"%")
     print("maximum rating :",maxr," - with possible booking of:",maxr*100,"%")

     minimum rating : 0.0  - with possible booking of: 0.0 %
     maximum rating : 1.0  - with possible booking of: 100.0 %
```

Figure 17 :The code bloc of encoding data.

IV.3Split train/test sets:

Using sklearn we can import the train_test_split library that splits the dataset into 10% for testing and 90% for training

Having X as an input array with triplet and Y as the array for the goal

```
[ ]  X = df[['user', 'item' , 'cnt']].values
     y = df['Goal'].values
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
     X_train.shape, X_test.shape, y_train.shape, y_test.shape

     n_factors = 50
     X_train_array = [X_train[:, 0], X_train[:, 1],X_train[:, 2]]
     X_test_array = [X_test[:, 0], X_test[:, 1],X_test[:, 2]]
```

Figure 18 :The code bloc of splitting data.

IV.4 Neural Network:

The neural network was made using embeddings

Creating embeddings for the user , item and context as an input first and direct the whole embedding to the reshape and adding to the concatenate thenconnecting all the layers at last all together before compiling it

```
[ ]  # creating user embedding
     user = Input(shape=(1,))
     u = Embedding(n_users, n_factors, embeddings_initializer='he_normal',
                   embeddings_regularizer=l2(1e-6))(user)
     u = Reshape((n_factors,))(u)

     # creating item embedding path
     item = Input(shape=(1,))
     i = Embedding(n_items, n_factors, embeddings_initializer='he_normal',
                   embeddings_regularizer=l2(1e-6))(item)
     i = Reshape((n_factors,))(i)

     # creating context embedding path
     cnt = Input(shape=(1,))
     j = Embedding(n_cnt, n_factors, embeddings_initializer='he_normal',
                   embeddings_regularizer=l2(1e-6))(cnt)
     j = Reshape((n_factors,))(j)

     # concatenate features
     x = Concatenate()([u, i,j])


     # add fully-connected-layers
     fc1 = Dense(128, activation='relu')(x)
     fc2 = Dense(32, activation='relu')(fc1)
     out = Dense(1)(fc2)

     model = Model([user, item ,cnt], out)
     opt = Adam(lr=0.001)
     model.compile(optimizer = opt, loss = 'mean_absolute_error', metrics=['accuracy'])
```

Figure 19 :The code bloc of the neural network.

And here the final model of our neural network with detail :

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 1)]               0         []

input_2 (InputLayer)         [(None, 1)]               0         []

input_3 (InputLayer)         [(None, 1)]               0         []

embedding (Embedding)        (None, 1, 50)             173900    ['input_1[0][0]']

embedding_1 (Embedding)      (None, 1, 50)             5000      ['input_2[0][0]']

embedding_2 (Embedding)      (None, 1, 50)             1500      ['input_3[0][0]']

reshape (Reshape)            (None, 50)                0         ['embedding[0][0]']

reshape_1 (Reshape)          (None, 50)                0         ['embedding_1[0][0]']

reshape_2 (Reshape)          (None, 50)                0         ['embedding_2[0][0]']

concatenate (Concatenate)    (None, 150)               0         ['reshape[0][0]',
                                                                  'reshape_1[0][0]',
                                                                  'reshape_2[0][0]']

dense (Dense)                (None, 128)               19328     ['concatenate[0][0]']

dense_1 (Dense)              (None, 32)                4128      ['dense[0][0]']

dense_2 (Dense)              (None, 1)                 33        ['dense_1[0][0]']

==================================================================================================
Total params: 203,889
Trainable params: 203,889
Non-trainable params: 0
```

Figure 20 :neural network model.

IV.5 Recommending Hotels:

Listing the testing data for the model to make predictions for the user

```
[ ]  hotel_data = np.array(list(set(df.Hotel_id)))
     user = np.array([1 for i in range(len(hotel_data))])
     cnt_data = np.array([1 for i in range(len(hotel_data))])
     predictions = model.predict([user, hotel_data,cnt_data])

     predictions = np.array([a[0] for a in predictions])

     recommendID = (-predictions).argsort()[:5]

     predictions[recommendID]

     array([0.10615929, 0.1007418 , 0.1002061 , 0.06323546, 0.05768406],
           dtype=float32)
```

Figure 21 :The code bloc of predicting output.

At last we have the predictions ready to use for the user id number 16

```
rows = []
hotel = list(df.drop_duplicates(subset='Hotel_id', keep='first').Hotel_id)
names = list(df.drop_duplicates(subset='Hotel_id', keep='first').Hotel_id)

for i in range(len(hotel)):
    rows.append( dict({'name': names[i], 'Hotel_id' : hotel[i], 'User_id' : 16}))
test_data = pd.DataFrame(rows)
```

```
[ ]  for id, (score, name) in enumerate(sorted(zip(recommendID, names), reverse=True)):

         print(id+1,"- hotel ID recommended:", name ,"/ with Score of: ", round(float(score), 2) )
```

```
1 - hotel ID recommended: 41 / with Score of:  96.0
2 - hotel ID recommended: 21 / with Score of:  91.0
3 - hotel ID recommended: 92 / with Score of:  88.0
4 - hotel ID recommended: 80 / with Score of:  66.0
5 - hotel ID recommended: 1 / with Score of:  35.0
```

Figure 22 : The Recommendations code bloc.

IV.6 Evaluation:

As we mentioned earlier in the second chapter in the evaluation part of the recommendation systems, there are different evaluation protocols of which we have chosen among them the hold out protocol which divides the dataset into two parts (training, testset) according to a given percentage, we have set 90% dataset in the training part and the remaining 10% in the testset part

Among the different metrics, we used RMSE

```
[ ]  def rmse(y_test, y_train):
         return K.sqrt(K.mean(K.square(y_test - y_train)))
```

Figure 23: The code bloc of the metric used

```
trainEV = model.fit(x=X_train_array, y=y_train, batch_size=64, epochs=5,
                    verbose=1, validation_data=(X_test_array, y_test))

Epoch 1/5
1407/1407 [==============================] - 8s 5ms/step - loss: 0.2626 - accuracy: 0.9173 - val_loss: 0.2618 - val_accuracy: 0.9168
Epoch 2/5
1407/1407 [==============================] - 9s 6ms/step - loss: 0.2558 - accuracy: 0.9173 - val_loss: 0.2611 - val_accuracy: 0.9169
Epoch 3/5
1407/1407 [==============================] - 13s 9ms/step - loss: 0.2536 - accuracy: 0.9170 - val_loss: 0.2624 - val_accuracy: 0.9167
Epoch 4/5
1407/1407 [==============================] - 7s 5ms/step - loss: 0.2523 - accuracy: 0.9171 - val_loss: 0.2644 - val_accuracy: 0.9158
Epoch 5/5
1407/1407 [==============================] - 8s 5ms/step - loss: 0.2504 - accuracy: 0.9175 - val_loss: 0.2683 - val_accuracy: 0.9133
```

Figure 24 : Process of Evaluation

After launching our application we obtained the results shown on figure 25 with 5 epochs we see that our plot shows inconsistence in the accuracy which means the lack of learning process, so we stepped up the level to 50 epochs to see if it affects the process and it did like it shown in figure 26

This means that our system preformed well and its accuracy has good reliability on results

```
loss = trainEV.history['accuracy']
plt.plot(loss, color='b', label='Accuracy')
plt.title("Plot of accuracy")
plt.legend()
plt.show()
```

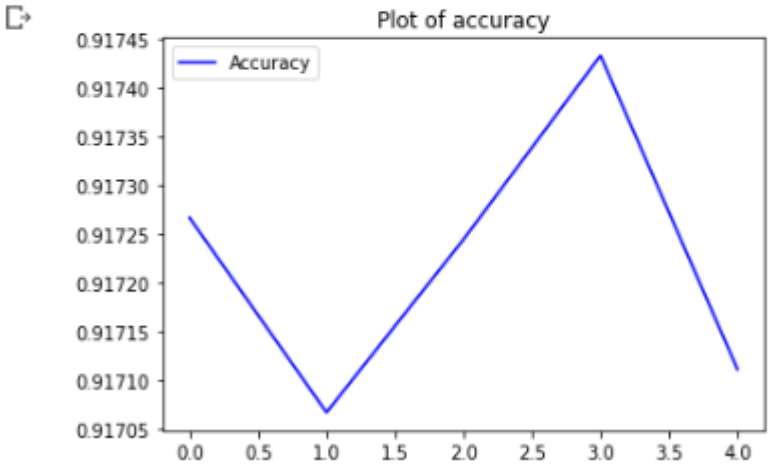

Figure 25:Result of Evaluation with 5 epochs

```
[ ]  loss = trainEV.history['accuracy']
     plt.plot(loss, color='b', label='Accuracy')
     plt.title("Plot of accuracy")
     plt.legend()
     plt.show()
```

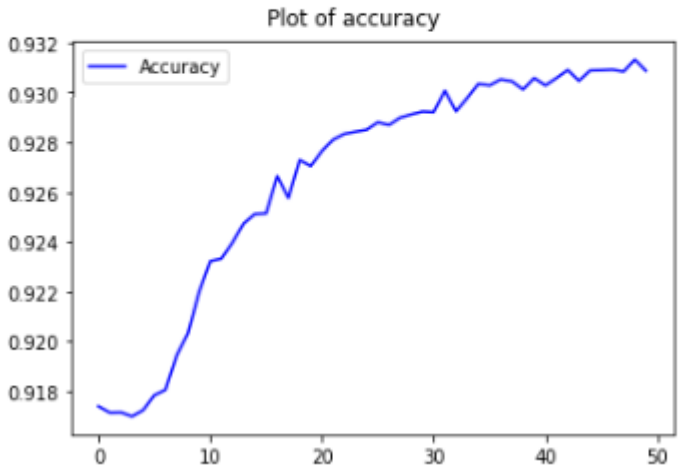

Figure 25:Result of Evaluation with 50 epochs

VI. Conclusion

In this chapter we have presented the hardware and software environment used for this project as well as an explanation of our recommender system and the whole machine learning which is based on Context-awareness, we also presented the final results of our practical case with figures and concluded the work with evaluation of the system that made a good results

General Conclusion

This thesis proposes a context-aware model for a hotel recommendation based on a supervised machine learning algorithm. The aim of the proposed method is to enhance the quality of recommendations over time and context.

This system uses contextual features as input to the neural network. It considers the user context situation and his previous interactions. After the training stage our model, can provide the best candidates items to be recommended. Finally, the system generates a Top-N recommended items by taking the best 5 hotels candidates for a target user.

The experimental results indicate that the quality of the proposed system preformed really well. However, the recommendation given by the proposed system can improve over time.

In a future work, we plan to enhance the quality of the system by introducing more machine learning techniques.

Bibliographic references

[1] LemdanRoza. Thèse de Doctorat de l'Université Paris-Saclay préparée à Ecole doctorale n◦580 Sciences et technologies de l'information et de la communication, 2016

[2] Michael J. Pazzani and Daniel Billsus.The adaptive web.chapter Content-based Recommendation Systems, pages 325-341. Springer Verlag, Berlin, Heidelberg, 2007.

[3] Burke, R. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4) :331–370, 2002.

[4] Margaritis, K. and Vozalis, E. Analysis of recommender systems' algorithms. In 6th Hellenic European Conference on Computer Mathematics its Applications (HERCMA), Athens, Greece, 2003.

[5] Bettini C., Brdiczka O., Henricksen K., Indulska J., Nicklas D., Ranganathan A., and Riboni D., A survey of context modelling and reasoning techniques. Journal of Pervasive and Mobile Computing, 2010.

[6] InBeat: JavaScript recommender system supporting sensor input and linked dataKuchařJaroslavKliegrTomáš2017· Knowledge-Based Systems

[8] Schilit, B., Hilbert, D., and Trevor, J. Context-aware communication. IEEE Wireless Commun, 9(5):46–54, 2002.

[9] Baltrunas. L. and Ricci, F. Experimental evaluation of context-dependent collaborative filtering using item splitting. User Modeling and User-Adapted Interaction, pages 1–28, 2009.

[10] [L. Baltrunas, X. Amatriain, 2009] Baltrunas. L, Amatriain. X, Towards Time-Dependant Recommendation based on Implicit Feedback, cars 2009.

[11] [Henricksen. K etIndulska. J,2006] Henricksen K. and Indulska J., Developing contextaware pervasive computing applications: Models and approach. Journal of Pervasive and Mobile Computing, 2006.

[12]Adomavicius and Tuzhilin, 2015; Fling, 2009

[13] Laouar, A. Korichi, K. Un système de recommandation pour l'assistance à la navigation sur Internet. Mémoire MASTER ACADEMIQUE Spécialité : Informatique industrielle. UNIVERSITE KASDI MERBAH OUARGLA. 2016.

[14]Erdt, M., Fernandez, A., and Rensing, C. Evaluating recommender systems for technology enhanced learning: a quantitative survey. IEEE Transactions on Learning Technologies, 8(4), (pp. 326-344), 2015.

[15] Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommenderperformance.InACMRecSys'11,WorkshoponContext-AwareRecommender Systems, 2011.

[16] Tadlaoui. M. Système de recommandation de ressources pédagogiques fondé sur les liens sociaux : formalisation et évaluation. THESE de DOCTORAT DE L'UNIVERSITE DE LYON opérée au sein de L'INSA Lyon et délivrée en partenariat international avec L'université de Tlemcen, 2018.

[17]Gabrielsson, S. &Gabrielsson, S. The use of Self-Organizing Maps in Recommender Systems, A survey of the Recommender Systems field and a presentation of a State of the Art Highly Interactive Visual Movie Rec- ommender System. Mémoire de master, Uppsala University, 2006.

[18] https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/

[19] https://dataanalyticspost.com/Lexique/auto-encodeur/

[20] https://www.lebigdata.fr/tensorflow-definition-tout-savoir

[21] https://fr.linkfang.org/wiki/Keras

[22] https://datafranca.org/wiki/PyTorch

Résumé

Grâce à l'émergence du Bigdata et son importance dans plusieurs domaines d'application, tels que les achats en ligne et le divertissement, les systèmes de recommandation sont devenu l'un des domaines les plus étudiés en intelligence artificielle. De nombreuses approches, telles que le filtrage collaboratif (CF), le filtrage basé sur le contenu (CBF) et les techniques hybrides, ont été proposées pour construire des systèmes de recommandation robustes correspondant aux intérêts et aux préférences des utilisateurs. Plusieurs approches proposées traitent l'interaction utilisateur-système comme un processus statique, ce qui dégrade l'expérience utilisateur au fil du temps. Dans ce mémoire, nous avons proposé un modèle de système de recommandation basé sur un algorithme d'apprentissage automatique pour suggérer des hôtels appropriés à l'utilisateur cible. Le modèle proposé extrait les informations contextuelles de l'utilisateur et génère les 5 meilleurs items. Les résultats expérimentaux montrent l'efficacité de notre système ainsi que son amélioration significative en capturant le comportement et le contexte de l'utilisateur pour fournir des recommandations adaptées à ses préférences.

Mots-clés : système de recommandation contextuel, apprentissage automatique, réseau de neurones.

Abstract

Due to the emergence of big data and their importance in a wide range of industries such as online shopping and entertainment, recommendation systems are one of the most widely studied domains in the field of artificial intelligence. Many approaches, such as collaborative filtering (CF), content-based filtering (CBF), and hybrid techniques, have been proposed to build robust recommendation systems that match users' interests. Several proposed approaches treat user—system interaction as a static process, which degrades user experience over time. In this thesis, we proposed a recommender system model based on a machine learning algorithm to suggest appropriate hotels to a target user. The proposed model extracts the user's contextual information and generates the top 5 recommended items. The experiment demonstrates the effectiveness of our system as well as the significant improvement through capturing user's behaviors and context to provide adapted recommendations to user preferences.

Keywords: Context-aware Recommendation System, Machine learning, neural network.

ملخص

نظرًا للأهمية المتزايدة للبيانات الضخمة وأهميتها في مجموعة واسعة من الصناعات مثل التسوق عبر الإنترنت والترفيه ، تعد أنظمة التوصية واحدة من أكثر المجالات التي تمت دراستها على نطاق واسع في مجال الذكاء الاصطناعي. تم اقتراح العديد من الأساليب ، مثل التصفية التعاونية (CF) ، والتصفية القائمة على المحتوى (CBF) ، والتقنيات الهجينة ، لبناء أنظمة توصية قوية تتوافق مع اهتمامات المستخدمين. تعامل العديد من الأساليب المقترحة تفاعل المستخدم مع النظام كعملية ثابتة ، مما يحط من تجربة المستخدم بمرور الوقت. في هذه الرسالة ، اقترحنا نظام التعلم الآلي للتوصية بالفندق. يستخرج السياق المقترح من المستخدم ويولد أهم العناصر الموصى بها. . لقد أظهر تحسنًا مثيرًا للاهتمام في التقاط سلوكيات المستخدم وتقديم التوصيات ذات الصلة باهتمامات المستخدم..

الكلمات المفتاحية: نظام التوصيات الواعي بالسياق ، التعلم الآلي ، الشبكة العصبية