REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Réseaux et Télécommunications

Par :

CHAIB Fatima Zohra

Sur le thème

Vers la prise en compte du contexte dans la recommandation des services Foursquare

Soutenu publiquement le .. / 09 / 2019 à Tiaret devant le jury composé de :

Mr KOUADRIA Abderrahman	Grade	Maitre-Assistant	Président
Mr BOUDAA Boudjemaa	Grade	Maitre de conférences	Encadreur
Mr ABID Khaled	Grade	Maitre-Assistant	Examinateur

Dedications

I dedicate this work that will never have been possible without the support unwavering and unlimited of my dear parents who keep giving me with love the necessary so that I can arrive at what I am today.
May God protect you and may success always be within reach so that I can fill you with happiness.
I dedicate it also to my twin Spirit Nour el houda who has supported me throughout the process. I will always appreciate all she has done Thank you. My love for you all can never be quantified. God bless you.
I dedicate my dissertation work to my family all my and many friends.
A special feeling of gratitude to my loving brother Mohamed and his family.

CHAIB.FZ

Thanks

I thank God Almighty my creator, my strong pillar, my source of inspiration, wisdom, knowledge and understanding. He has been the source of my strength throughout my career and on His wings only have I soared.

This is it, my thesis is over. A rather special period of my life has just ended. A rich period of experiences and scientific learning, but also human informative exchanges. If this period ended well, it is thanks to the support of the pleasant people that I have been able to meet. Given their qualities, my thanks seem modest.
I hope to give them here a share of what they deserve, while wishing that life gives me again the opportunity to express my gratitude to them.

Then I want to thank and express my gratitude specially to Mr. BOUDAA Boudjemaa. On the one hand for giving me the opportunity to participate in this project, which gave me a capacity for research and adaptation.

On the other hand, for having accepted to be my supervisor, with constant monitoring and a demonstrated interest throughout my work.

I also thank all the members of the jury for agreeing to attend at the presentation of this modest work.

Finally, I would like to thank the teachers and administrators in our university that assisted me with this project. Their excitement and willingness to provide feedback made the completion of this research an enjoyable experience.

“ We are leaving the age of information and entering the age of recommendation”

Chris Anderson in The Long Ta

Résumé

Les systèmes de recommandations (SR) sont des plateformes dont l'objectif est de recommander des services pertinents à l'utilisateur. En d'autres termes, ils tentent de prédire pour un utilisateur parfois mobile l'intérêt d'un service. Le service dans ce contexte peut être une information sur un évènement, un produit à acheter, un film à regarder, une page web à consulter ou un lieu à voir, voire une route à éviter.

L'importance des informations contextuelles a été reconnue par les chercheurs dans plusieurs domaines, y compris la recherche d'information, l'informatique ubiquitaire, le marketing et management, etc. Cependant, les travaux de recherche sur les systèmes de recommandation ont assez peu exploité les informations contextuelles. Des informations telles que le temps, la localisation, la compagnie d'autres personnes peuvent pourtant améliorer le processus de recommandation dans certains domaines tel que le tourisme (Système de recommandation sensible au contexte).

La technologie Foursquare permet de rechercher des sites (hôtels, restaurants, …) à proximité d'emplacements géographiques donnés et de connaître le nombre de personnes qui se trouvent actuellement dans ces lieux. Ces informations peuvent être utiles, par exemple, pour un touriste dans une ville. Néanmoins, elles nécessitent d'être adaptées à l'information de contexte (par exemple, informations météorologiques, pour une visite des sites en plein air). Ce travail consiste à proposer un système de recommandation sensible au contexte des services Foursquare pour un utilisateur parfois mobile.

Mots-clés

services Foursquare, Les systèmes de recommandations (SR), contexte, application mobile, tourisme …

Abstract

The recommendation system are platforms which give the user consistent services in other word it may predict to the mobile user the interest of the given service. The service in this context could be an event information, a product to buy, a film to watch, a web page to consult or a site to visit, even a rood to ovoid.

The need of contextual information is provided by the researchers in different domain, even in the information retrieve: Ubiquities information, marketing, and management. The research in the recommendation system did not exploit the contextual information. The information such as time, place, accompaniment of a person could complete the recommendation process in tourism (recommendation is quite sensible to the context)

The Foursquare technology allows the search of places (Hotel, restaurant…) near to a given geographical site, it allows also to know the number of person in the site. this information is necessary for a tourist in a town. This information adapted to the context information (example meteorological information in the open areas). In this work we have proposed a recommendation system sensible to the contextual services Foursquare for a user sometimes a mobile user.

Keywords :

Recommandation system, Contextual data, Foursquare services, Mobile user, tourisme….

SUMMARY

LIST OF FIGURES

LIST OF ABBREVIATION

RS: Recommendation Systems RS

CF: Collaborative filtering

CBF: Content-based filtering

CARS: Context-Aware Recommendation Systems

API: Application Programming Interface

TF: Term Frequency

IDF: Inverse Document Frequency

LIST OF TABLES

GENERAL INTRODUCTION

Recommendation System belongs to the class of Information Retrieval, Data Mining and Machine Learning. Recommender systems play a major role in today's e-commerce industry, tourism even in our daily live; witch expose us to make many decisions and choices related to our preferences and needs. These entities recommend to us the ideal items (the favorite restaurant, the top movies, the best tourist trip…).

During the last few decades, with the rise of Youtube, Amazon, Netflix and many other such web services, recommender systems have taken more and more place in our lives. From e-commerce (suggest to buyer’s articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys.

In a very general way, recommender systems are algorithms aimed to suggest relevant items to users (items being movies to watch, text to read, products to buy or anything else depending on industries).

It can be applied in scenarios where many users interact with many items. It can *generate meaningful recommendations to a collection of users for items or products that might interest them.* Suggestions for books on Amazon, or movies on Netflix, are real world examples of the recommender systems operating in the industry.

Recommender systems are really critical in some industries as they can generate a huge amount of income when they are efficient or also be a way to stand out significantly from competitors. As a proof of the importance of recommender systems, we can mention that, a few years ago, Netflix organizes a challenges (the “Netflix prize”) where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1 million dollars to win.

Our work aims to build a mobile application, to help enhance a visitor's experience (tourism). It's about to personalize the user's visit, according to his preferences and its context. Our work is geared towards systems of recommendation that have proven to be very effective in helping users to access the resources by which they would potentially be interested. And it is based on the content-based approach.

This manuscript is divided into 4 chapters presented as following:

Chapter I: this chapter summarize the three classical felting approaches which are the collaborative hybrid and the content based used in our proposal as well as detailed description of recommender systems then we try to listing the majority of the evaluation metrics of this kind of systems.

Chapter II: this chapter is reserved to describe the contextual recommendation system and to integrate the context into the classical recommendations systems previously mentioned in the first chapter.

Chapter III: In this chapter we describe the filtering approach used and the calculation of the similarity method named the TF-IDF by giving an example, and after that we present the proposed architecture with its description.

Chapter IV: the 4th chapter is divided in two parts, the first part represents the different plugins and software used in the implementation of our application and we explain the steps of its building in the second part. finally, we conclude it by a minimal evaluation.

The conclusion is given to summarize the projects and presents the steps needed for the achievement of the work we studied. the research we performed during the studies of the subject allow us to get a satisfactory result. Then the wished perspectives are mentioned for the evaluation of the system.

Chapter I: Classical Recommendation Systems

1. Introduction

Recommendation systems are for many of us mysterious entities that seem to guess our ideas. Just think of Netflix, who suggests movies, or Amazon, who thinks he knows which products to buy. Since their apparition, recommendation tools have been improved and have continued to enrich the user experience.

In this chapter, we begin by defining what a recommendation system is and the concepts related to it. Next, we present the three main filtering approaches that allow the recommendation, with the pros and cons of each approach as well as the used techniques, to satisfy the needs of the users. Finally, we go on to explain some evaluation metrics used to evaluate this kind of systems.

2. Brief history of the recommendation system

Recommendation systems have been used to cope with the problem of overload and the wealth of information available, particularly through the web or e-services. Recommendation systems aim to provide an active user with one or more recommendations for items that may be of interest. These recommendations may concern an article to read, a book to order, a movie to watch, a restaurant to choose, etc.

"Information Lens System" [Malone et al., 1987] can be considered as the first recommended system. At that time, the most common approach to the problem of information sharing in the environment of email was the list of distributions based on the group's interest. The first definition for filtering was also given by Malone:

"Even though the term has a literal connotation of letting things out (negative filtering: removal), we use it here in a more general sense which consists of selecting things from a broader set of possibilities (positive filtering: selection) ".

The academic literature has introduced the term collaborative filtering by the "Tapestry" system [Goldberg et al., 1992], It was developed in 1992 by the "Xerox" research center in the United States, this is a recommendation system integrated into an electronic mail application that allowed users to create permanent queries, based on annotations (tags) of users.

A few years later, a certain number of academic Recommendation systems started in 1994 and in 1995, such as the recommendation system for news articles and of films developed by "GroupLens" [Resnick and Varian, 1997] and the music recommendation "Ringo" by [Maes and Shardanand, 1995]. These two systems are also based on collaborative filtering, books [Resnick and Varian, 1997], videos, movies, web pages, Usenet news articles and Internet links.

When, Resnick and Varian published their paper [Resnick and Varian1997] named "Recommender Systems" in which they argue that collaborative filtering is not the only one approach to make RSs like Fab [Balabanović and Shoham1997] and PHOAKS [Terveen et al. 1997]. Since then, multiple syntheses have followed each other to describe the evolution of the domain of the recommendation [Maes et al. 1999, Tintarev and Masthoff, 2007, Wei et al. 2007, Almazro et al.2010, Prasad 2012, Lü et al 2012].

Subsequently, with the rise of the Internet and web applications, there has been a craze for the RSs that have developed.

3. Recommendation systems

3.1. What is a recommendation system?

The recommendation system is a specific form of filtering information that aims to provide the user with elements that may interest him, based on his preferences and behavior. The system tries to predict the user's estimation of something to suggest him the most relevant and suitable things for him (As shown in Figure1**)**.

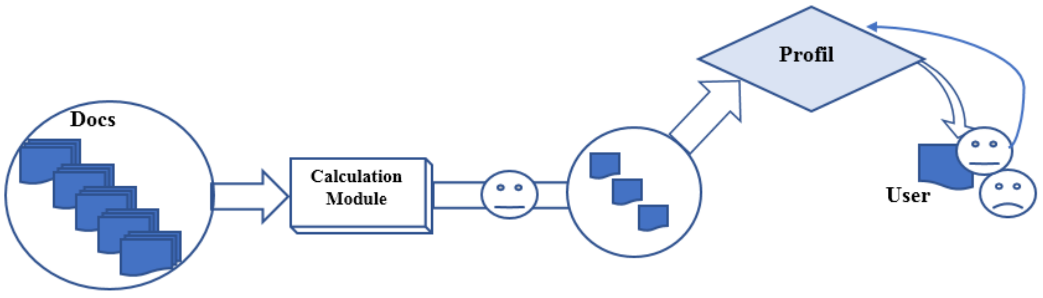


Figure 1: General scheme of information filtering

The following three parts constitute a recommendation system:

• Producers: These are the ones that will make recommendations; they will "provide" the data for.

• The calculation module: which is the algorithm itself. On the input there is all the data and the demands and on the exit the different recommendations.

• The consumer: This is the one who asks for the recommendation.

3.1.1 Recommended system requirements

The main requirements of the recommendation systems can be seen as important parameters to evaluate the system. Namely the parameters are:

- Ability of adapting to many data sources where industrial systems must be able to deploy to different services with different types of data.
- Robustness to noisy or possibly corrupted data, as some data sources may be more or less reliable.
- Ability to scale because operating systems must be able to cope with large data.
- The reactivity of the system, because the response time and online learning problems are also crucial.
- Relationship of trust, based on transparency and explanation: the transparency of a

recommendation system is an important factor in accepting the recommendations.

➢ The management of "long tail", i.e. the management of all items that are not often bought / seen but that we want to promote in some cases.

3.1.2. Major problems of recommender system

Despite their growing popularity, recommendation systems have suffered some problems and failures. In this section, we try to identify some of them:

• Cold start:

concerns both new users and new items that are introduced into the system. A new user who has not noticed any item cannot receive a recommendation since the system does not know his tastes. This problem is known as (user cold start) [Ricci et al.,2011]. A solution to this problem is to explicitly ask him to note a certain number of items. Other solutions consist in recommending at the beginning the most popular items or even random recommendations. This Cold start problem also arises when adding a new item. It cannot be recommended until it has been noted by a certain number of users.

• Sparsity:

The number of candidate items for recommendation is often huge and users only notice a small subset available items. As a result, the matrix of notes is a hollow matrix with a missing value rate of up to 95% of the total values [Papagelis et al., 2005]. Collaborative filtering systems have difficulties in this case, the number of notes to predict being much higher to the number of notes already known. The problem of parsimony can be reduced by using model approaches that reduce the size of the matrix of notes.

• The problem of gray sheep:

Users who have Strange tastes (which vary from standard or out of the ordinary) will not have many neighbors. It will be difficult to do relevant recommendations for such users [Ghazanfar and Prügel-Bennett, 2014].

• Limit of content analysis:

A natural limit of the content-based recommendation is the need to have a varied representation and rich content of the items, which is not always the case. The accuracy of recommendations is related to the amount of information available to the system to discriminate items appreciated from those not appreciated by the user [Lops et al., 2011]. Unlike collaborative filtering that can handle everything type of items without any information on their content, the content-based approach can only handle items with content that can be analysis.

• Over-specialization:

The system can only recommend items that are similar to the user profile. The user cannot therefore, only receive recommendations close to the items he has noted or observed in the past [Adomavicius and Tuzhilin, 2005]. However, the diversity of recommendations is often appreciated and proves to be an evaluation criterion important of the recommendation systems [Yu et al., 2009].

Ideally, the user must receive relevant and diversified recommendations. For example, it is not worth recommending all the songs of Jacques Brel to a user who liked one of his songs.

• Critical mass.

In order to form better communities, the system requires a sufficient number of evaluations in common between users to compare them. For example, we cannot conclude that two people are in the same community if they have only one assessment in common. The finding today is that despite the huge size of all documents, purchases, etc., in systems, the number of assessments shared between users may be low.

• Principle of induction.

Recommendation systems are based on the premise that a user who has exhibited behavior in the past will tend to exhibit similar behavior in the future. However, this principle is not necessarily true in the real context. Indeed, a user can completely change domain of interest or have several. To cope with this problem, interest drift or Context Shift techniques have emerged.

• Centralized or distributed.

Recommendation systems, such as Amazon, are typically based on a centralized architecture. The centralized recommendation engine is used to save the user profile and the calculation of recommendations on a central server. However, despite their popularity, centralized Recommendation systems suffer from several problems: cost, robustness, security, portability, and so on. One of the solutions to these problems is to distribute the system. It is believed that a distributed recommendation system could be designed to take advantage of the computing power available on users' computers.

• Security and credibility.

Recommendation systems cannot prevent acts of deception. It becomes easy to forge a new identity and to engage in vandalism. It is more difficult to control the identity of users and penalize malicious behavior. Therefore, it is essential to have means allowing each user to decide which users and which contents to trust.

• Protection of private life.

Another problem with the recommendation systems is the protection of sensitive information that constitutes the user's profile (personal information, interests, tastes, habits, etc.). Given the nature of the information, these systems must provide such protection. Thus, means, to preserve the anonymity of users and encrypt the data transmitted, are necessary.

3.3. Recommendation Systems functionalities

The main functionalities that can be mentioned and cited in the case of the recommendation systems are:

Essential function	**Description**	**Main function for recommendation**
Decision support	Let 1 Article I, a user u wants to know if he appreciates i	Score prediction
Comparison support	Let n article, a user u wants to know witch article to choose	Rang prediction
Exploration support	Let article 1, a user wants to know the K similar articles	Article to article recommendation
Discovering support	Taking into account a large articles catalog, the user u wants to find the k new interesting articles	Personalized recommendation

Table I: features of recommendation systems

3.4. How to build a recommendation system?

In order to make customized recommendations, RS needs to know what the preferences of each user are. It then tries to obtain the information necessary for construction User Profiles. In particular, it exploits the effects left by the users themselves. It collects the remaining effects explicitly or implicitly [Oard and Kim1998, De Gemmis et al 2009]. Explicit traces are provided by the user [Ricci and al.2011]. The implicit effects are collected by drawing the user's automatic procedures on the move [Oard and Kim1998, Castagnos2008, Brun et al.2011]. Several studies have been conducted on methods of collection and usefulness [Amatriain et al.2009^b, O'Donovan and Smyth2005]. They often try to distinguish the most useful users from the RS. These users, named experts, can be assigned to register items [Amatriain et al. 2009a], or have users who reside a lot of items [Jones2010], or can be extracted using automated learning methods Esslimani2010, Esslimani et al. In [Jones2010], the author explains that the notes are more suitable for experts, while the implicit notes are more suitable for other profiles. In [Amatriain et al. 2009a, Esslimani2010], the authors try to use only expert data, and consider that non-expert data contains a lot of noise.

3.4.1. How to classify systems of different recommendations?

Several factors are taken into consideration in order to classify the recommendation systems.

1. Know the user (i.e. his profile according to his taste).

2. Locate the user from others (concept classes or Networks of users).

3. Know the elements of the recommendation.

4. Knowledge of different categories of items to recommend.

These factors produce different types of recommendations that are most commonly used. Craft is content-based filtering and collaborative filtering. These two methods are presented in addition to their hybridization.

- **Classical classification:**
  Figure 2 represents the classification of [Adomavicius and Tuzhilin, 2005] which is recognized by three types of filtering; collaborative filtering (CF), content-based filtering
  (CBF) and hybrid filtering.

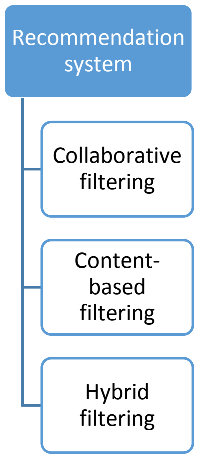


Figure 2: Classical classification of RS

- **The Classification of [Su et al, 2009]:**
  It is used in collaboration systems. They propose a sub-classification that includes Hybrid techniques and classify them in collaborative methods hybrids. [Su et al, 2009] classifies collaborative filtering into three categories:
  - CF approaches based on memory: for K-nearest neighbors.
  - Model-based CF approaches encompassing a variety of techniques such as: clustering, Bayesian networks, factoring of matrices, Markov's decision processes.

• Hybrid CF that combines a CF recommendation technique with one or more other methods (As shown in Figure 3)

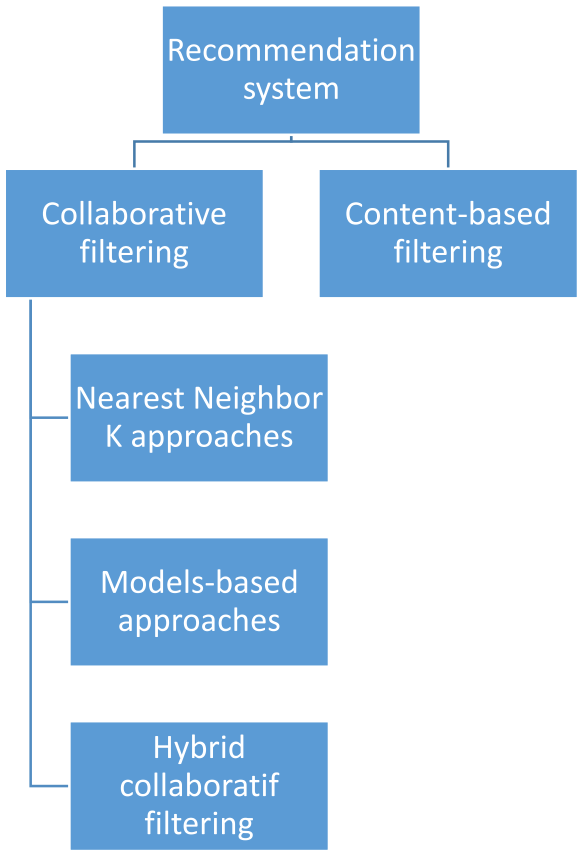


Figure 3: Classification of RSs [Su & Khoshgoftaar 2009]

❖ The classification of [Rao and Talwar, 2008]:

It is a classification according to the source of information used.

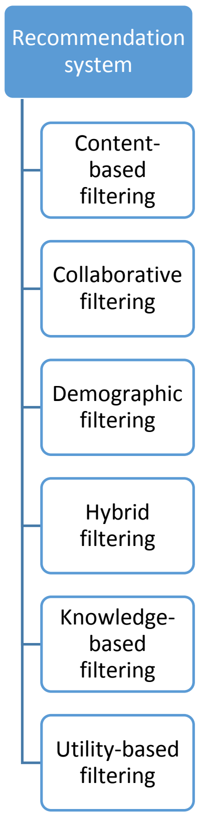


Figure 4: Classification of Rag & Talwar 2008

❖ Figure 5 represents another classification proposed by Charif Alchiekh Haydar[1]:

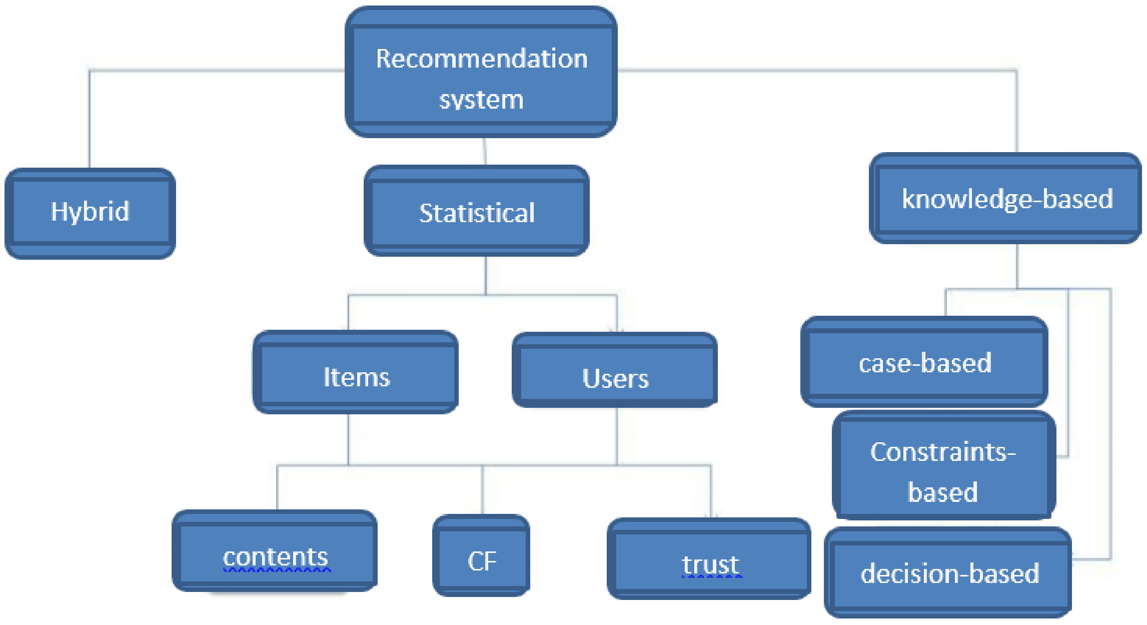


Figure 5: Classification of RS (2014)

[1] http://docnum.univ-lorraine.fr/public/DDOC_T_2014_0203_ALCHIEKH_HAYDAR.pdf

4. What are the filtering approaches?

There are basically three important types of recommendation engines:

- Collaborative filtering
- Content-Based Filtering
- Hybrid Recommendation Systems

4.1 Collaborative filtering

This filtering method is usually based on collecting and analyzing information on user’s behaviors, their activities or preferences and predicting what they will like based on the similarity with other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and thus it is capable of accurately recommending complex items such as movies without requiring an “understanding” of the item itself. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. For example, if a person A likes item 1, 2, 3 and be like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.

Further, there are several types of collaborative filtering algorithms:

4.1.1. User-User Collaborative Filtering

Here, we try to search for lookalike customers and offer products based on what his/her lookalike has chosen. This algorithm is very effective but takes a lot of time and resources. This type of filtering **(**Figure 6) requires computing every customer pair information which takes time. So, for big base platforms, this algorithm is hard to put in place.

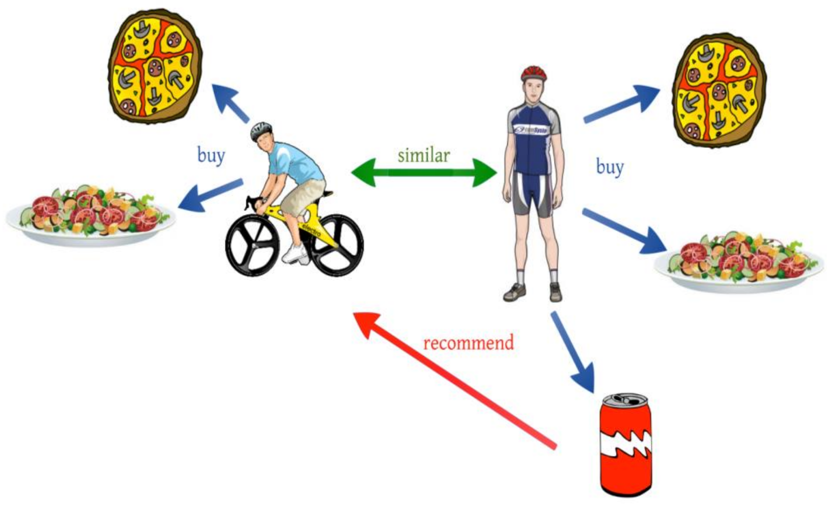


Figure 6: User-User Collaborative Filtering

4.1.2. Item-Item Collaborative Filtering

It is very similar to the previous algorithm, but instead of finding a customer lookalike, we try finding item look alike. Once we have item lookalike matrix, we can easily recommend alike items to a customer who has purchased any item from the store. This algorithm (Figure 7) requires far fewer resources than a user-user **collaborative filtering.** Hence, for a new customer, the algorithm takes far lesser time than user-user collaborates as we don't need all similarity scores between customers. Amazon uses this approach in its recommendation engine to show related products which boost sales.

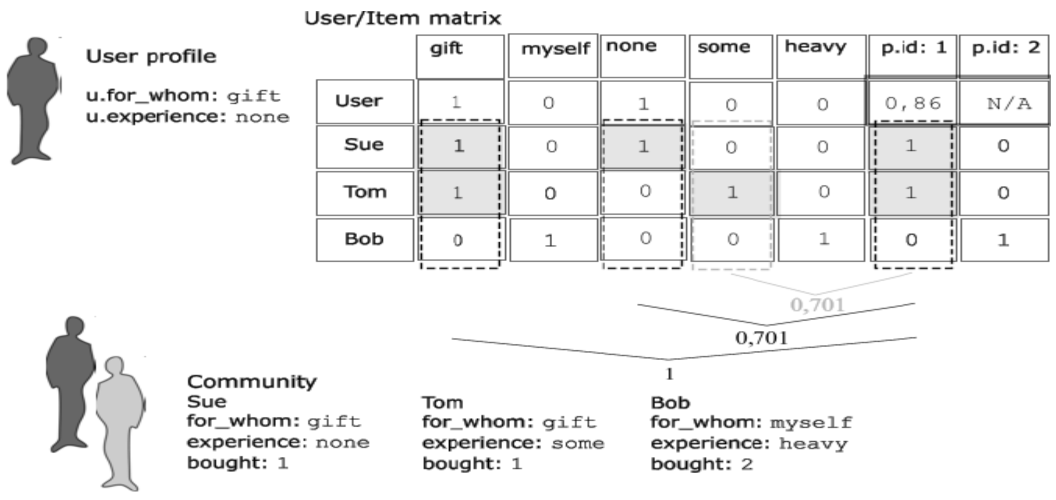


Figure 7: Item-Item Collaborative Filtering[2]

- **Other simpler algorithms:** There are other approaches like market basket analysis, which generally do not have high predictive power than the algorithms described above.

4.2. Content-based filtering

These filtering methods are based on the description of an item and a profile of the user's preferred choices. In a content-based recommendation system, keywords are used to describe the items; besides, a user profile is built to state the type of item this user likes. In other words, the algorithms try to recommend products which are similar to the ones that a user has liked in the past. The idea of content-based filtering is that if you like an item, you will also like a 'similar' item, for example, when we are recommending the same kind of item like a movie or song recommendation. This approach has its roots in information retrieval and information filtering research.

A major issue with content-based filtering is whether the system is able to learn user preferences from user actions about one content source and replicate them across other different content types. When the system is limited to recommending the content of the same type as the user is already using, the value from the recommendation system is significantly less when other content types from other services can be recommended. For example, recommending news articles based on browsing of news is useful,

[2] https://www.researchgate.net/figure/Example-item-based-collaborativfiltering_fig5_220116257

but wouldn't it be much more useful when music, videos from different services can be recommended based on the news browsing (see Figure 8).

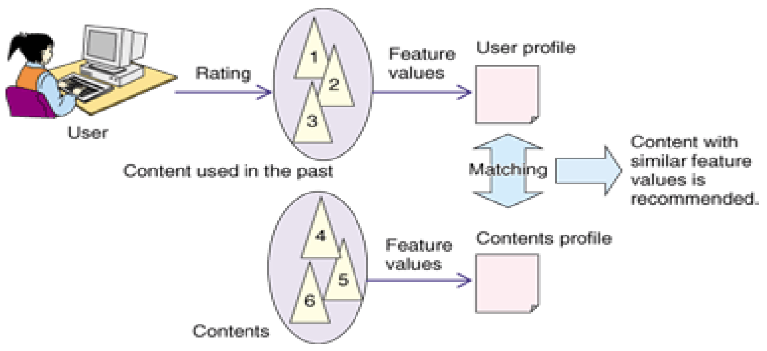


Figure 8: Content-based Filtering[3]

4.3. Hybrid Recommendation systems

A recommendation system is called hybrid when it combines two or more different recommendation approaches. The content-based recommendation of the content and the collaborative recommendation have often been considered complementary [Adomavicius and Tuzhilin, 2005]. Content-based approaches have the advantage of being able to recommend new items not yet evaluated by a user, while collaborative filtering can only recommend an item if it has been noted by a number of users before. Approaches content-based need to have attributes of the items, in addition to a step of analysis to extract and represent them, while collaborative filtering does not require no access to the content of the items to be able to make the recommendation. He relies solely on the matrix of user notes for the different items.

Hybridization of these two techniques, in order to address the shortcomings of each technique used alone and take advantage of their strengths, was the subject of several research work. The FAB system [Balabanović and Shoham, 1997] is one of first hybrid recommendation systems. It combines collaborative filtering and a content-based approach to deal with both the startup cold problem for items and over-specialization. In this system, two criteria must be satisfied to recommend an item: its content should be similar to the profile of the user, and it must be appreciated by the nearest neighbors.

There are many ways to do hybridization and no consensus has been defined by the research community. However, Burke [Burke, 2002] identified seven different ways to hybridize:

• Weighted: the score or prediction obtained by each of the two techniques is combined into a single result.

[3] http://findoutyourfavorite.blogspot.com/2012/04/content-based-filtering.html

• By selection (Switching): the system switches between the two techniques of recommendation depending on the situation.

• Mixed: lists of recommendations from both techniques are merged into one list.

• By combination of properties (Feature combination): the data both techniques are combined and transmitted to a single algorithm of recommendation.

• By increasing properties (Feature increase): the result of a technique is used as the input of the other technique.

• In cascade: In this type of hybridization, a recommendation technique is used to produce a first ranking of candidate items and a second technique then refines the list of recommendations.

• By setting a meta level: This method is analogous to the method by increasing properties but it is the learned model that is used in entry of the second technique and not the result list of recommendations. (see Figure 9)

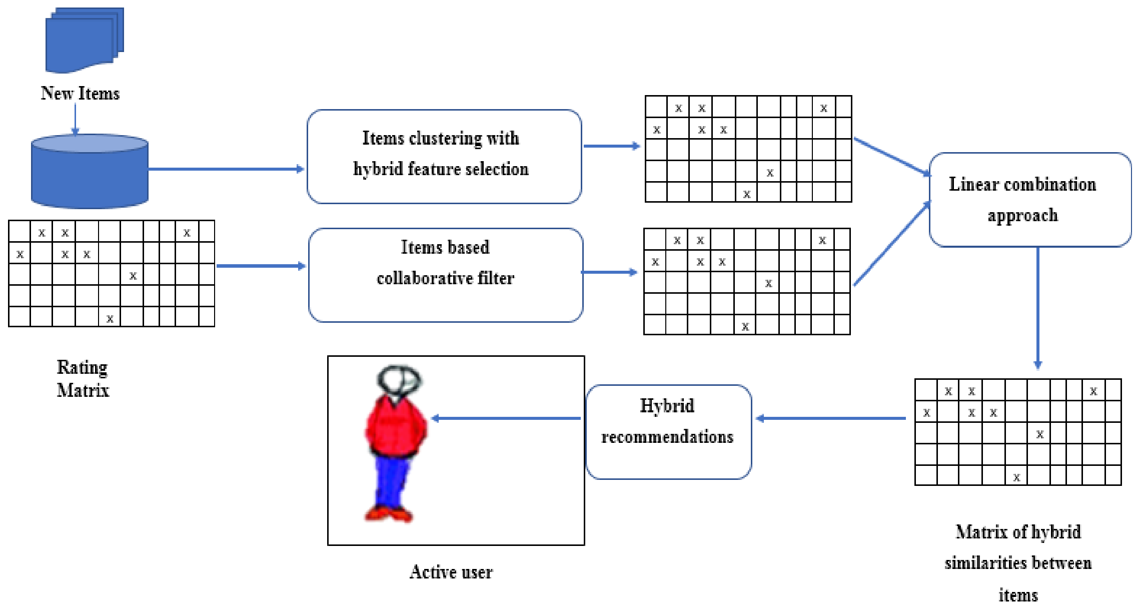

New Items
Rating Matrix
Items clustering with hybrid feature selection
Items based collaborative filter
Linear combination approach
Hybrid recommendations
Active user
Matrix of hybrid similarities between items

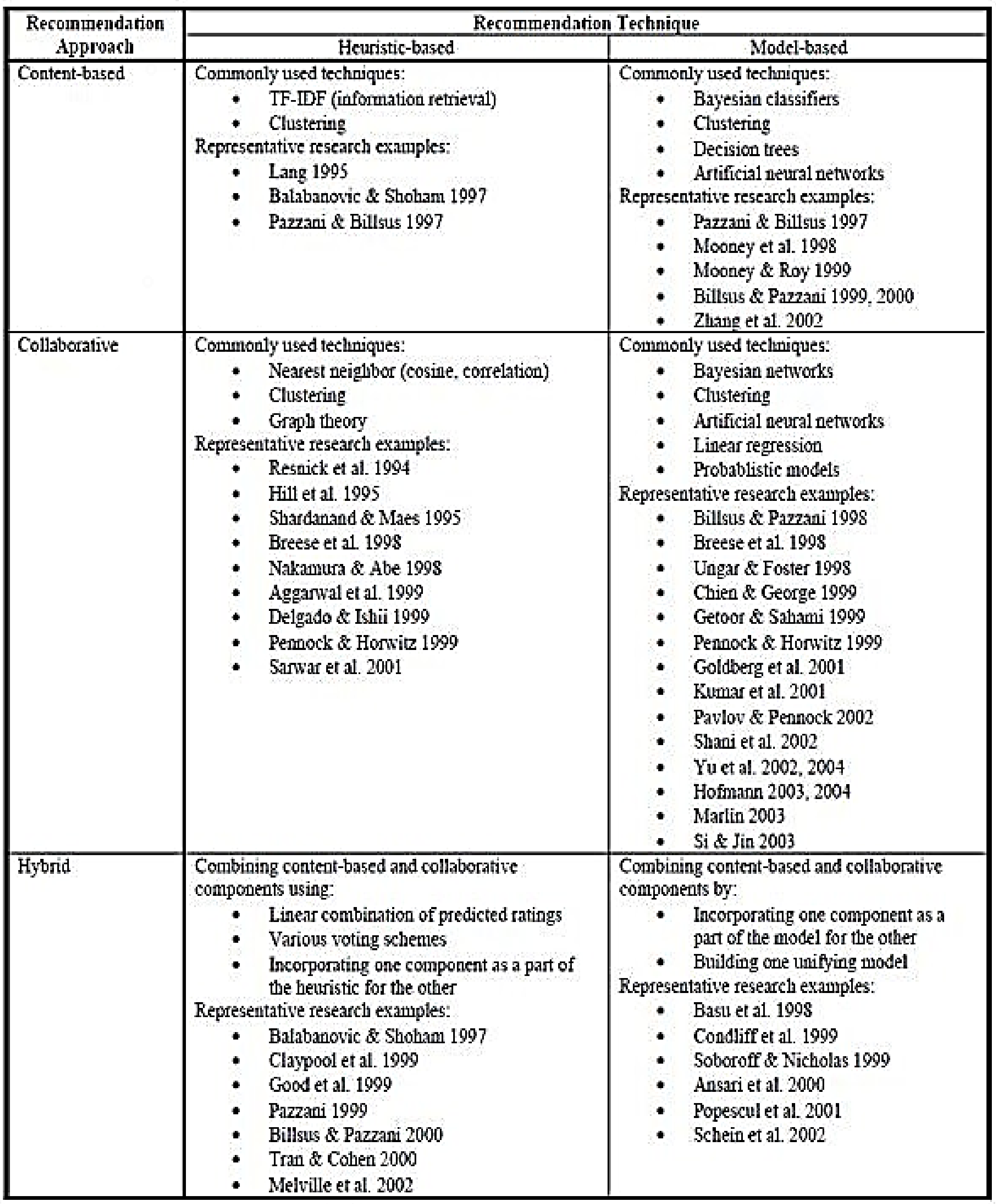
Figure 9: Hybrid Recommendation systems[4]

5. The main recommendation techniques

In the area of the recommendation, several techniques have been used by researchers, including heuristics based and model based. Among these techniques we can mention: Baesian networks, clustering, decision trees, etc. (see ref [S. Castagnos. « Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information », thèse de doctorat de l'université Nancy 2. Novembre

[4] https://link.springer.com/chapter/10.1007/978-3-319-25017-5_27

2008.]). Table IV summarizes the different recommendation techniques used by the research community in the three recommendation approaches (BCF, CF and hybrid).



Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probablistic models Representative research examples: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Table II: Classification of recommendation techniques

6. Advantages and strengths of recommendation system

• Users Autonomy:

The content-based recommendation techniques treat each user independently. Thus, only the user's own ratings are taken into account to build his user profile and make the recommendation, which is not the case for approaches using collaborative filtering.

• Immediate consideration of a new item:

Filtering based on the content can recommend newly introduced items in the database

before they receive an assessment from a user, at least contrary to collaborative approaches that cannot recommend an item only if it has been previously evaluated by a group of users. Authors [Ekstrand et al., 2011] demonstrated that an item must have received at least 20 notes for collaborative filtering to recommend it so relevant.

• Surprise effect (serendipity):

The effect of surprise is identified when the user can receive a relevant recommendation that he would not have found alone, but is often desirable. Filtering-based algorithms Collaboration generally makes it possible

For example, if a user u is close to a user v. because the user v only looks at comedies, and by the same way he appreciates a film of a different kind. This film can be recommended to u because of its proximity to v.

• Not required for domain knowledge:

Collaborative filtering-based recommendation systems do not require knowledge of items. These methods can recommend items without needing to understand their meaning or have their attributes. The recommendation is based only on the notes given to the item

7. Evaluation metrics for recommendation algorithms

The quality of a recommendation algorithm can be evaluated using different types of measurement.

We will examine how we can evaluate the performance of the RS to ensure its ability to meet the needs that led to its creation. The choice of a measure, must be depending on the type of data to be processed, and the interests of users [Herlocker et al.2004].

As the domain of the recommendation derives from the field of information retrieval, so it is often normal to use measures of evaluation of the search for information [Voorhees2002]. Some of these measures have been adjusted to the needs of the area of the recommendation. The performance evaluation of RS in the literature is often limited to calculating the prediction accuracy [Herlocker et al. 2004]. Precision measures, in general, the difference between the values of the notes predicted by the recommendation system and the values actually provided by users.

The evaluation of recommendation systems usually follows one of three methods: offline, user sample studies, or online evaluation [Shani and Gunawardana2011].

Offline evaluation is the easiest to do and the least risky. This is basically divide the available data into two parts, the learning part and the test part, before using the learning part to predict the test part. This type of evaluation does not pose of user leakage problem, so we can take the risk to test even approaches very fluctuating. It allows to easily integrate a large mass of users.

It is not very sensitive to potential changes in the behavior of the user. It tries to recopy a behavior that the user had while the recommendation system did not intervene to advise the user, but she is not able to measure the impact such an intervention [Shani and Gunawardana2011].

The second method (sample studies) is to recruit a group of volunteers, which are required to perform specific tasks using the RS, to monitor and to record their behavior during the experiment. Then we can also ask questions to participants about their impressions of the experiment and the RS. This type of test can answer a wide range of questions. It tracks the behavior of a user during his interaction with the RS, and to observe if he has influenced the behavior of the user. The questionnaire also makes it possible to collect data qualitative data to explain the quantitative results. This type of evaluation is expensive. It is not always easy to recruit a sufficient number user, sometimes motivated by rewards or compensation. The number of participants is often limited, and conclusions cannot be drawn about mass users. Moreover, because of the time constraints of the participants, we cannot ask for excessively long tests. Nevertheless, each scenario must be repeated several times to ensure the reliability of the result [Shani and Gunawardana 2011].

The last type is the online evaluation. It is applied on the real users of the system in real time [Kohavi et al.2009]. This test can be a simple comparison of turnover before and after the application of the RS. It is applied to a sample of users (drawn at random), their reactions are observed, and compared with those of the rest of the population. This guy evaluation involves risks. Indeed, one can lose a user if the system recommends irrelevant items. For this reason, it is recommended to proceed to a risk-free evaluation to guarantee a minimum quality of recommendation [Shani and Gunawardana2011]. The characteristics of the tests make the first type of evaluation (offline) the most appropriate for our work, because this work focuses on behavioral phenomena in a large number of users. In order for the results to emerge, it is necessary to integrate the behavior a considerable number of users. A sample-based test will not be suitable for conclude with this kind of results. In the next section we will detail the offline tests by showing the quality measures used to validate our results.

7.1. Evaluation by offline tests

It is necessary to begin by distinguishing two strategies according to which RHs communicate their results to users. In the first, the system answers the question: will the user appreciate this item? These systems seek to predict all the missing values of the matrix of notes, and display their predicted value next to the item when viewed by the user, Movielens is an example of this strategy. In the second, the RS answers the following question: What are the items that the user will appreciate? Systems that follow this strategy output an ordered list of best items that the user will appreciate, the numerical value of the predicted note is not a priority, the bottom line is that the list contains relevant items. The measurement must be consistent with the strategy followed by the system. Two classes evaluation measures are used: the class of predictive accuracy measures, and the class Precision measures of ranking [Herlocker et al. 2004].

7.1.1 Predictive Precision Measurements

The purpose of these measurements is to evaluate the accuracy of the prediction. The best known measure in this category is the absolute average error (MAE) [Shardanand and Maes1995], this is a statistical measure that is based on the average of the differences between each predicted value and its actual value:

$$MAE = \frac{\sum_{i=1}^{N} |Pi - ai|}{N}$$

This is good for overall assessment of RS's ability to predict ratings users [Herlocker et al. 1999, Melville et al.2002, Vozalis and Margaritis2003, Massa Avesani2004, Xue et al.2005, Amatriain et al.2009a]. A large number of variants of this measure have been proposed to further evaluate accuracy [Andersen and FenandezLuna2012]:

	The item is selected	**The item is not selected**	**Summe**
Pertinent	**Nb$_{ps}$**	**Nb$_{pr}$**	**Nb$_{p}$**
Non Pertinent	**Nb$_{ms}$**	**Nb$_{mr}$**	**Nb$_{m}$**
	Nb$_{s}$	**Nb$_{r}$**	**Nb**

Table III: F-measure

Absolute Average Error(NMAE)

This is the standard version of MAE. She is used primarily to compare two models where rating scales are not equal.

$$NMAE = \frac{MAE}{v_{max} - v_{min}}$$

where v_{max} and v_{min} are respectively the maximum and minimum value of notation.

Absolute average error per user(UMAE)

This measure is very important for estimate the satisfaction per user. A local average of the predicted values is calculated per user, after calculating the overall average. Suppose a RS predicts 10 notes of which 8 of good quality. It is possible that the 8 values are for a single user (which notes a lot), and the other two values are for two other users. In this case, this system

$$UMAE = \frac{\sum_{j=1}^{N} \frac{\sum_{i=1}^{N} j|P_{ij} - a_{ij}|}{Nj}}{N}$$

will have a good score MAE while it satisfies only one user out of three. UMAE is used to detect the fluctuation of the RS, and measures its ability to satisfy the maximum of its users; where N is the total number of users, Nj is the number of predicted notes for the user j.

Absolute average error for high values (HMAE)

The versions examined for the moment measure the ability of the system to predict values. HMAE considers errors prediction based on their impact on the recommendation. She tolerates mistakes in low notes, but not in high notes. The equation of HMAE is identical than that of MAE, except that it applies only when the actual scores are high (for we a high score is 4 or 5 in a rating scale between 1 and 5).

7.1.2. Rating accuracy measures

These measures do not take into account the value that the RS predicts, but they consider its whether to recommend an item. The RS is rewarded for the right decisions (integration relevant items in its list of recommendations), and penalized for the bad (the inclusion of irrelevant items in the list, or the absence of a relevant item from the list). The objective is to measure the frequency of good and bad judgments carried by the system recommendation for the items. F-measure is the most commonly used measure in this category [Herlocker et al.2002]. She was first proposed by Cleverdon in 1968 for information retrieval systems. It was first applied for RH by [Basu et al.1998]. It consists of two values: recall and precision. We will explain this measure with the help of (Table III).

The recall is the ratio between the number of relevant items selected and the total number relevant items. Formally:

$$\mathbf{R = Nbps / Nbp}$$

Precision is the ratio between the number of relevant items selected and the total number of selected items. Formally:

$$\mathbf{P = Nbps / Nbs}$$

The F-measure is a compromise between recall and precision:

$$F = \frac{2 * R * P}{R + P}$$

Several other approximations of recall and precision exist [Herlocker et al.2004, Sarwar et al. 2000].

7.1.3. Precision measurements of the ranking:

These measures are useful in the case where the RS must predict the order of the list of recommended items. The relevance of an item is not a binary value but conditioned by its position in the list of recommendations. The system can be penalized by these measures even if the list contains only relevant items, if these items are misdirected.

In this category, we are interested in the Middle Reciprocal Rank (MRR). It is a measure of quality used to evaluate RSs which must give as an output ordered list with only one relevant element. The reciprocal rank (RR) of a list is equal to 1 = r, where r is the rank given by the evaluated approach to the relevant element. The average rank reciprocal is the average RR value of all lists. The value of this measurement varies between 0 and 1, where 1 is the best precision score. This measurement is appropriate for some applications question-answer systems, where the RS must make the most accurate answer to a given question.

7.1.4. Coverage

Coverage in RS can have two meanings: coverage of the item catalog, and user coverage. In the first case (item coverage), the percentage of items that the approach is able to recommend to users is calculated for the set of available items [Lü et al 2012]. In the second case (user coverage), the percentage of users for whom the RS is able to generate recommendations.

8. Conclusion

In this chapter, we first introduced the notion of recommendation systems, detailing the three most used approaches, namely, the content-based filtering, collaborative filtering and hybrid filtering. And we have given the mathematical expression to evaluate all the systems. We have given as well he advantages and the limits.

Chapter II: Context-Aware Recommendation Systems

1. Introduction

Recommender systems have recently been singled out as a fascinating area of research, owing to the technological progress in mobile devices, such as smartphones and tablets, as well as to the rapid growth of social networking. In this respect, the main purpose of recommender systems is to suggest items that help users to make decisions from a large number of possible actions such as, what place to visit, what movie to watch, or which friend to add to a social network system. In mobile environment, many personal, social and environmental contextual factors can be integrated into the recommendation process in order to provide the correct recommendation to a special user, at the perfect moment, in the appropriate location based on his/her current activity, preferences and past behavior.

This chapter will tackle a definition of context-awareness and how can to integrate it in the classical recommendation systems.

2. Context

The notion of context is quite universal. It designates all the elements that can influence the understanding of a particular situation. This description is then used to better understand the environment.

2.1. Context definition

In artificial intelligence, Brézillon [Brézillon, 2002] defines the context by: "The context is what does not intervene directly in the resolution of a problem but constrains its resolution"

Dey [Dey and al., 2001] defines by context: "Any information that can be used to characterize the situation of an entity (person, physical object or computer). And more generally any element that can influence the behaviour of an application" [Norha and al,2018]

In the context of ubiquitous computing, we can cite some examples of entities and their associated contexts:

- The used terminology: network connection bandwidth, memory availability, processor load, screen size, and user interaction modes
- The user: user's identity, geographical location
- The room: level of light or noise, temperature.

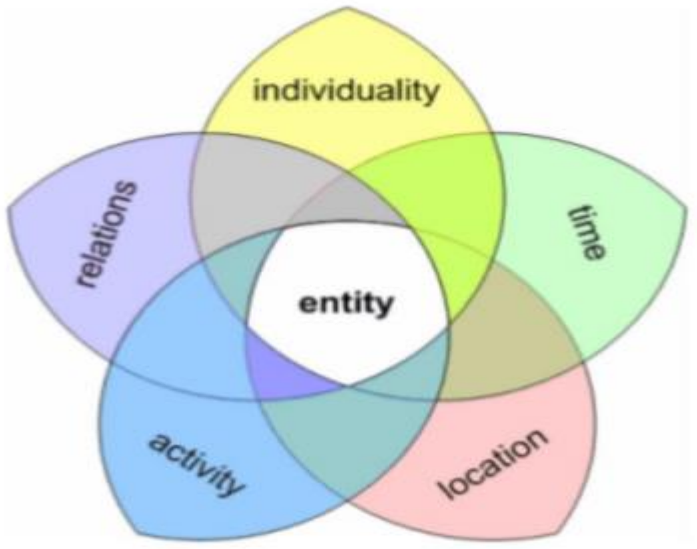


Figure 10: The different elements of context [Zimmermann and al., 2007]

Zimmermann defines context as any information that can be used to characterize the situation of an entity. The context elements include information about activity, location, time, and relations as shown in Figure 10 [Zimmermann and al., 2007].

2.1.1. Context categories

[Villegas and al,2010] characterize context along five general categories: individual, location, time, activity, and relational. Other characterizations, which can be instantiated from these general categories, have been proposed for domain specific CARS (e.g., the one proposed by Verbert and al., in [Villegas and al,2012]or CARS in the learning realm). To identify the context types exploited by the CARS, we based on the classification of context information proposed by Villegas and al., which is summarized as follows:

- **Individual context:**

Corresponds to information observed from independent entities (e.g., users or items) that may share common features. This category can be sub-classified into *natural, human, artificial*, or *groups of entities. Natural context* represents characteristics of living and non-living entities that occur naturally, that is, without human intervention (e.g. weather information). *Human context* describes user behavior and preferences (e.g., user payment preferences). *Artificial context* describes entities that result from human actions or technical processes (e.g., hardware and software configurations used in e- commerce platforms). The last subcategory, *groups of entities,* concerns groups of independent subjects that share common features, and that might relate each other (e.g., preferences of users in the user's social network).

- **Location context:**

Refers to the place associated with an entity's activity (e.g., the city where a user lives). This category is sub-classified as *physical* (e.g., the coordinates of the user's location, a movie theater's address, or the directions to reach the movie theater from the costumer's current location), and *virtual* (e.g., the IP address of a computer that is located within a network).

• **Time context:**

Corresponds to information such as time of the day, current time, day of the week, and season of the year. Time context can be categorized as *definite* and *indefinite*. *Definite* context indicates time frames with specific beginning and end points. *Indefinite* context refers to recurrent events that occur while another situation takes place, so it does not have a defined duration (e.g. a user's session in an e-commerce application).

• **Activity context:**

Refers to the tasks performed by entities (e.g., shopping, the task a user does at a particular time).

• **Relational context:**

Refers to entity relationships that arise from the circumstances in which the entities are involved [Zimmermann and al., 2007]. Relational context can be defined as *social* (i.e., interpersonal relations such as associations or affiliations), *functional* (i.e. the usage than an entity makes of another).

2.1.2. How to get a context?

A crucial question is how to get rich contextual information? In general, there are three main types of contextual information sources: explicit, implied or inferred.

Explicit:

Context information is already included in the data or directly requested from the user (form). As a person who communicates to the recommendation system the place that he wishes to visit it.

Implicit:

Information is obtained from data or the environment in which there is a user without explicitly asking him. For some platforms for recommending tourist sites from smartphones, for example, one can know the exact geographic location of an individual time, to make a recommendation.

Inferred:

Information is obtained using methods of exploitation and data mining. For example, the identity of a person browsing television channels may not be explicitly known for a society of cable television. However, the system can get to learn the moment of the day, the channel and the type of program watched by the different users from the same household (father, mother, children ...) and this with an acceptable accuracy in using data mining techniques.

Finally, the contextual information can be "hidden" in the data in some latent form, and we can use it implicitly to better estimate the unknown ratings without explicitly knowing this contextual information. For instance, in the previous example, we may want to estimate how much a person likes a particular TV program by modelling the member of the household (husband, wife, etc.) watching the TV program as a latent variable. It was also shown in [Palmisano and al., 2008] that this deployment of latent variables, such as intent of purchasing a product (e.g., for yourself vs. as a gift, work-related vs. pleasure, etc.), whose true values were unknown but that were explicitly modelled as a part of a Bayesian Network (BN), indeed improved the predictive performance of that BN classifier. Therefore, even without any explicit knowledge of the contextual information (e.g., which member of the household is watching the program), recommendation accuracy can still be improved by modelling and inferring this contextual information implicitly using carefully chosen learning techniques (e.g., by using latent variables inside well-designed recommendation models). A similar approach of using latent variables is presented in [Anand and Mobasher,2007].

2.2. Context-awareness definition

Context-Aware Computing is a term introduced by [Schilit and Want, 1994],[Brown and al., 1997]: "Context-aware applications are applications whose behaviour may vary depending on the context."
In [Dey and al., 2001]: "A system is context sensitive if it uses the context to provide information or services relevant to the user."

The following (**Table IV**) represents some context-aware recommendation system

Approach / **criteria**	**Micro-Profiling**	**DaVI**	**Discover**	**News@hand**	**MyMap**	**Sourcetone**	**Amazon**	**PAM**
Approach Recommendation	CF	CF	Hybrid	Hybrid	CBF	-	CF+CBF	CF+CBF
Type of data collection	Implicit	Implicit	Implicit	Implicit	Implicit / Explicit	Explicit	Implicit / Explicit	Implicit / Explicit
Type of context	Pre-filtering	Post-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering
Context definition	Time	Access to a item or web page	Any	Built in real time	Any	State of mind	Any	Any
Evaluation area	Music	Any	Any	Article Information	Tourism	Music	e-commerce	Movies
Flexibility	No	Yes	Yes	No	No	No	No	Yes

Table IV: Summary table of different context-aware recommendation systems

3. Context Modeling for Recommender Systems

Traditional recommendation systems are two-dimensional (2D) because they only consider the dimensions of the user and the item. In the frame of the recommendation incorporating the context, it is seen as another given information. In addition to the user and the item, we enhance the dimension of the context which will help improve the recommendation provided by the system. A system of context-sensitive recommendation will therefore consider scoring functions Under the form:

$$R: \text{User} \times \text{Item} \times \text{context} \rightarrow \text{Rating}$$

As can be seen, the contextual information can be of different aspects, such as time or location. In addition, each contextual aspect can have a complex structure reflecting the nature of the contextual information (as text). To cope with this complexity, information contextual is often hierarchical and represented as a tree.

[Adomavicius and al., 2005] and [Palmisano and al., 2008] also represent the context by a set of contextual dimensions K, each dimension contextual $k \in K$ being defined by a set of q attributes: $k = \{k^1, k^2, ..., k^q\}$ having a hierarchical structure and capturing a particular aspect of the context.

3.1. Methods for incorporating the context

Incorporation of contextual information can be done at different stages of process in a recommendation system. [Adomavicius and Tuzhilin, 2011] define three major approaches to contextualization following the moment when the context is injected. These approaches are:

• pre-filtering (contextual pre-filtering)

• post-filtering (contextual post-filtering)

• contextual modeling

We briefly present these three approaches in the following.

3.1.1. Contextual pre-filtering

The incorporation of the context by pre-filtering or pre-processing **(Figure 11)** consists in selecting a subset of data that is meaningful for the context in which we are restricting the recommendation process to this subset. this implies to build a template for each context. To illustrate this approach let us take the example of a movie recommendation system that uses the context time: if a user wants to watch a movie during the weekend, only films available during the weekend are candidates for the recommendation and only the notes of the users who saw the films during the weekend are used for the prediction of notes. The use of this filtering has been criticized because the whole data is reduced and can create problems for the prediction of notes if the system does not have enough data.

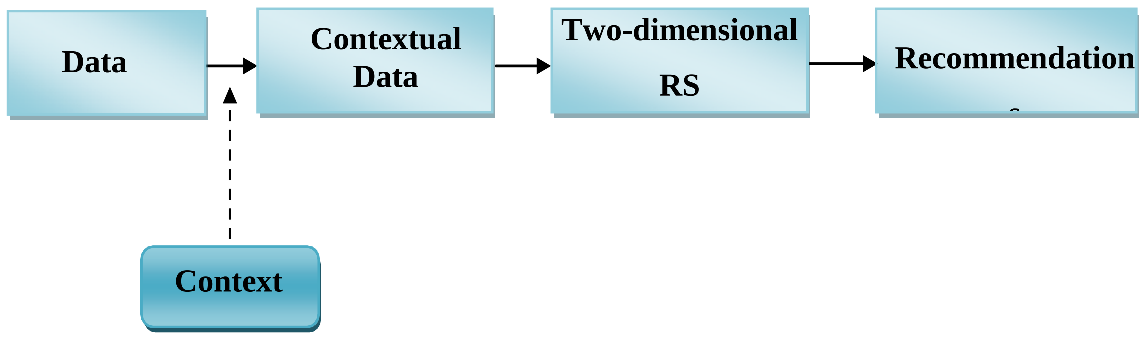


Figure 11: Contextual pre-filtering

3.1.2. Contextual post-filtering

In an approach of incorporating the context by contextual post-filtering, the recommendation system does not take into account the contextual data during of the recommendation process. The outputs of recommendation algorithms are modified retrospectively to reorder the list of recommended items in context function **(Figure 12)**. For example, a recommendation system of tourist places will use the location of the user (location context), and may decide to eliminate, after the event, recommendations from places too far from the position of the user.

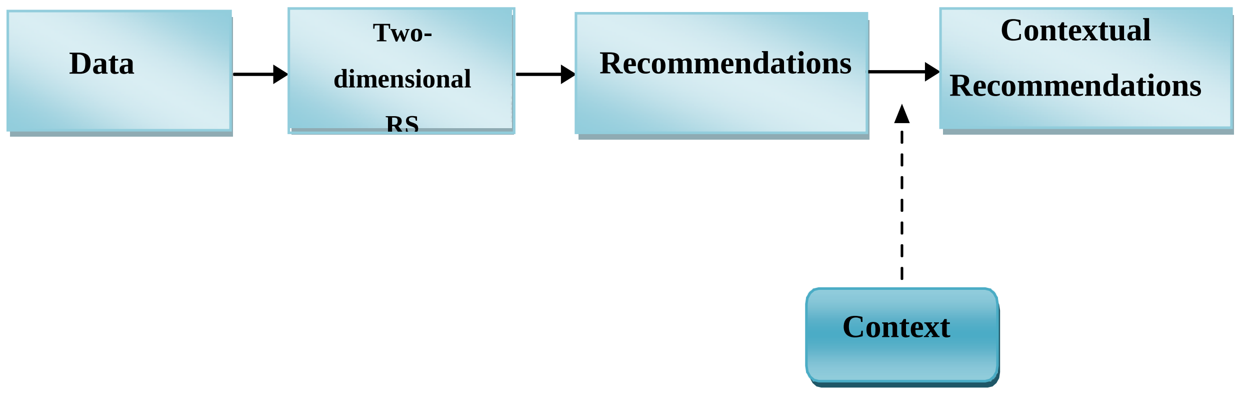


Figure 12: Contextual Post-filtering

3.1.3. Contextual modeling

The Context Modeling approach **(Figure 13)** consists of directly integrate contextual information into the recommendation process for the prediction of notes for items. To integrate the context, [Karatzoglou and al., 2010] propose methods of tensor factorization. For these methods, in addition to the first two dimensions traditionally used for items and users, each type of context is considered a new dimension. The note is no longer considered as a function with the two parameters item and user, but a function with parameters like the item, the user and the aspects of the context.

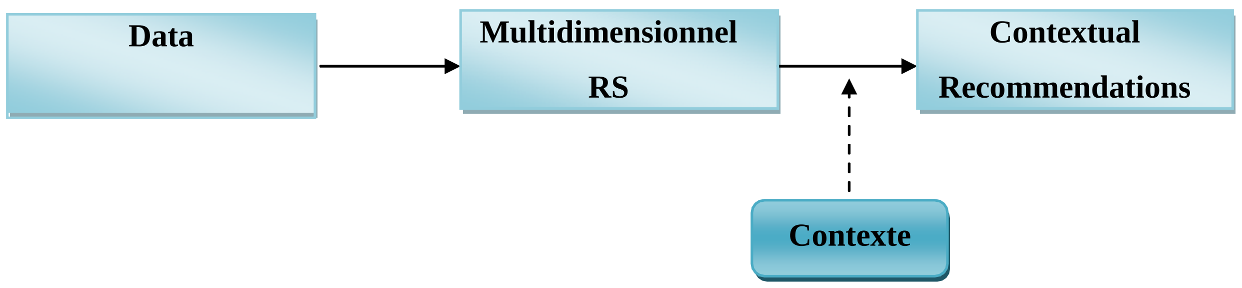


Figure 13: Contextual Modeling

4. Importance of the context in recommender System

In most custom recommendation systems, some problems may arise, such as recommending inappropriate products to a user, who makes a purchase of a gift on the internet to offer to his friend. The system may recommend similar products, although it is not his own preferences.

This problem can be overcome if the system knows the "context" in which the transaction took place. Although it is not easy to obtain this kind of information, it is possible to deduce from the data which is available in an explicit way (asks the user to specify contextual variables) or implicitly (as an example, to notice if the place of residence of a user has changed, or, when he made the transactions through his log files).

According to S. Lombardi and al, the contextual model always gives better results compared to the non-contextual model in terms of precision (number of correctly classified cases among all the cases), and of recall (number of all the cases considered relevant by the user and which are provided correctly).

5. Conclusion

In this chapter, we have presented the context concept as well as the context awareness, then we detailed how to integrate them in a recommendation system. Next we have listed some examples and applications that need the notion of context sensibility as well as their advantages and limits. The next chapter will be focused on the conception of our context-aware recommender system.

Chapter III: A proposed Recommendation System architecture

1. Introduction

In this chapter, we present our proposal system for a mobile recommendation system, adaptable to the user's profile and sensitive to its context. Our goal is to improve the visitor experience by offering a visit itinerary tailored, with works that correspond to his preferences and his context of visit (visit time, location, etc.). We propose a content-based and context-sensitive filtering approach, which adds context to the moment when we get the answer to the recommendation.

2. problematic and proposed solution

Our work aims at building a mobile application, to help enhance a visitor's experience (tourism). It's about to personalize the user's visit, according to his preferences and its context. Our work is geared towards systems of recommendation that have proven to be very effective in helping users to access the resources by which they would potentially be interested.

3. The used contextual recommendation approach

As we have mentioned, a significant amount of researches has dealt with the issue of recommendation in recent years. These works dealt with several areas such as machine learning, statistics and especially the search for information. The recommendation techniques can be classified in different ways. Sometimes several terms are used to designate the same method or approach.

Same factors produce different types of recommendations that are most commonly used Craft is content-based filtering and collaborative filtering. These two methods are presented in addition to their hybridization.

To offer a recommendation system for tourism, there are many issues to be faced. Some of them are generic and are related to any recommendation system. in our case we have chosen to use the content-based recommendation approach [Norha and al.,2018] (Figure 14)

3.1. How do Content Based Recommender Systems work?

A content based recommender deals with data that the user provides, either explicitly (rating) or implicitly. Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

3.1.1. How to represent the content?

The content of an item is a very abstract notion that gives us different options. One option may use a lot of variables. For example, for a restaurant we could consider the name, the category, the location, the description of the restaurant … the list goes on.

When we know which content we will consider. We need to transform all this data into a Vector Space Model, an algebraic representation of json documents.

Generally, we do this with a Bag of Words model, that represents documents ignoring the order of the words. In this model, each document looks like a bag containing some words. Therefore, this method allows word modeling based on dictionaries, where each bag contains a few words from the dictionary.

A specific implementation of a Bag of Words is the TF-IDF representation, where TF is for Term Frequency and IDF is Inverse Document Frequency.

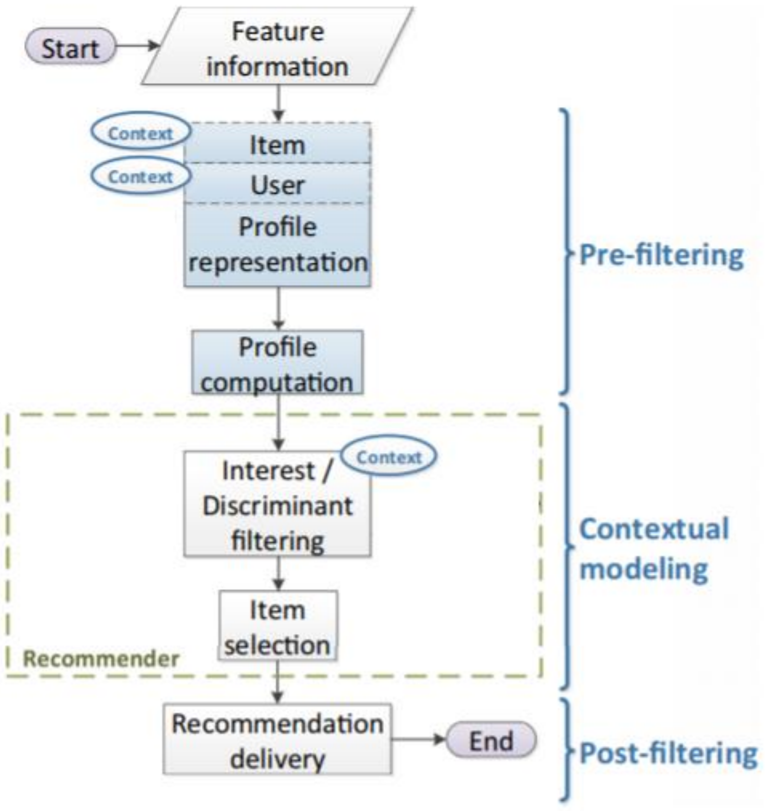


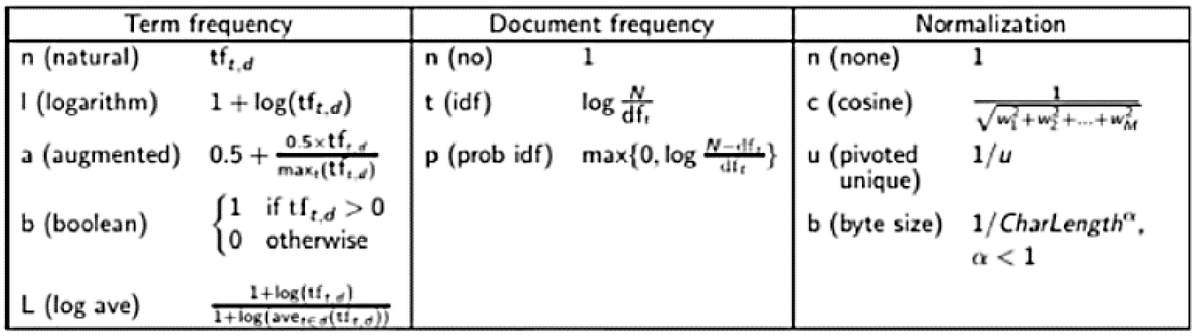
Figure 14: Process followed by content-based CARS

3.2. What are the used concepts in Content Based Recommenders?

The concepts of Term Frequency (***TF***) and Inverse Document Frequency (***IDF***) are used in information retrieval systems and also content based filtering mechanisms (such as a content based recommender). They are used to determine the relative importance of a document / article / news item / movie etc.

3.2.1. Term Frequency (TF) and Inverse Document Frequency (IDF)

TF is simply the frequency of a word in a document. Variants have been proposed. A simpler choice, called "binary", is to put 1 if the term appears in the document and 0 otherwise. In contrast, logarithmically the gross frequency can be normalized to smooth out the differences. A common normalization to take into account is the length of the document which is used to normalize the data by the maximum raw frequency of the document (Table V).



Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \ldots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - \mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(\mathrm{tf}_{t,d})}{1 + \log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Table V: Different Components of a TF-IDF representation

IDF is the inverse of the document frequency among the whole amount of documents. TF-IDF is used mainly because of two reasons: Suppose we search for ***"the rise of analytics"*** on Google**.** It is certain that "***the"*** will occur more frequently than "***analytics"*** but the relative importance of analytics is higher than the search query point of view. In such a way, TF-IDF weighting negates the effect of high frequency words in determining the importance of an item (document).

But while calculating TF-IDF, log is used to dampen the effect of high frequency words. For example: TF = 3 vs TF = 4 is vastly different from TF = 10 vs TF = 1000. In other words, the relevance of a word in a document cannot be measured as a simple raw count and hence the equation below:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{Tf_{t,d} \rightarrow w_{t,d} : 0 \rightarrow 0,\ 1 \rightarrow 1,\ 2 \rightarrow 1.3,\ 10 \rightarrow 2,\ 1000 \rightarrow 4,\ etc.}$$

It can be seen that the effect of high frequency words is dampened and these values are more comparable to each other as opposed to the original raw term frequency.

After calculating TF-IDF scores, how do we determine which items are closer to each other, rather closer to the user profile? This is accomplished using the **Vector Space Model** which computes the proximity based on the angle between the vectors.

3.2.2. How does Vector Space Model works?

In this model, each item is stored as a vector determined by its attributes (which are also vectors) in an **n-dimensional space** and the angles between the vectors (Figure 15) are calculated to **determine the similarity between the vectors**. Next, the user profile vectors are also created based on his actions on previous attributes of items and the similarity between an item and a user is also determined in a similar way.

we propose an example so as to try to make more clear the explanation.

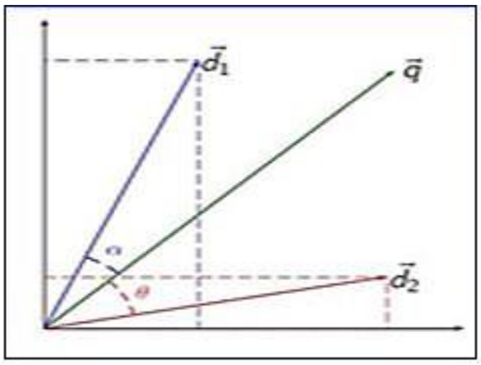


Figure 15: the angles between the vectors

The (Figure 16) Shown above is a 2-D representation of a two attributes, Cloud & Analytics. M1 & M2 are documents. U1 & U2 are users. The document M2 is more about Analytics than cloud whereas M1 is more about cloud than Analytics. In this case how to make it sure to know how the relative importance of documents are measures.

User U1, likes article on the topic 'cloud' more than the ones on 'analytics' and vice-versa for user U2. The method of calculating the user's likes / dislikes / measures is calculated by taking the cosine of the angle between the user profile vector *(U_i)* and the document vector.

The ultimate reason behind using cosine is that the value of cosine will increase with decreasing value of the angle between which signifies more similarity. The vectors are length normalized after which they become vectors of length 1 and then the cosine calculation is simply the sum-product of vectors.

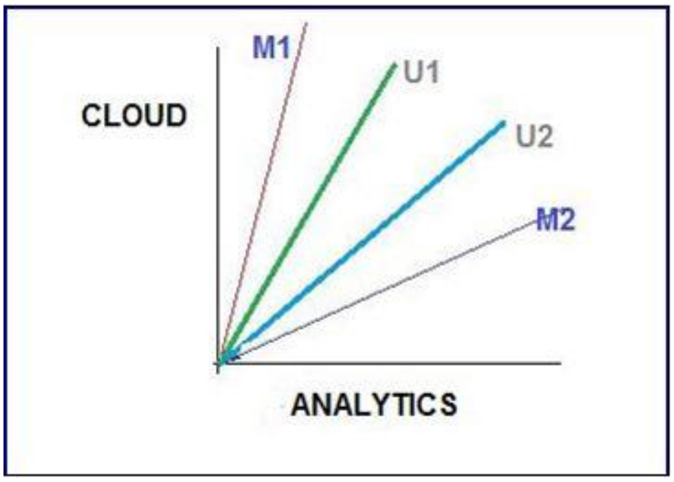


Figure 16: 2-D representation of a two attributes

4. Foursquare API Application Programming Interface (API'S)

July 24, 2014 Foursquare just announced, on its blog and in an interview of the responsible of the company with The Verge, a complete repositioning with the launch of a new application - Swarm - and a redesign of the current Foursquare mobile application advocating a clear division between "social" and "discovery of places". A new social network is added to the already long list[5].

The Foursquare Places API provides location based experiences with diverse information about venues, users, photos, and check-ins. The API supports real time access to places, Snap-to-Place that assigns users to specific locations, and Geotag. Additionally, foursquare allows developers to build audience segments for analysis and measurement. JSON is the preferred response format[6].

4.1. Foursquare city guide and Swarm: two applications for two uses

Foursquare is a social network that has been successful a few years ago with its check-in system allowing each user to "check" in a place and to signal their passage. A badge system enriched and gamified the user experience. Foursquare also allowed users to check places "checked" by users to find restaurants or cool venues.

in 2014 Foursquare splits into two complementary applications: the new Foursquare and Swarm.

- Foursquare will now focus on the social discovery of places
- Swarm will become the application dedicated to check-ins and sharing of your position.

4.2. The Swarm app

Foursquare has just launched its new downloadable application on Google Play and on iOS: Swarm witch is a new mobile app that will work in parallel with Foursquare focusing on the flagship feature of this service: check-ins.

4.2.1. What is Swarm for?

Swarm allows the user to check-ins just by opening the app and quickly sharing the location with friends. A "Hello, I'm well arrived" or a "I'm at the Café Station, who wants to join me?" Implied.

The "Neighborhood Sharing" feature allows to the user to give the location for a short time to his friends in the area. Convenient if the user wants to warn that the friends are available for lunch or show their location to facilitate an appointment.

The user can also propose a plan to his friends (ex: "who is going for a cinema?"), The notification will be pushed to all his friends in the neighborhood.

Swarm also makes it easy to spot who your friends are around for a drink in town or when you're landing in a city like Facebook's "Friends Nearby" feature. Just open the app to find that some

[5] https://www.programmableweb.com/api/foursquare

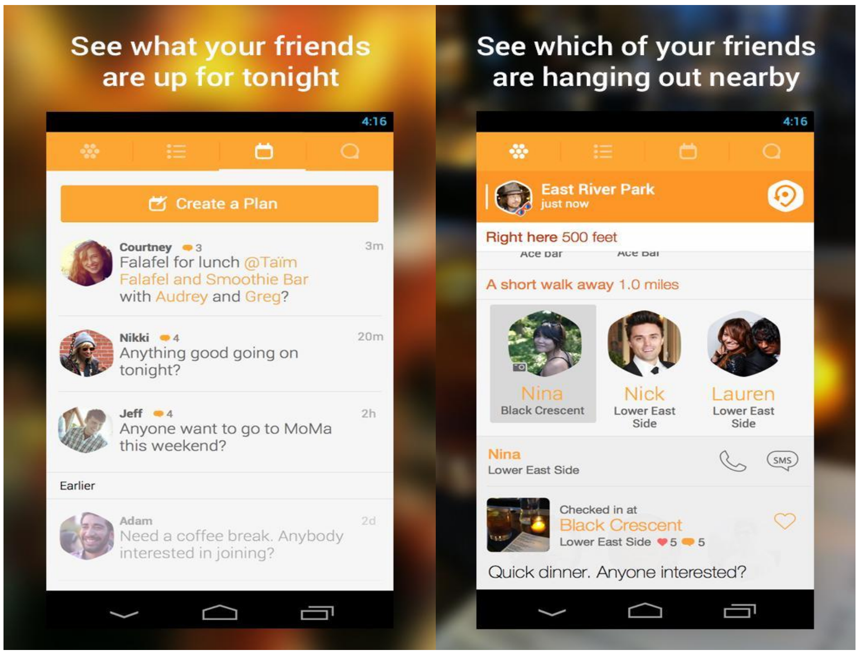
[6] https://www.kriisiis.fr/foursquare-45-millions-utilisateurs/

of your knowledge would be good to go out for example. Swarm will not publish your precise coordinates but a "place". Noah Weiss, vice president of product management talks about a "passive geolocation that users want". In short, intelligent geo-fencing.

The "Pilgrim" technology must allow Foursquare to "guess" the place where you are without "check-in" (the "p-check-ins" for "Pilgrim check-ins" or automatic check-ins). Taking advantage of the database of 60 million places identified in the world since the creation of Foursquare, Swarm will be able to understand in which cafe, discotheque or store you are at the moment when you open the application without you name specifically the name of the place. Pilgrim will use your GPS and Wifi signal as well as your travel habits to locate you accurately and validate the "check-in" in a transparent way.

Swarm makes it possible to share more than a simple localization: messaging, photos, comments, status, all the arsenal of the social network is there.

Swarm (Figure 17) should therefore allow Foursquare to regain control by collecting more and more places in a more intuitive way (concept of "frictionless sharing" of Facebook except that this time we will have the choice to download or not).



Courtney 3 3m
Falafel for lunch @Taïm Falafel and Smoothie Bar with Audrey and Greg?

Nikki 4 20m
Anything good going on tonight?

Jeff 4 2h
Anyone want to go to MoMa this weekend?

Earlier

Adam 2d
Need a coffee break. Anybody interested in joining?

Nina
Lower East Side

Checked in at
Black Crescent
Lower East Side 5 5
Quick dinner. Anyone interested?

Figure 17: Swarm on Android

With Swarm, Foursquare is reviving its position sharing feature and plans to reposition the Foursquare mobile app only around rating, "recommending" and "searching" for nearby places.

Indeed, billions of check-ins and comments have allowed Foursquare to gain a global understanding of places of interest in every city in the world. This database is continually updated by its users.

Here are the first screenshots of the new Foursquare:

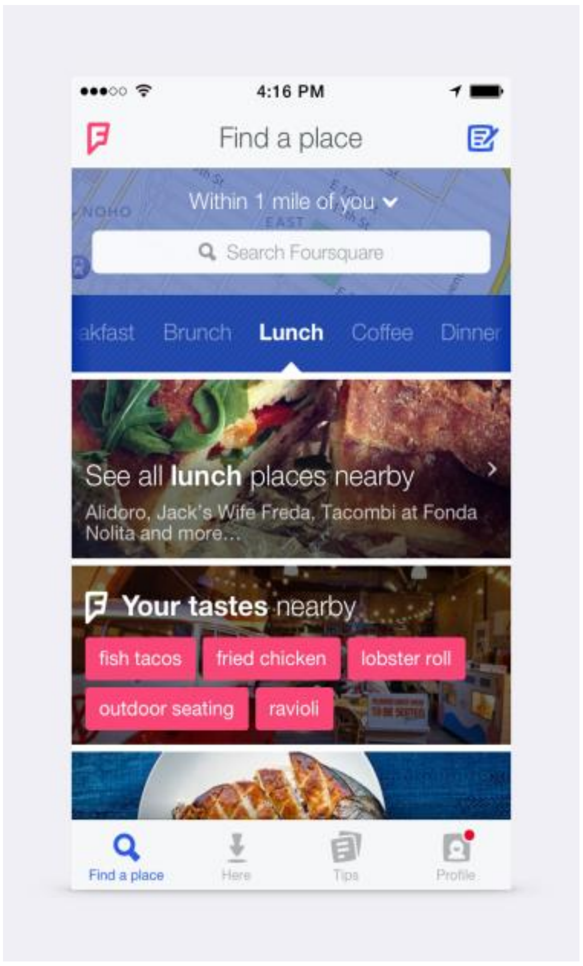


Figure 18: The first screenshots of the new Foursquare

4.3. The new Foursquare city guide application becomes

After some modifications on the application in 2014, it become:

- A real local search engine.
- A personalized recommendations engine. In the example given by Dennis Crowley, founder of Foursquare: If you go to 4 or 5 Japanese restaurants in the last six, your recommendations will be impacted by your behavior to serve you the best user experience.
- It will of course always be possible for stores, outlets, venues, restaurants and touristic places to be featured on Foursquare through their advertising program (see the article on how to integrate social networks in its marketing strategy)

5. Architecture of the proposed system

After having chosen the ideal approach witch is the content based Recommendation System relying on a particular methodology to solve the problem presented. We have proposed the following architecture (Figure 19):

In the project processing we have been able to propose the architecture to represents the global case, so the system works as follow:

After sending a user request the process starts with a raw data collection from the foursquare API, facebook, etc… then goes to

The Collection & Generalization & standardization module: starts its work which is the standardization and the structuration of data taken from the **Data source** in vectors whose attributes are also vectors. it is divided on three part:

- **profile analyzer**
- **The item analyzer**
- **Context analyzer**

Then, **Filtering module**: calculates the similarity between the users and items vectors using the TF-IDF method. the context vector is added to the processes using the post-filtering method witch incorporate the context in to the 2D recommendation by **Contextual recommendation module**. in order to give the contextual recommendation list to user.

Finely, the user feedback will be storage in the **Recommendation DB,** in order to be used a next time.

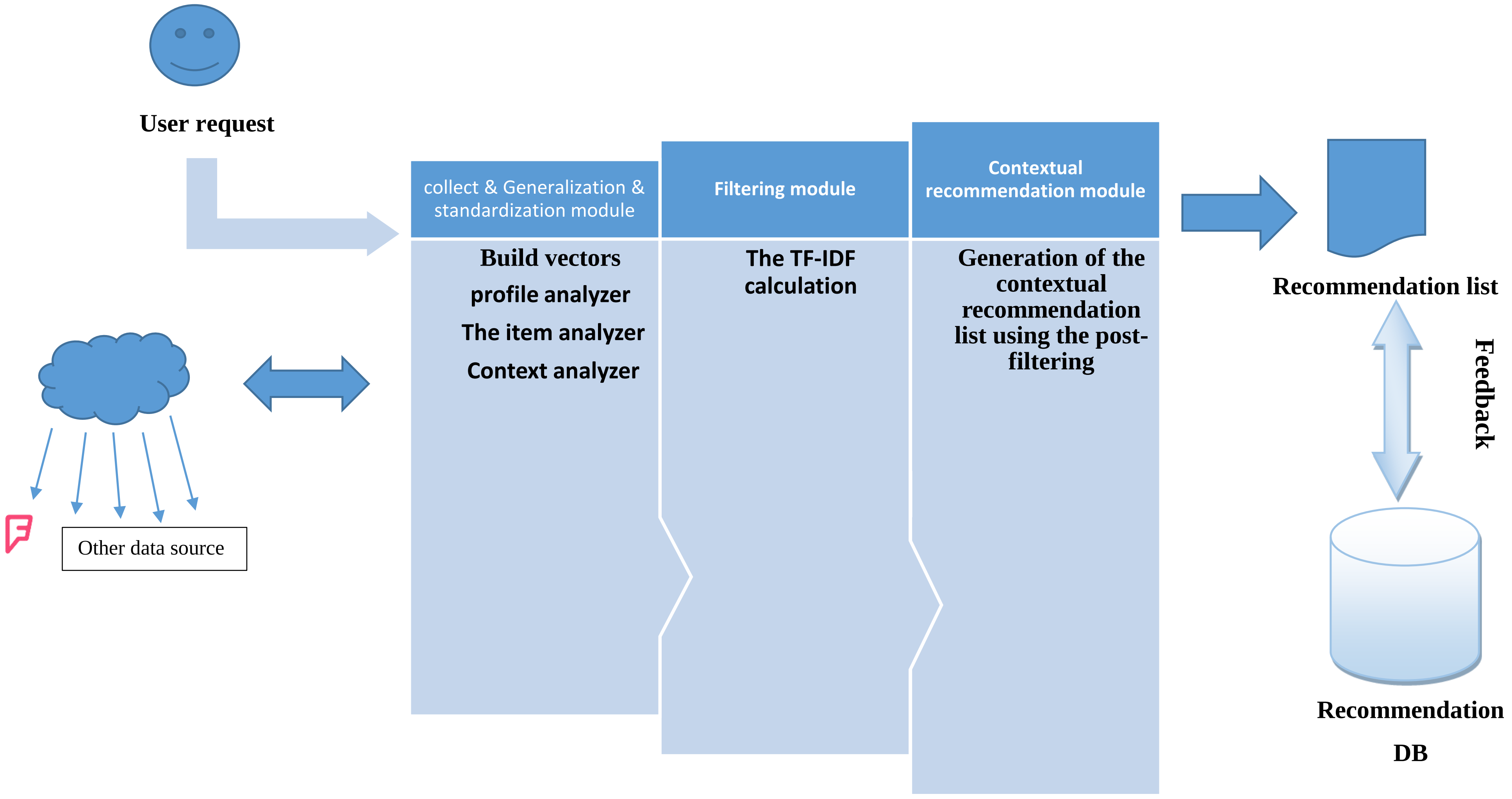


Figure 19: The global Architecture of the proposed system

5.1. Description

In order to solve the presented problem, we suggested the system architecture presented in Figure 19 above, which includes the following modules

The Collection & Generalization & standardization module: is divided on three part:

profile analyzer: this module collects representative data and preferences of users then it generalizes it, in order to learn and build the user profile in the form of a vector which its attributes are also vectors.

Context analyzer: this module aims to perform the pre-processing to extract the relevant information, structure and represents it in an appropriate target form (eg a keyword vector).

The item analyzer and also used to collect and structure the characteristics of each item taken from the Foursquare API

Filtering module: This module filters the relevant items by matching the representation of the user profile to the candidate items in the recommendation. The relevance of the item is calculated by the use of the TF-IDF calculation between the item under consideration and the user's profile.

Contextual recommendation module is used to join the context to the results of filtering in order to get an N-dimensional recommendation system

At last, after getting the list of recommended items we store them in the historical recommendation database and we memorize also the feedback of user

5.2. Architecture functionality

The proposed structure or architecture is split in three parts:

The first part:

Is presented by the data sources which describe all the kinds of information of the system designed by two type: explicit and implicit.

Explicit:

Which is a form completed by the user. In order to get more information allowing the construction of the user profile.

Implicit:

Which is a collect of internet data such as foursquare API used for the construction of the item vectors and others API (Facebook, etc…) to construct the user vectors.

The part two: The Collection & Generalization & standardization module

This part is composed of two new parts: analysis and storage of data.

The Collection & Generalization part:

Is presented by the three modules:(context analyzer, item analyzer and profile analyzer). these modules have the characteristics to compute and organize the information incoming from data sources module allowing the construction of the necessary vectors for the functionality of the system.

standardization part:

In this sub-system the structured data will be maintain in data bases in order to be used in the future if it is necessary (vectors).

Part three

The filtering module use the two vectors $V_{i,}$ U_i in order to give a two dimensional recommendation (**Filtering module**) after that the vector of context will be introduced into the system for a second calculation (post-filtering) so as to show the similarity rate of the three vectors. All those operations are realized by the **Contextual recommendation module**. It delivers the list of recommended items.

By the end, the user dives his own opinion on the recommendation which will be stored into the data base in order to be used for enriching the user profile and increase the performance of the proposed system.

6. Conclusion

In this chapter, we have detailed the approach of recommendation used in the realization of our application. The calculation of the similarity is mentioned and used and finally we present our architecture and its description and functionality.

we conclude by enriching our chapter with important information on the Foursquare API which explains how it is used in our application in the next chapter the implementation of hole system will be presented.

Chapter IV: Implementation and TEST

1. Introduction

We present in this last chapter the steps of realization of our application which consists of completing the previous phases. We proceed as follows:

- Choice of development tools (technological and software) that we used.
- Descriptions of the features offered by our mobile application.

Then we present screenshots of the application to include it and illustrate the main features. we conclude with a small evaluation of our proposed system

2. Technical choices

In this part we present the different languages and tools used for the development of our mobile application

2.2 Plugins and software

This part lists each software and development tools to use in creating the application:

2.2.1. Anaconda (Python .3)

Anaconda (Figure 20) is an open source distribution of Python and R which aims to simplify the management and deployment of modules(Please see the official website[7]).

Figure 20: anaconda logo

2.2.2. Python

Python (Figure 21) is an interpreted programming language with a design philosophy stressing the readability of code. Its syntax is simple and expressive. Python has an extensive, easy-to-use library of standard modules.

[7] https://www.anaconda.com/

The capabilities of Python can be extended with modules developed by third parties. In general, to simplify operations, it is left up to individual users and groups to install these third-party modules in their own directories. However, most systems offer several versions of Python as well as tools to help the installation of the third-party needed modules (for more information see the official website[8]).

Figure 21: python logo

2.2.3. Jupyter

"Project Jupyter (Figure 22) is a non-profit, open-source project, born out of the IPython Project in 2014. It is evolved to support interactive data science and scientific computing across all programming languages."[9]

Figure 22: jupyter logo

2.2.4. Mongo dB

MongoDB[10] (Figure 23) is a cross-platform and open-source document-oriented database, it is a kind of NoSQL database. As a NoSQL database, MongoDB shuns the relational database's table-based structure to adapt JSON-like documents that have dynamic schemas which it calls BSON.

This makes data integration for certain types of applications faster and easier. MongoDB is built for scalability, high availability and performance from a single server deployment to large and complex multi-site infrastructures.

[8] https://www.python.org/
[9] http://jupyter.org/about.html
[10] https://www.mongodb.com/fr

Figure 23: mongo DB logo

2.2.5. Fltter/dart

Flutter[11] (Figure 24) is an open-source mobile application development framework created by Google. It is used to develop applications for Android and iOS, as well as being the primary method of creating applications for Google Fuchsia.

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets

Dart platform

Dart[12] is a client-optimized programming language for fast apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, backend and web applications.

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

On Android, and on Windows, macOS and Linux via the semi-official Flutter Desktop Embedding project, Flutter runs in the Dart virtual machine which features a just-in-time execution engine. Due to App Store restrictions on dynamic code execution, Flutter apps use ahead-of-time (AOT) compilation on iOS.

Figure 24 : Flutter & Dart logo

2.2.6. Vs Code

Visual Studio Code[13] is a free, multi-platform, open-source code editor (Windows, Mac, and Linux) developed by Microsoft, not to be confused with Visual Studio, the proprietary IDE of Microsoft. VS Code is developed with Electron and exploits advanced editing features of the Monaco Editor project.

[11] https://flutter.dev/docs/get-started/install
[12] https://dart.dev/tools/sdk
[13] https://code.visualstudio.com/

Primarily designed for application development with JavaScript, Typescript and Node.js, the editor can adapt to other types of languages thanks to a well-supplied extension system.

- **Main features**

VSCode offers different elements that can be interesting for developers at all levels, so that compared to other text editors (e.g. Brackets), the level is rather intermediate / advanced. Nevertheless, VSCode can be a good starting choice even for a beginner in the prospect of then reaching some expertise. In addition, the advanced editing features of VSCode can also be exploited in other areas such as formatting / cleaning of text files or raw data.

Among the main features of the software are:

- **IntelliSense:**

an advanced technology that offers, in addition to highlighting the syntax and automatic completion of the code, an inference system articulated and based directly on the logic of the source code;

- **Native integration with Git:**

the software implements the Git version management system directly in the editor interface, which is an advantage to be able to perform versioning operations more easily

- **Integrated command line:**

Always in the editor interface, it is possible to launch the command line and execute all the commands available on the operating system;

- **Eco-system of extensions:**

the extensions are at the heart of the project and there is even a simple system to develop / publish its own extensions

- **Integrated Debugging:**

For more advanced developers, there are also debugging features directly inside the editor.

Visual Studio Code

Figure 25: VS code logo

2.2.7. Flask

Flask[14] is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frameworks related tools. Extensions are updated far more frequently than the core Flask program.

3. Realization of the context-sensitive recommendation system offered by a mobile application

First of all, we are preparing to start programming the application

The first stage:

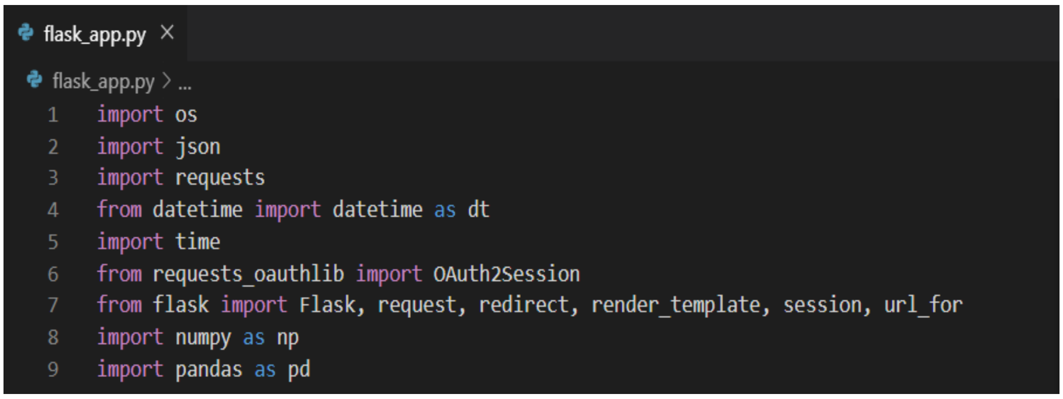
The first step provides a brief description of how getting started with Foursquare API and discusses how data with Python can be wield.

We'll talk about: how to get data from Foursquare API? and also the other APIs used in the project.

After setting all the needed steps to access the API, then we get started with the codes which are the flask script (backend) and the flutter application(frontend).

The Flask (Python framework) is used as back end for our REST API and Flutter for mobile. We will also use MongoDB as our database to store data about the recommendation and the user feedback.

In order to use the functions of the necessary libraries, the part of code is given in Fig 26.

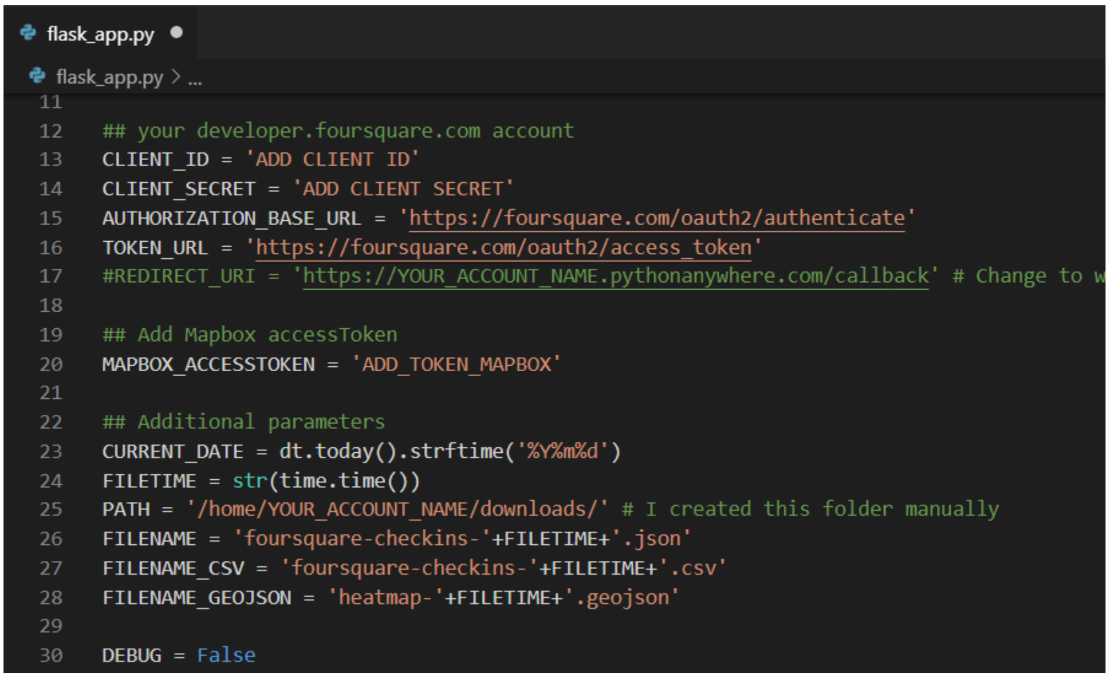


```python
import os
import json
import requests
from datetime import datetime as dt
import time
from requests_oauthlib import OAuth2Session
from flask import Flask, request, redirect, render_template, session, url_for
import numpy as np
import pandas as pd
```

Figure 26: necessary libraries

Figure 27 represents the function that allow us to get the data from the foursquare API with python.

[14] https://palletsprojects.com/p/flask/

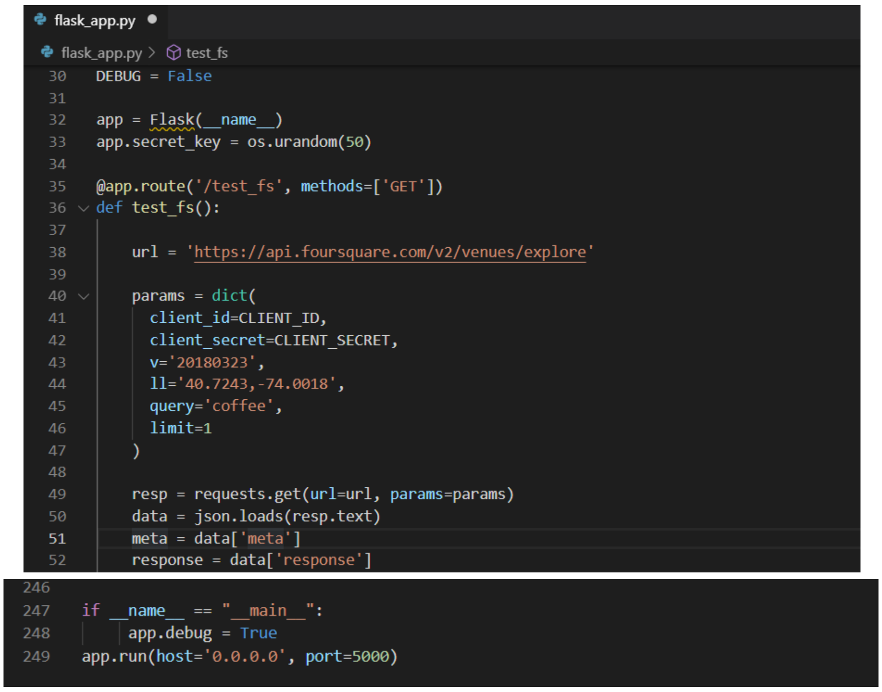


```python
## your developer.foursquare.com account
CLIENT_ID = 'ADD CLIENT ID'
CLIENT_SECRET = 'ADD CLIENT SECRET'
AUTHORIZATION_BASE_URL = 'https://foursquare.com/oauth2/authenticate'
TOKEN_URL = 'https://foursquare.com/oauth2/access_token'
#REDIRECT_URI = 'https://YOUR_ACCOUNT_NAME.pythonanywhere.com/callback' # Change to w

## Add Mapbox accessToken
MAPBOX_ACCESSTOKEN = 'ADD_TOKEN_MAPBOX'

## Additional parameters
CURRENT_DATE = dt.today().strftime('%Y%m%d')
FILETIME = str(time.time())
PATH = '/home/YOUR_ACCOUNT_NAME/downloads/' # I created this folder manually
FILENAME = 'foursquare-checkins-'+FILETIME+'.json'
FILENAME_CSV = 'foursquare-checkins-'+FILETIME+'.csv'
FILENAME_GEOJSON = 'heatmap-'+FILETIME+'.geojson'

DEBUG = False
```

Figure 27: API call

❖ My account information:

- CLIENT ID
  - ❖ MXNJ2A14M21WWWHFE3WGY20N12MZNEIAJPWUTEVQTNS3O1RO
- CLIENT SECRET
  - ❖ sssshjhH521441Z3GXT0BXMFBJ4E4IPEEFWRB2TFNJQHDXXJOZF53EW
- MAPBOX_ACCESSTOKEN
  - ❖ sk.eyJ1IjoiZmF0aTM0ZmF0aW1hIiwiYSI6ImNrMHVtd3J6aDA0aWgzaWxndmxzZzJzb2cifQ.cUYzgFTsAeehMbFAHqMkkA



```python
DEBUG = False

app = Flask(__name__)
app.secret_key = os.urandom(50)

@app.route('/test_fs', methods=['GET'])
def test_fs():

    url = 'https://api.foursquare.com/v2/venues/explore'

    params = dict(
      client_id=CLIENT_ID,
      client_secret=CLIENT_SECRET,
      v='20180323',
      ll='40.7243,-74.0018',
      query='coffee',
      limit=1
    )

    resp = requests.get(url=url, params=params)
    data = json.loads(resp.text)
    meta = data['meta']
    response = data['response']
```

```python

if __name__ == "__main__":
    app.debug = True
app.run(host='0.0.0.0', port=5000)
```

Figure 28: basic API functionality of locating a coffee place in NY

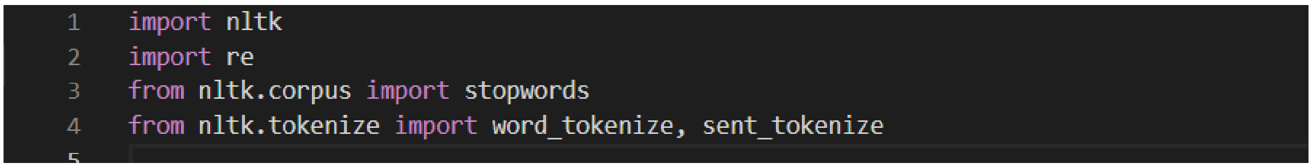
After doing a collection of API data. Now, we should go to the most important part: The Content Based Recommender Systems using the similarity calculus using TF-IDF method.

Based on the description of the item, it can be proposed to a user. The description goes deeper into the product details, such as: title, summary, taglines, genre. It provides much more information about the item. The format of these details are in text format(string) and is important to convert

Term Frequency-Inverse Document Frequency(TF-IDF) TF-IDF is used in Information Retrieval for feature extraction purposes and it is a sub-area of Natural Language Processing(NLP). Also, TF-IDF is a measure used to evaluate how important a word is to a document in a document corpus. The importance of the word increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

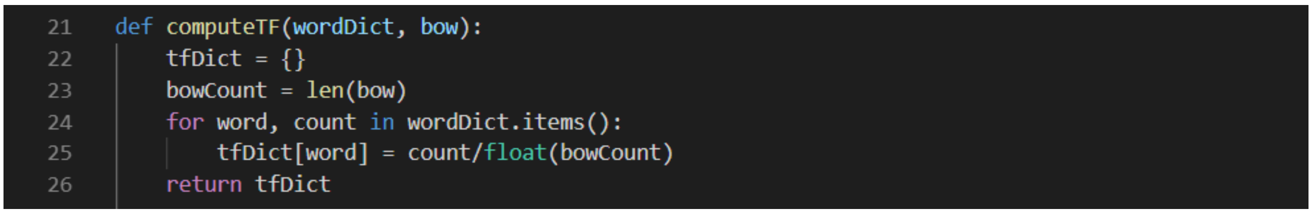
❖ **LET US START COODING ……**

Restarting with importing the required libraries:



```python
import nltk
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
```

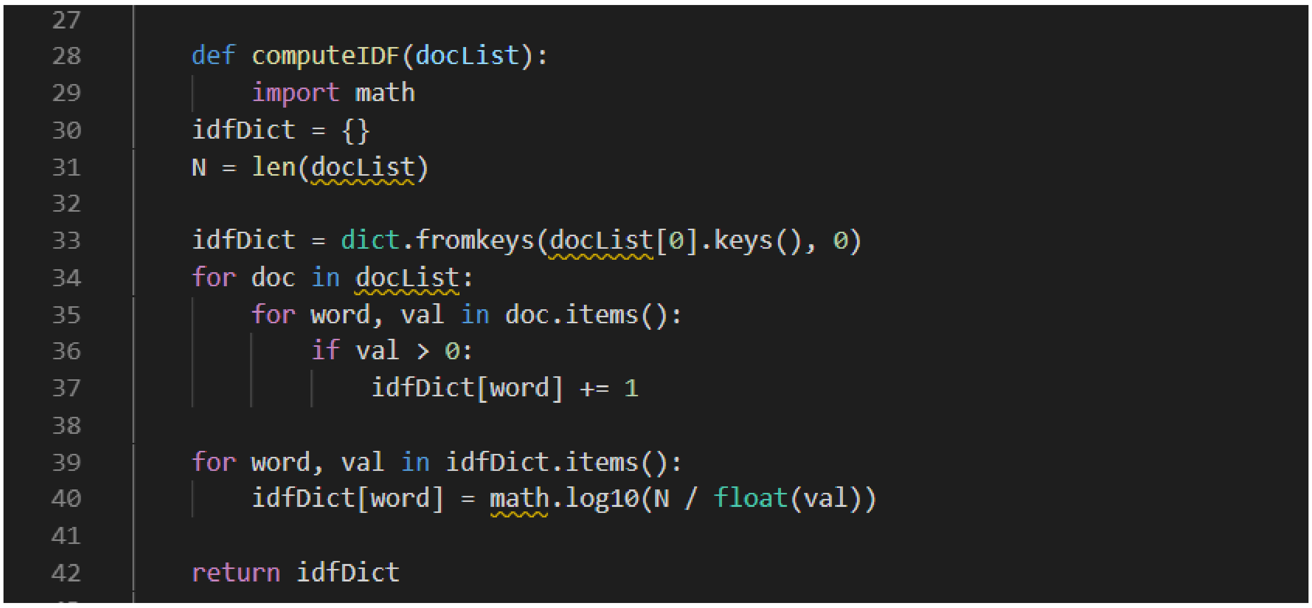
After a very simple pre-processing function which aim to clean all the unnecessary data. The function (figure 29) computeTF computes the TF score for each word in the corpus, of the json file.



```python
def computeTF(wordDict, bow):
    tfDict = {}
    bowCount = len(bow)
    for word, count in wordDict.items():
        tfDict[word] = count/float(bowCount)
    return tfDict
```

Figure 29: The function computeTF

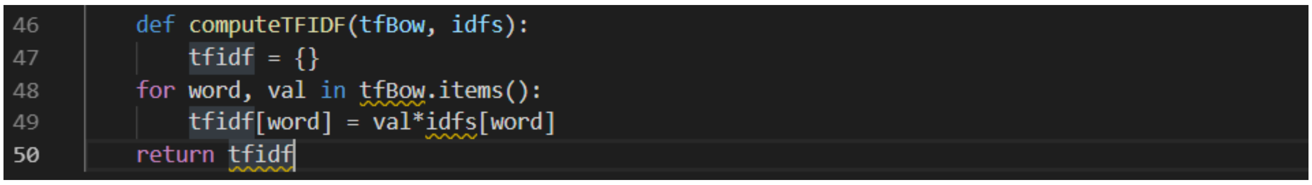
The function (Figure 30) computeIDF computes the IDF score of every word in the corpus.



```python

def computeIDF(docList):
    import math
idfDict = {}
N = len(docList)

idfDict = dict.fromkeys(docList[0].keys(), 0)
for doc in docList:
    for word, val in doc.items():
        if val > 0:
            idfDict[word] += 1

for word, val in idfDict.items():
    idfDict[word] = math.log10(N / float(val))

return idfDict
```

Figure 30: The function computeIDF

computeTFIDF (Figure 31) multiply the TF and IDF score.

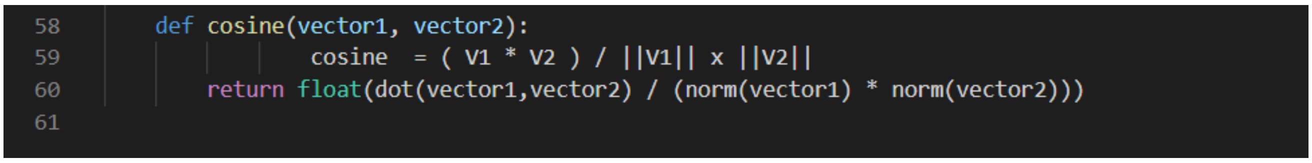


```python
def computeTFIDF(tfBow, idfs):
    tfidf = {}
for word, val in tfBow.items():
    tfidf[word] = val*idfs[word]
return tfidf
```

Figure 31: The function computeTFIDF

To find Related, the cosine of the angle between the vectors allow us to make comparison between them. We use the cosine of the angle as a metric for comparison. If the cosine is 1 then the angle is 0° and hence the vectors are parallel (and the document terms are related). If the cosine is 0 then the angle is 90° and the vectors are perpendicular (and the document terms are not related).

We calculate the cosine between the document vectors in python using (see Figure32):

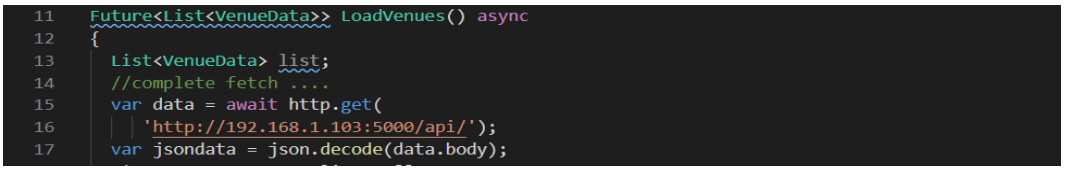


```python
def cosine(vector1, vector2):
            cosine  = ( V1 * V2 ) / ||V1|| x ||V2||
    return float(dot(vector1,vector2) / (norm(vector1) * norm(vector2)))

```

Figure 32: calculate the cosine between the document vectors

After all this treatment in the backend, we go to the frontend of our flutter application:

We start by making the connection between the flask server and our application (Figure 33)



```dart
Future<List<VenueData>> LoadVenues() async
{
  List<VenueData> list;
  //complete fetch ....
  var data = await http.get(
      'http://192.168.1.103:5000/api/');
  var jsondata = json.decode(data.body);
```

Figure 33: NetworkCall

As a beginning figure 34 represents the first interface of our application which requires the user to activate the localization

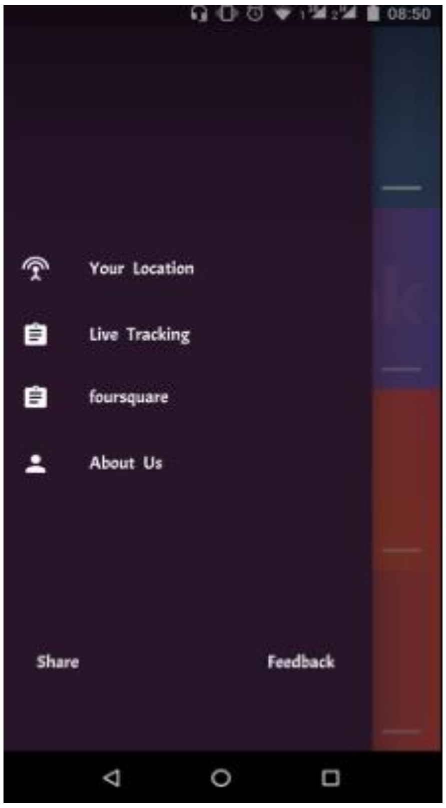


Figure 34: location interface

After that the user is free to choose which social networking sites can be used to build his profile: see figure 35

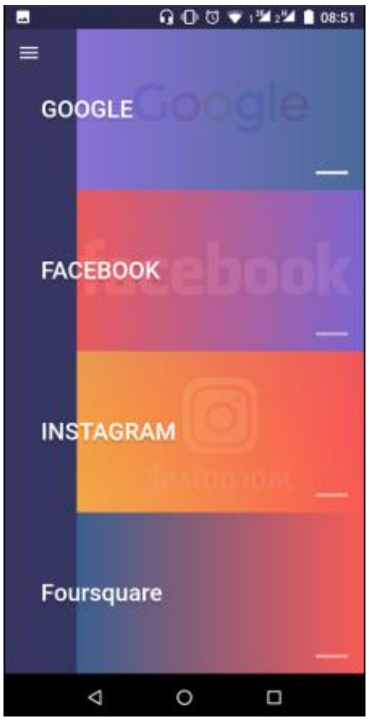
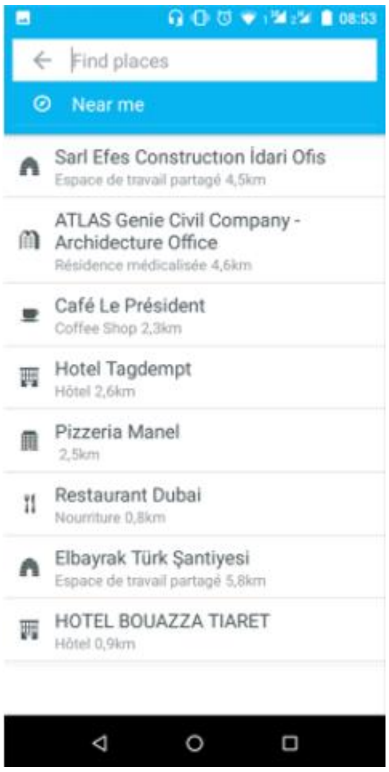


Figure 35: menu interface

The user information is collected in order to to determine which items are recommended. The last interface (figure 36) of our application that allows the user to request a recommendation he needed, and allow him to get the result.



Sarl Efes Construction İdari Ofis
Espace de travail partagé 4,5km

ATLAS Genie Civil Company - Architecture Office
Résidence médicalisée 4,6km

Café Le Président
Coffee Shop 2,3km

Hotel Tagdempt
Hôtel 2,6km

Pizzeria Manel
2,5km

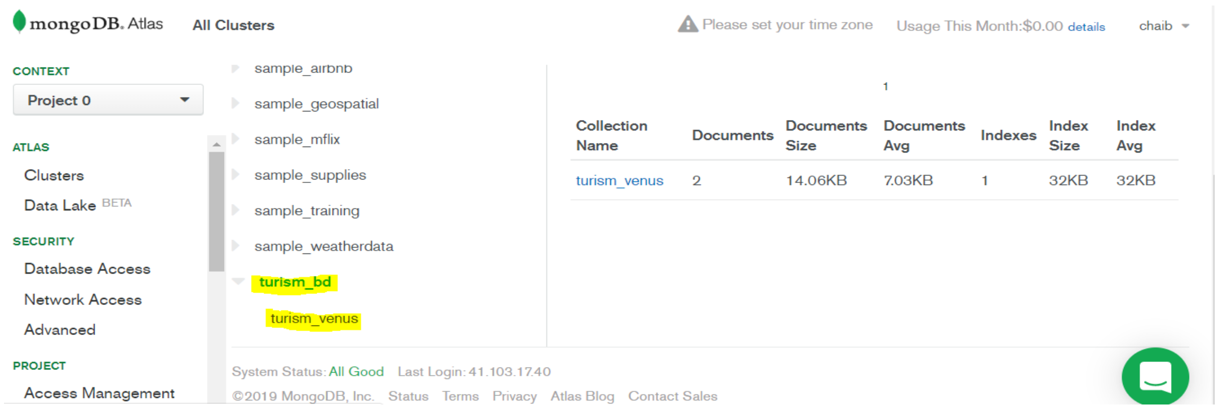
Restaurant Dubai
Nourriture 0,8km

Elbayrak Türk Şantiyesi
Espace de travail partagé 5,8km

HOTEL BOUAZZA TIARET
Hôtel 0,9km

Figure 36: navigation interface

In order to improve the proposed application, a database named "tourism_bd" is created under mongodb that will be used for the storage of the recommendation given by the system with the feedback of the user (Figure 37).



sample_airbnb
sample_geospatial
sample_mflix
sample_supplies
sample_training
sample_weatherdata
turism_bd
turism_venus

Collection Name	Documents	Documents Size	Documents Avg	Indexes	Index Size	Index Avg
turism_venus	2	14.06KB	7.03KB	1	32KB	32KB

System Status: All Good Last Login: 41.103.17.40
©2019 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

Figure 37: our database

4. Results and Evaluation

In order to evaluate the effectiveness of our application it is necessary to test it. But we can say that despite its limitations, the user can receive effective recommendations in record time according to his context

With a focus on helping users mimic the experience of having visited a place before going to it, we needed to rebuild the place page from the ground up. We focused heavily on the “evaluation stage,” which is when users analyze and decide if a place actually satisfies what they are looking for. We elevated features such as tagged photos to give users a quick look at the most popular food tastes at the venue, interior and exterior photos, popular tips and feedback from users and a quick bulleted list of what the venue had to offer.

5. conclusion

In the fourth chapter we have described the work environment, as well as the different software used in the development and implementation of the application. We also presented all the interfaces of the application and its functionalities.

GENERAL CONCLUSION

The recommendation systems are becoming a very interesting domain of studies and research, indeed it become a part of all the areas of the recent science. It is namely used in the ecommerce, tourism, social network which it is used to create a user profile needed for giving a value to the recommendation to satisfy the users.

The user profile is not enough to make a good recommendation because of the poverty of the data and can be not sure information. For this reasons we opted to integrate the context (locations, time and preference) into the recommendations.

In this project we based all our intention in the content base approach using the post filtering in order to integrate the context into classical recommendation. When working on the project e have used the very recent technology of development to achieve the development. And we base the work on the API which content a quit a lot of information needed for the tourism recommendation.

In the project an application is proposed which satisfy the need and the constraint of the users. This application uses the TF-IDF methods to calculate the similarities between the three vectors: namely user vector Vi, Item vector Ui and context vector Ci. This method gives us a consistent recommendation.

The application we proposed satisfy only a part of the needs of the user, because of limits the lack of persistent information and the limits of chooses of the proposed API. This limits will allow us to give another approach such as a psychological user profile to get a consistent information about the user. The mentioned problems give us the opportunities to integrate more contextual information.

BIBLIOGRAPHY

[Malone et al., 1987] Malone, T., Brobst, S., Cohen, S., Grant, K., and Turbak, F. (1987). Intelligent information des systèmes de partage. In Communications of the ACM, volume 30, pages 390–402.

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. In Commun. ACM, volume 35, pages 61–70.

[Resnick and Varian, 1997] Resnick, P. and Varian, H. (1997). Recommender systems. In Communications of the ACM, volume 40, pages 56–58.

[Resnick and Varian1997] P. Resnick and H. R. Varian. Recommender systems. Communications of the ACM, 40(3) :56–58, 1997.

[Maes and Shardanand, 1995] Maes, P. and Shardanand, U. (1995). Social information filtering: algorithms for automating "word of mouth". In the SIGCHI conference on Human factors in computing systems, Denver, Colorado, United States. ACM Press/Addison-Wesley Publishing Co.

[Maes et al.1999] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell. Commun. ACM, 42(3) :81–ff., March 1999.

[Tintarev and Masthoff2007] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In 2007 IEEE 23rd International Conference on Data Engineering Workshop, pages 801–810, 2007.

[Wei et al.2007] Kangning Wei, Jinghua Huang, and Shaohong Fu. A survey of e-commerce recommender systems. In 2007 International Conference on Service Systems and Service Management, pages 1–5, 2007.

[Almazro et al.2010] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. arXiv e-print 1006.5278, June 2010.

[Prasad2012] Rvvsv Prasad. A categorical review of recommender systems. International Journal of Distributed and Parallel systems, 3(5) :73–83, September 2012.

[Lü et al.2012] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. Physics Reports, 519(1) :1–49, October 2012.

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

[Su and Khoshgoftaar 2009] Xiaoyuan *Su* and Taghi M. *Khoshgoftaar*. A Survey of Collaborative Filtering Techniques Department of Computer Science ... Received 9 February *2009*; Accepted 3 August *2009*. Academic Editor: Jun Hong.

[Palmisano and al., 2008] Palmisano, C., Tuzhilin, A., and Gorgoglione, M., Using context to improve predictive modeling of customers in personalization applications. IEEE Transactions on Knowledge and Data Engineering, 20(11):1535–1549, 2008.

[Anand and Mobasher,2007] Anand, S.S., and Mobasher, B., Contextual recommendation. WebMine, LNAI, 4737:142– 160, 2007.

[Zimmermann et al., 2007] Zimmermann, A., Lorenz, A., and Oppermann, R. (2007). An operational definition of context. In International and Interdisciplinary Conference on Modeling and Using Context, pages 558–571. Springer.

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and contextawareness. In Handheld and ubiquitous computing, pages 304–307. Springer.

[Adomavicius and Tuzhilin, 2011] Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. In Recommender systems handbook, pages 217–253. Springer.

[Karatzoglou et al., 2010] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010). Multiverse recommendation : n-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the fourth ACM conference on Recommender systems, pages 79–86. ACM.

[Brezillon, P. (2002)] Expliciter le contexte dans les objets communicants. In ´ Gerard, P., Kintzig, C., Privat, G., and Favennec, P., editors, ´ Les Objets Communicants, pages 295–303. Hermes.

[Dey & al., 2001] Dey, A., Abowd, G., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. HumanComputer Interaction, 16(2) :97–166.

[Norha & al,2018] Characterizing context-aware recommender systems: A systematic literature review Norha M. Villegas ∗, Cristian Sánchez, Javier Díaz-Cely, Gabriel Tamura]

[Schilit, B. and Theimer, M. (1994)] Disseminating active map information to mobile hosts. IEEE Network : The Magazine of Global Information Exchange, 8(5) :22– 32.

[Brown, P and al., 1997] Brown, P., Bovey, J., and Chen, X. (1997). Context-aware applications : from the laboratory to the marketplace. IEEE Personal Communications, 4(5) :182–196.

[Asthana and al., 1994] https://ieeexplore.ieee.org/abstract/document/4624426

[Abowd and al., 1997] Cyberguide: A mobile context-aware tour guide

GD Abowd, CG Atkeson, J Hong, S Long, R Kooper, M Pinkerton Wireless networks 3 (5), 421-433

[Dey and al., 1999] https://www.cc.gatech.edu/fce/ctk/pubs/ISWC99.pdf

https://ieeexplore.ieee.org/abstract/document/806639

[Want and al., 1992] http://www.cs.columbia.edu/~coms6998-11/papers/activebadge.pdf

[Villegas et al,2010] N.M. Villegas, H.A. Müller, Managing dynamic context to optimize smart interactions and services, in: The Smart Internet, Springer, 2010, pp. 289–318.

WEBOGRAPHY

[1] https://en.wikipedia.org/wiki/Context-sensitive_user_interface

[2] Ref https://www.programmableweb.com/api/foursquare

[3] https://www.kriisiis.fr/foursquare-45-millions-utilisateurs/

[4] https://www.kriisiis.fr/foursquare-45-millions-utilisateurs/

[5] Image Source: easyrec.org

More useful website

[http://dataconomy.com/2015/03/an-introduction-to-recommendation-engines/]

[https://towardsdatascience.com/what-are-product-recommendation-engines-and-the-various-versions-of-them-9dcab4ee26d5]

[https://towardsdatascience.com/what-are-product-recommendation-engines-and-the-various-versions-of-them-9dcab4ee26d5]

[https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/]