SAPIENZA
UNIVERSITÀ DI ROMA

Recommendation Systems in Tourism: Design of a Novel Method and Comparative Evaluation with State of the Art Techniques

Facoltà di Ingegneria dell'informazione, Informatica e Statistica

Corso di Laurea Magistrale in Data Science

Candidate

Mateusz Antoni Zmyslowski

ID number 1311508

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Academic Year 2021/2022

Thesis defended on 27 January 2023
in front of a Board of Examiners composed by:

Prof. Anagnostopoulos Aristidis (chairman)
Prof. Brutti Pierpaolo
Prof. Chatzigiannakis Ioannis
Prof. Daraio Cinzia
Prof. Petti Manuela
Prof. Quattrociocchi Walter
Prof. Tieri Paolo

Recommendation Systems in Tourism: Design of a Novel Method and Comparative Evaluation with State of the Art Techniques
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: zmyslowski.1311508@studenti.uniroma1.it

Abstract

Recently [3], recommendation systems have become an active topic. Studies indicated that existing tourism recommendation systems provide not accurate recommendations that do not meet tourist's expectations. One of the main reasons for this problem is that most of these systems neglect previous user reviews. This paper proposes a tourism recommendation system with the integration of the user review element. The user reviews are analyzed, processed and then used for the recommendations.
It will be described the latent Dirichlet allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document [4].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sometimes happens that people takes decisions not knowing the existence of other alternatives and this can lead people to trusting in third parties rather than in themselves while taking decisions. This lack of awareness, and also the increasing amount of information that new technologies, such as the Internet, supply are having a positive effect on the importance of having filtering and selection methods that facilitate the decisionmaking process [1].
To users are given a very large quantity of information that they eventually get overwhelmed. The recent appearance of new and practical tools to help this matter in a more effective way. These tools provide users some directions that can help them understand the information needed to make a more advantageous use of it. The combination of the large amount of information accessible, the great range of alternatives available and the users inexperience in some sectors make it clear the need of automatic or semiautomatic systems, capable of helping users in the decision-making process through directions and suggestions [2].

1.1 Tourism

Tourism has to do with the activities that people makes during their trips in places that are different from their regular environment. Progressively, the tourism sector has become one of the main activities that collect more money around the world; not only for the direct contribution it supplies, but also for the expansive effect that has exerted on a large range of related activities.
Present tourism is characterized by the appearance of new markets, greater opportunities to select receptive centres, higher participation and interest in being in contact with the nature, possibility to distribute holidays through the year, and application of new technologies in the tourism industry.
Since tourism market is under constant changes, there is need of a great diversity of products and activities that are appealing enough to meet customers' expectations. People these days are getting more used to turn to new technologies when planning a trip. This reality can be explained by the fact that Internet is part of our daily life. For this purpose, several institutions and companies that offer varied touristic information about the destination have been set up.
Generally, the user starts from an elementary knowledge of the place to visit and of the places that can be of interest, either for the artistic, social or leisure value they may have. Besides this, each person can adopt different profiles depending on the trip they want to do or the circumstances that surround them; therefore, several

profiles can be distinguished, such as cultural, gastronomic, or family profile, among others.
Users can have access to a huge deal of data and information related to a specific place, but for sure is preferred to filter that information and get those elements or activities that match their profile or particular interests. Each of these profiles determines the different places to visit or the different ways to plan a trip. For instance, gastronomic travellers will put their culinary preferences in the first place; that is, the restaurant they want to eat, and will leave in the second place, the places and monuments to visit around it. For example, users who travel with children will avoid visiting many museums, and will consider practicing outdoor activities, such as gardens or amusement parks.
For the purpose of improving tourist experiences, *Recommender Systems* supply personalised information to users. In other words, the system selects the most suitable options from a large list of offers, by taking the users profile and interests into account. [5]

1.2 Recommender systems applications in Tourism

Since nowadays users can find a lot of information on the Internet, it may becomes a hard and complex task to select the information a user is interested in. The user is often unable to look through all the available information. Therefore, highly interesting information can get lost in the middle of a sea of data and that is why there are many kind of systems for searching information, the main three are: *search engines*, *helping systems* and *information filtering and retrieval systems*:

- *search engines*: systems that index files stored in web servers. Searches are made with key words or hierarchical trees based on different topics. The result of the searches is a list of URLs in which the topics related to the introduced key words appear. Usually, the vast majority of information that search engines offer does not suit the user interests. Examples are such as Google or Yahoo.

- *helping systems*: these systems teach the user how to use a particular program by describing it and explaining how it works. For instance, Microsoft includes a helping system that gathers information about the user characteristics and its actions, the state of the program, and the words that the user has searched, in order to calculate the probability of help that the user will need in related issues. These systems usually link web pages that provide additional information.

- *information filtering and retrieval systems*: the goal of these systems is to obtain the most relevant information for the user, minimizing the number of irrelevant items. Thus, filtering systems remove a huge amount of unwanted information. However, they could be more useful if they took user preferences into account with automatic learning techniques. If these systems were provided with artificial intelligence, they would supply users the content they are interested in, instead of just offering them a huge amount of information.

In the last two decades there has been a growth in interest in *Recommender Systems* whose are a specific type of information filtering technique that attempts to present information items (such as movies, music, news) that are likely of interest to the user.
The aim of Recommender Systems is to help *users* to find *items* that they should

appreciate from huge catalogues; the main characteristic of a Recommender Systems is the *personalization*, which goal is to provide users what they need without explicitly asking for it. The system can infer what the user demands not only according to the information that he initially provides, but also by comparing his profile with others users with a similar one and by considering his demographic data. Recommender Systems can be split in two macro-areas: Personalized and non-Personalized.

Chapter 2

Recommender Systems techniques

In Personalized Recommender Systems, it is possible to encounter different approaches: *collaborative filtering* (which is based on a set of user ratings on items) or *content-based* (which uses item content descriptions and user thematic profiles) but also a hybrid approach, while in the second does exist the *popularity-based* approach.

While collaborative filtering systems often result in better predictive performance, content-based filtering offers solutions to the limits of collaborative filtering, as well as a natural way to interact with the users. These complementary approaches thus motivate the design of hybrid systems.

2.1 Collaborative filtering

In collaborative filtering, the input to the system is a set of user ratings on items. Users can be compared based upon their shared appreciation of items, creating the notion of user neighbourhoods. Similarly, items can be compared based upon the shared appreciation of users, rendering the notion of item neighbourhoods. The item rating for a given user can then be predicted based upon the ratings given in her user neighbourhood and the item neighbourhood.

Collaborative filtering techniques are more often implemented than the Content-based and Hybrid and often result in better predictive performance. Three general approaches will be presented:

- user-based approaches associate a set of nearest neighbours with each user and then predict the user's rating for unscored items using the ratings given by the neighbours on that item [8];
- item-based approaches associate an item with a set of nearest neighbours, and then predict the user's rating for an item using the ratings given by the user on the nearest neighbours of the target item [9];
- model-based approaches, and more specifically those based on clustering, tend to be more scalable, by constructing a set of user groups or item groups, and then predicting a user's rating for an item using the mean rating given by the group members [10].

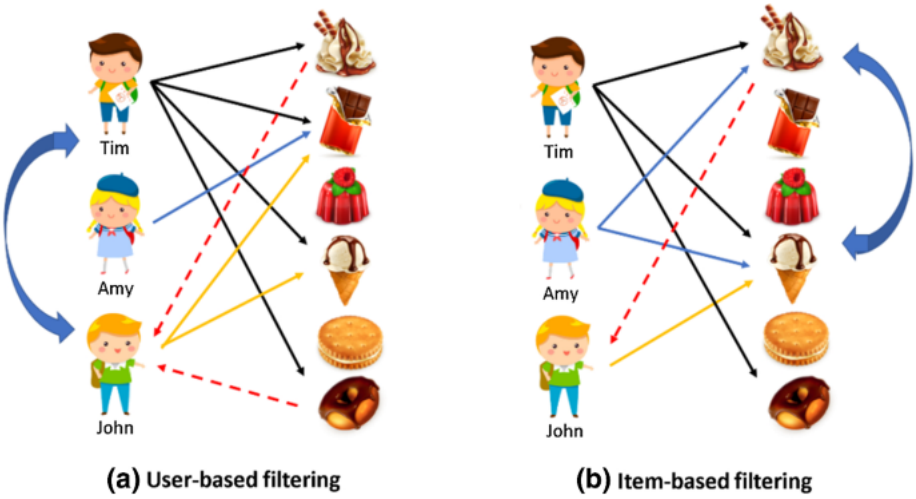

Figure 2.1. Graphical representation of Collaborative Filtering [58]

The definition of similarity between users and items is a key problem in each approach. While traditional similarity measures can be used, bespoke ones, which are tailored to type of data that is typically available (i.e. very sparse), tend to lead to better results.

Collaborative filtering systems, are based on users' preferences for items, which can carry a more general meaning than is contained in an item description.

On the other hand, collaborative filtering systems suffer from the cold start problem: they cannot make predictions on items that have not yet been rated by any user. The following approaches of Recommender System, Content-based, are able to handle such situations. These complementary approaches thus motivate the design of hybrid systems.

2.2 Content-based filtering

In the case of content-based filtering, however, item content descriptions are used to construct user thematic profiles that contain information about user preferences, such as, in the context of films, "like comedy and dislike war". The user's predicted appreciation of a given item is then based on the proximity between the item description and the user profile [11].

The input to content-based recommender systems is a set of item content descriptions, such as the genre, director and actors, in the context of films. Such techniques can also be divided into three general approaches:

- profiling information can be obtained from users explicitly, through questionnaires about their preferences for the item descriptions;
- user profiles may be built implicitly from user preferences for items, by searching for commonalities in liked and disliked item descriptions;
- user models may be learned implicitly by an automatic learning method, using item descriptions as input to a supervised learning algorithm, and producing user appreciations of items as output.

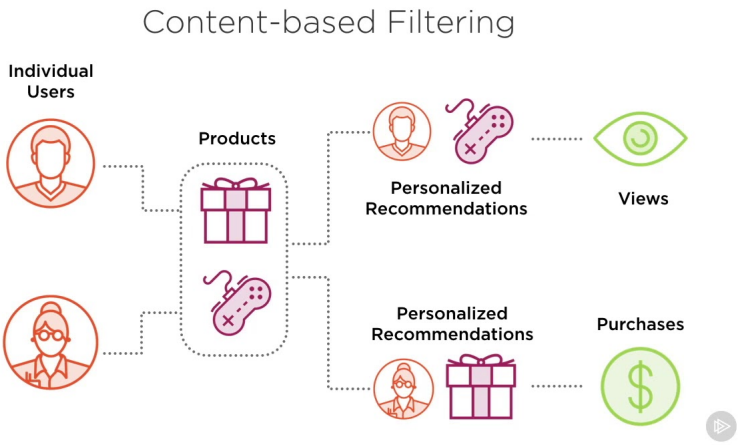

Figure 2.2. Graphical representation of Content Based Filtering [59]

User profiles are often represented as vectors of weights on item descriptions. Any other user model may be considered if an automatic learning method is used. If rule induction algorithms are used, user models may be of the type "if genre is action and actor is Schwerzenegger then film is liked" [28]. Finally, items that have a high degree of proximity to a given user's preferences would be recommended. To be efficient, content-based approaches need rich and complete descriptions of items and well-constructed user profiles. This is the main limitation of such systems. Since well-structured item descriptions are hard to come by in many domains, such approaches have mainly been applied in those where items are described by textual information, that can be parsed automatically, such as documents, web sites. Besides, content-based approaches can also suffer from overspecialisation [29].

That is, they often recommend items with similar content to that of the items already considered, which can lead to a lack of originality. On the other hand, privacy issues [30], such as users who do not want to share their preferences with others, are avoided.

2.3 Evaluation Metrics

Collaborative filtering seems to be more suitable as the core method of the recommender system while content-based filtering offers solutions to the limits of collaborative filtering, as well as a natural way to interact with the users. Indeed, users should be allowed to exert control over the system, thus building a meaningful relationship, and leading to psychological benefits such as the increase of trust in recommendations.

This brings naturally to the issue of evaluating the performance of a recommender system. Two methods can be used:

- error rate evaluation using cross-validation;
- user satisfaction evaluation.

For the first approach, many measures can be used to compare the results of different recommender systems. The most widely used ones are:

- Mean Absolute Error (MAE);
- Root Mean Squared Error (RMSE);
- Precision measures.

The first two measures evaluate the capability of a method to predict if a user will like or dislike an item, whereas the third measure evaluates its capacity to order a list of items based on user tastes. These measures thus carry different meanings [16]. In the first two cases, the method needs to be able to predict dislike, but there is no need to order items. In the last case, however, the method only focuses on items that users will like and the order in which these items are ranked is important. Beyond the importance of the predictive performance of recommender systems, other elements may be taken into consideration in their evaluation. The scalability of the proposed system is for example an important characteristic that needs to be taken into account. The coverage of a method, that is the proportion of recommendations it can provide, can also be considered. Finally, the system's ability to provide a level of confidence in a recommendation [18] and to explain why a recommendation was made [17] [19] can be used to define its potential interest to the user.
Evaluating a recommender system based on real users' opinions is also because, in many cases, recommending the set of items that maximise their predicted ratings does not necessarily lead to user satisfaction. For instance, users may estimate that such recommendations lack originality, or they may think that the proposed list of recommendations is not varied enough [20]. Users may also want to have some control over the system, rather than having little or no direct influence in the results.

2.4 Hybrid filtering

In the case of hybrid filtering, both types of information, collaborative and content-based, are exploited.

The first direct way to design a hybrid recommender system is to run independently a collaborative and a content-based one, and then combine their predictions. In [21], the combination is performed by forcing items to be, at the same time, close to the user thematic profile, and highly rated by her neighbours.
In [22], users are compared according to their content profiles, and the generated similarity measures are then used in a collaborative filtering system.
In [23] and [24], the rating matrix is enriched with predictions based on the content, and then a collaborative filtering is run. In [25], the similarity between items is computed by using their content descriptions as well as their associated rating vectors. Then an item-based collaborative filtering is used.
In [26], it is proposed to extend the prediction list of a collaborative filtering method to the items whose content is close to the recommended items. Based on the same idea, a content-based similarity between items is used in [27] in order to compare users not only according to their shared appreciations for some items, but by considering also their shared appreciations for items whose contents are similar.

A hybrid system can also be designed that follows a content-based filtering strategy and uses the data produced from collaborative filtering to enrich item similarity descriptions. At its core, it is a content-based filtering system that makes use of attribute similarities, similar to a personalised information retrieval system where requests are null ([7]). Items are recommended that are similar to the user's likes but not to their dislikes. The similarities between genres, nationalities and

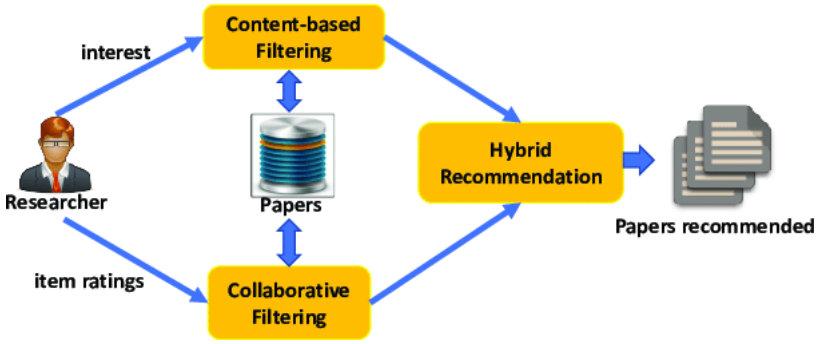

Figure 2.3. Graphical representation of Hybrid Filtering [60]

language attributes are defined by hand, and using the Cosine measure for directors and actors. Each film also contains a unique identification attribute. This identifier is compared to the films that have been previously noted by the user. The notion of similarity, for attributes of this type, is that embodied in the collaborative filtering algorithms. The more that a strictly collaborative filtering algorithm would recommend a film, the closer the film is to the user's profile. This type of hybrid system thus treats social data (found through collaborative filtering) as an attribute of an item, like any other. By weighting the importance of each characteristic, the system can vary from being purely content-based through to purely collaborative.

2.5 Non-Personalized Recommender Systems

Non personalized recommender systems are the most simple type of recommender systems. As suggested by the name, these type of recommender systems do not take into account the personal preferences of the users.
Non-personalized recommendations are usually where most sites start because it's easy and doesn't require that you know anything specific about the users. Non-personalized recommendations are good because you can always show those, despite how little you know about the users. People might say that non-personalized recs should only be shown until the system knows enough about the user to show more personalized recs, but always remember that humans are flock animals by nature, so most will be suckers for knowing what content items are the most popular—if for no other reason than to ensure what not to like [6].

2.6 Commercially adopted Recommender Systems

In this chapter will be presented some Recommender Systems adopted in commercial fields, such as *TripAdvisor*, *AirBnb*, *Amazon* and *Netflix* [14].

2.7 Deep Learning-based Models

In recent years, deep learning has yielded tremendous success across multiple domains, from image recognition to natural language processing. Recommender systems [15] have also benefited from deep learning's success. In fact, today's state-of-the-art recommender systems such as those at Youtube and Amazon are powered by complex deep learning systems, and less so on traditional methods. The

life-cycle of deep learning for recommendation can be split into two phases: training and inference. In the training phase, the model is trained to predict user-item interaction probabilities (calculate a preference score) by presenting it with examples of interactions (or non-interactions) between users and items from the past. Once it has learned to make predictions with a sufficient level of accuracy, the model is deployed as a service to infer the likelihood of new interactions.
Majority of today's Recommender Systems based on DL model are built on *implicit feedback*:

- *explicit feedback*: are direct and quantitative data collected from users. For example, Amazon allows users to rate purchased items on a scale of 1–10. These ratings are provided directly from users, and the scale allows Amazon to quantify user preference. Another example of explicit feedback includes the thumbs up/down button on YouTube, which captures users' explicit preference (i.e. like or dislike) of a particular video. The problem with explicit feedback is that they are rare. If you think about it, when was the last time you clicked the like button on a YouTube video, or rated your online purchases? Chances are, the amount of videos you watch on YouTube is far greater than the amount of videos that you have explicitly rated [31].

- *implicit feedback*: are collected indirectly from user interactions, and they act as a proxy for user preference. For example. videos that you watch on YouTube are used as implicit feedback to tailor recommendations for you, even if you don't rate the videos explicitly. Another example of implicit feedback includes the items that you have browsed on Amazon, which are used to suggest other similar items for you. The advantage of implicit feedback is that it is abundant. Recommender systems built using implicit feedback also allows us to tailor recommendations in real time, with every click and interaction. Today, online recommender systems are built using implicit feedback, which allows the system to tune its recommendation in real-time, with every user interaction.

Following are showed some examples of DL-models used for Recommendation Systems:

- *Neutral Collaborative Filtering*: The Neural Collaborative Filtering (NCF) model is a neural network that provides collaborative filtering based on user and item interactions. The NCF model treats matrix factorization from a non-linearity perspective. NCF TensorFlow takes in a sequence of (user ID, item ID) pairs as inputs, then feeds them separately into a matrix factorization step (where the embeddings are multiplied) and into a multilayer perceptron (MLP) network. The outputs of the matrix factorization and the MLP network are then combined and fed into a single dense layer which predicts whether the input user is likely to interact with the input item [32].

- *Variational Autoencoder for Collaborative Filtering*: An autoencoder neural network reconstructs the input layer at the output layer by using the representation obtained in the hidden layer. An autoencoder for collaborative filtering learns a non-linear representation of a user-item matrix and reconstructs it by determining missing values. The NVIDIA GPU-accelerated Variational Autoencoder for Collaborative Filtering (VAE-CF) is an optimized implementation of the architecture first described in Variational Autoencoders for Collaborative Filtering. VAE-CF is a neural network that provides collaborative filtering

based on user and item interactions. The training data for this model consists of pairs of user-item IDs for each interaction between a user and an item.
The model consists of two parts: the encoder and the decoder. The encoder is a feedforward, fully connected neural network that transforms the input vector, containing the interactions for a specific user, into an n-dimensional variational distribution. This variational distribution is used to obtain a latent feature representation of a user (or embedding). This latent representation is then fed into the decoder, which is also a feedforward network with a similar structure to the encoder. The result is a vector of item interaction probabilities for a particular user [33].

- *Wide and Deep*: Wide & Deep refers to a class of networks that use the output of two parts working in parallel—wide model and deep model—whose outputs are summed to create an interaction probability. The wide model is a generalized linear model of features together with their transforms. The deep model is a Dense Neural Network (DNN), a series of hidden MLP layers, each beginning with a dense embedding of features. Categorical variables are embedded into continuous vector spaces before being fed to the DNN via learned or user-determined embeddings.What makes this model so successful for recommendation tasks is that it provides two avenues of learning patterns in the data, "deep" and "shallow". The complex, nonlinear DNN is capable of learning rich representations of relationships in the data and generalizing to similar items via embeddings but needs to see many examples of these relationships in order to do so well. The linear piece, on the other hand, is capable of "memorizing" simple relationships that may only occur a handful of times in the training set. In combination, these two representation channels often end up providing more modeling power than either on its own. NVIDIA has worked with many industry partners who reported improvements in offline and online metrics by using Wide Deep as a replacement for more traditional machine learning models [34].

- *DLRM*: DLRM is a DL-based model for recommendations introduced by Facebook research. It's designed to make use of both categorical and numerical inputs that are usually present in recommender system training data. To handle categorical data, embedding layers map each category to a dense representation before being fed into multilayer perceptrons (MLP). Numerical features can be fed directly into an MLP [35].

  At the next level, second-order interactions of different features are computed explicitly by taking the dot product between all pairs of embedding vectors and processed dense features. Those pairwise interactions are fed into a top-level MLP to compute the likelihood of interaction between a user and item pair.

2.8 Amazon

Amazon's recommendation system is capable of intelligently analyzing and predicting customers' shopping preferences in order to offer them a list of recommended products.
n order to provide customers with accurate product recommendations, Amazon's algorithm must analyze huge amounts of data. In this way, it better understands the behavior of all users and the interests of each viewer. To do this, the recommendation engine collects two types of information, such as general data about products and

users and data on relations and dependencies between them. Getting to know the existing relationships in the online store will provide the recommendation engine with an insight into the real mechanisms governing customers' purchasing decisions. Amazon's recommendation algorithm analyzes 3 main types of dependencies and relationships for its operation:

- *User-product*: this type of relationship occurs when some users with certain characteristics prefer products of a given type and buy them more often.
- *Product-product*: product-product relationships occur when the products offered in the store are similar in terms of both appearance and specification.
- *User-user*: it occurs when individual customers with certain characteristics have similar tastes or preferences for certain products.

In addition to collecting information about relationships and connections Amazon's recommendation algorithm also uses different types of product and user data:

- *User behavior data*: this type of data is useful information about the preferences of individual customers, their interaction with the products in question. Amazon uses cookies to collect data about your browsing history, likes, or session length.
- *User Demographics data*: user demographics data is linked to personal information about individual customers, such as age, education, income and location. In order to collect such data, the user's consent is required.
- *Product Attribute Data*: product Attribute data is information related to the product itself, such as the specification of the computer, blouse size information, collection description.

2.9 Netflix

Netflix is a subscription service model that offers personalized recommendations, to help user to find shows and movies of interest. To do this, Netflix has created a proprietary, complex recommendations system.
80% of stream time is achieved through Netflix's recommender system, which is a highly impressive number. Moreover, Netflix believes in creating a user experience that will seek to improve retention rate, which in turn translates to savings on customer acquisition. Netflix utilises a two-tiered row-based ranking system, where ranking happens:

- *Within each row (strongest recommendations on the left)*
- *Across rows (strongest recommendations on top)*

Each row highlights a particular theme (e.g. Top 10, Trending, Horror, etc), and is typically generated using one algorithm. Each member's homepage consists of approximately 40 rows of up to 75 items, depending on the device the member is using.

The advantages can be seen from two perspectives:

- As a user, it is more coherent when presented a row of items that are similar, and then decide if he or she is interested in watching something in that category;

- As a company, it is easier to collect feedback as a right-scroll on a row would indicate interest whilst a scroll-down (ignoring the row) would indicate non-interest (not necessarily irrelevance).

Here is a summary of what Netflix models uses:

- *Personalised Video Ranking (PVR)*: this algorithm is a general-purpose one, which usually filters down the catalog by a certain criteria (e.g. Violent TV Programmes, US TV shows, Romance, etc), combined with side features including user features and popularity.
- *Top-N Video Ranker*: similar to PVR except that it only looks at the head of the rankings and looks at the entire catalog. It is optimised using metrics that look at the head of the catalog rankings (e.g. MAP@K, NDCG).
- *Trending Now Ranker*: this algorithm captures temporal trends which Netflix deduces to be strong predictors. These short-term trends can range from a few minutes a a few days. These events/trends are typically:
  - Events that have a seasonal trend and repeat themselves (e.g. Valentines day leads to an uptick in Romance videos being consumed);
  - One-off, short term events (e.g. Coronavirus or other disasters, leading to short-term interest in documentaries about them)
- *Continue Watching Ranker*: this algorithm looks at items that the member has consumed but has not completed, typically:
  - Episodic content (e.g. drama series);
  - Non-episodic content that can be consumed in small bites (e.g. movies that are half-completed, series that are episode independent such as Black Mirror).

  The algorithm calculates the probability of the member continue watching and includes other context-aware signals (e.g. time elapsed since viewing, point of abandonment, device watched on, etc).
- *Video-Video Similarity Ranker* (a.k.a. Because you watched (BYW)): this algorithm basically resembles that of a content-based filtering algorithm. Based on an item consumed by the member, the algorithm computes other similar items (using an item-item similarity matrix) and returns the most similar items. Amongst the other algorithms, this one is unpersonalised as no other side features are utilised. However, it is personalised in the sense that it is a conscious choice to display a particular item's similar items a member's homepage

Each of the above algorithms go through the row generation process seen in the image below. For example, if PVR is looking at Romance titles, it will find candidates that fit this genre, and at the same time come up with evidence to support the presentation of a row (e.g. previously watched Romance movies that the member has watched). From my understanding, this evidence selection algorithm is incorporated (or used together) in every other ranking algorithm listed above to create a more curated list ranking of items.

After the algorithms generate candidate rows (already ranked within each row vector), Netflix decide which of thesef rows to display focusing on not only accuracy, but also providing diversity, accessibility and stability at the same time. Other considerations include hardware capabilities (what device is being used) and which rows/columns are visible at first glance and upon scroll. Netflix wants to accurately predict what users want to watch in that session, but not forgetting that he/she might want to pick up on videos that were left off halfway.

The solution and approach that Netflix uses is a Machine Learning one, where they aim to create a scoring function by training a model using historical information of which homepages they have created for their members — including what they actually see, how they interacted with and what they played.

2.10 TripAdvisor

As the world's largest travel site, TripAdvisor provides a platform for billions of users to research, book, and review their trips across the world and provides travelers a collection of over 160,000 bookable experiences. TripAdvisor is a user-generated content website that "offers a plethora of reviews detailing travelers' experiences with hotels, restaurants, and tourist spots. TripAdvisor uses an item-based collaborative filtering model, which uses item-to-item cosine similarities based on page views. The Recommended for You (RFY) Model Architecture aggregates user's browsing history by taking a recency weighted average of the 100-dimensional item embeddings, followed by two fully connected layers with the final softmax output on 64,000 class probabilities, each of which corresponds to an experience that can be recommended.

2.11 AirBnb

A two-sided travel marketplace is an E-Commerce platform where users can both host tours or activities and book them as a guest. In Airbnb, the goal is to match people who are looking for accommodation — guests — with those looking to rent out their place — hosts. Guests reach out to hosts whose listings they wish to stay in, however a match succeeds only if the host also wants to accommodate the guest. Airbnb's search algorithm uses many factors to determine the order of listings in search results. It's designed to help guests quickly find places that meet their needs. The algorithm's recommendations reflect what it has learned from the millions of searches that have led to bookings in the past. The algorithm also takes into account specific details provided by the guest, such as desired destination, previous trips, and saved listings. Not every factor is weighed equally, and you don't need to have a luxurious space or an unbeatable location for your listing to rank well [36].

AirBnb uses automated machine learning with structural modeling technique for their system to provide effective recommendation. Its Recommender System is based on a search algorithm that supports deep learning and neural networks

AirBnb uses a knowledge graph which consists of a tree-structure taxonomy covering most generic concepts of tourism. These concepts are structured like a tree and the top nodes represent concepts that are more generic (e.g. Entertainment or Sports), and more specific concepts such as Comedy and Dance reside below Entertainment.
The second-level concept Dance can be further decomposed into Ballet, Disco and so on. A knowledge graph can be used to tag listings with concepts in the taxonomy.

For example, a new listing that is tagged with Ballet can be quickly categorized into Dancing and Sports.

2.12 Offline and Online Evaluation For Recommender Systems

Evaluation in recommender systems is divided in two main types, *offline evaluation* and *online evaluation* [37].
Before the beginning of the Offline evaluation, a collected and fixed dataset is used, it is divided into two parts, the first one is the train set and the second one is the test set. The system trains on the train set, and then it's evaluated over the test set.

Two types of datasets are often used in these evaluations, the first one is, natural or historical datasets, they are made up of data from the history of interactions of real users in a given system in a given period. This type uses explicit user ratings which can be collected directly or implicit user feedback which can be extracted from user history. The second type is the synthetic datasets which are constructed from artificial data. This type of dataset is usually used to test how recommendation algorithms work under certain conditions.
Offline evaluation has many advantages. It is economical and very quick to make large evaluations at the same time on several algorithms or datasets. In addition, since there are lots of already published datasets with their respective evaluations, researchers can easily evaluate and set up their algorithms by comparing their output with the expected one from the already published results. Also, in offline evaluation there is a fixed dataset and possible fixed user interactions with it, so comparing to online evaluations, the results of this type of evaluation is also reproducible in an easier way.

Although the offline analysis has several advantages, it has also some important disadvantages or weaknesses. Among them, the sparsity of ratings datasets can limit the items that can be evaluated, in this case we cannot evaluate the relevance of a recommended item to a user if we don't have the evaluation of this user to this element in the dataset.

Online evaluation is performed online with real users who are interacting with a recommender system, and they evaluate them by collecting associated metrics with user behavior in real time. This type of evaluation depends on many factors, for example, the user's intent to know the information which they specifically need, the user's context to measure the familiarity of the user with the items, and the interface where the recommendations are presented.

In opposite to offline evaluations, the online experiment has the possibility to collect real time user interaction that perform real tasks with the recommender system, like preferences, opinions, clicks, etc., which can help to have a good picture when evaluating the performance of the recommender system also, and will have more reliable results, so this is one of the advantages of online evaluation. In addition, having a real time data instead of statistic data, it is possible to give a deep analysis in several contexts.
Dynamic real time data have also some drawbacks in the evaluation. In order to maintain and deploy a system in a real life, it requires sufficient resources, like adequate user support and enough computational capacity, so in this case it will be very expensive. Moreover, to create or prepare the environment to test the recommender system, it is required to spend a massive amount of time to set it up.

In addition, the users over time get to know the system and any changes would be noticed so it will affect the results of the evaluation.

Chapter 3

Context of the project

Below are listed the tasks followed to be able to create the system; for first, it will be presented the tools used to develop this project, such as Google Colaboratory notebooks where to run the Python code, Gensim library to implement the LDA model, Ldavis library for graphical visualization; for second, there are the steps followed for the system.

- Used tools (3.1);
- General steps of the project (3.2).

3.1 Used tools

3.1.1 Python

Python [38] is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3.1.2 Pandas

Pandas [39] is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
Pandas allows to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

3.1.3 Google Colaboratory

Google Colaboratory [40] is an environment that allows to execute Jupyter Notebooks [41]. It was also ran on Google Drive, where data was saved and loaded

during the process.

3.1.4 Gensim

Gensim [42] is an open-source library for unsupervised topic modeling, document indexing, retrieval by similarity, and other natural language processing functionalities, using modern statistical machine learning.
The algorithms in Gensim, such as Word2Vec, FastText, Latent Semantic Indexing (LSI, LSA, LsiModel), Latent Dirichlet Allocation (LDA, LdaModel) etc, automatically discover the semantic structure of documents by examining statistical co-occurrence patterns within a corpus of training documents. These algorithms are unsupervised, which means no human input is necessary, just a corpus of plain text documents is needed.

Once these statistical patterns are found, any plain text documents (sentence, phrase, word...) can be succinctly expressed in the new, semantic representation and queried for topical similarity against other documents (words, phrases...).

Gensim is implemented in Python and Cython for performance. Gensim is designed to handle large text collections using data streaming and incremental online algorithms, which differentiates it from most other machine learning software packages that target only in-memory processing [43].

The core concepts of gensim are:

- A *Document*: some text;
- A *Corpus*: a collection of documents;
- A *Vector*: is a mathematical representation for each document;
- A *Model*: refers to a transformation from one document representation to another. In gensim documents are represented as vectors so a model can be thought of as a transformation between two vector spaces. The model learns the details of this transformation during training, when it reads the training Corpus.

.

Once the model is ready, it is possible to visualize different topics where each topic is a combination of keywords and each keyword contributes a certain importance to the topic. Since the basic Gensim visualization it is a kind of textual one: in the figure 8.1 a visualization example of three topics and the relative words.

Despite this visualization has all the necessary information, but it's difficult to understand it just by looking at a combination of words and numbers like above, it was chosen to use a package that represents graphically beeter topics and words: pyLDAvis.

3.1.5 pyLDAvis

PyLdavis is a Python library for interactive topic model visualization [44]. PyLDAvis is designed to help users interpret the topics in a topic model that has been fit to a corpus of text data. The package extracts information from a fitted LDA topic model to inform an interactive web-based visualization.

The visualization is intended to be used within an IPython notebook but can also be saved to a stand-alone HTML file for easy sharing.

```
[(0,
 '0.016*"car" + 0.014*"power" + 0.010*"light" + 0.009*"drive" + 0.007*"mount" '
 '+ 0.007*"controller" + 0.007*"cool" + 0.007*"engine" + 0.007*"back" + '
 '0.006*"turn"'),
(1,
 '0.072*"line" + 0.066*"organization" + 0.037*"write" + 0.032*"article" + '
 '0.028*"university" + 0.027*"nntp_post" + 0.026*"host" + 0.016*"reply" + '
 '0.014*"get" + 0.013*"thank"'),
(2,
 '0.017*"patient" + 0.011*"study" + 0.010*"slave" + 0.009*"wing" + '
 '0.009*"disease" + 0.008*"food" + 0.008*"eat" + 0.008*"pain" + '
 '0.007*"treatment" + 0.007*"syndrome"'),
(3,
 '0.013*"key" + 0.009*"use" + 0.009*"may" + 0.007*"public" + 0.007*"system" + '
 '0.007*"order" + 0.007*"government" + 0.006*"state" + 0.006*"provide" + '
 '0.006*"law"'),
```

Figure 3.1. Gensim LDA model textual topic visualization for Newsgroupc Panda's dataset

3.2 General steps of the project

This project aim to be applied on different kind of contexts: *TripAdvisor* and *AirBnb*, so for the first have been collected data for attractions, while for the second data on accomodations.
So for both contexts were performed different steps (each of these will be explained more in detail in the following chapters) on the data:

- Scraping of the data (3.2.1);
- Preprocessing of the data (3.2.2);
- Comparing models (3.2.3);
- Building the Recommender Systems (3.2.4);
- Recommender Systems results (3.2.5).

3.2.1 Scraping of the data

Data scraping, also known as web scraping, is the process of collecting information from a website. It's one of the most efficient ways to get data from the web and the extraction of web data has the goal give data in a format that is more useful for the user.
To collect data to work with, it was used Selenium framework [45].
Selenium is well-known as an open-source testing framework for web applications.

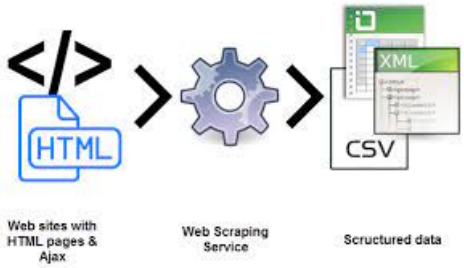

Figure 3.2. Web-scraping example [61]

3.2.2 Preprocessing of the data

Once data was collected, each user reviews needed to be manipulated in order to be available to process.
Text cleaning or Text pre-processing is a mandatory step when working with text in Natural Language Processing (NLP). In real-life human writable text data contain various words with the wrong spelling, short words, special symbols, emojis, etc. so the need is to clean this kind of noisy text data before feeding it to the machine learning model. Data preprocessing can refer to the manipulation or removal of data before it is used to ensure or improve performance. It is an important step in the data mining process. This stage is, however, essential, and the quality of the analyzed data for conducting an aspect-based sentiment approach. Basic Natural Language Processing methods were used to pre-process the data, such as:

- *Tokenization*;
- *Removing punctuation*;
- *Stemming and lemmatization*;
- *Stop word removal*;
- *Removing hashtags*;
- *Irrelevant words removal.*

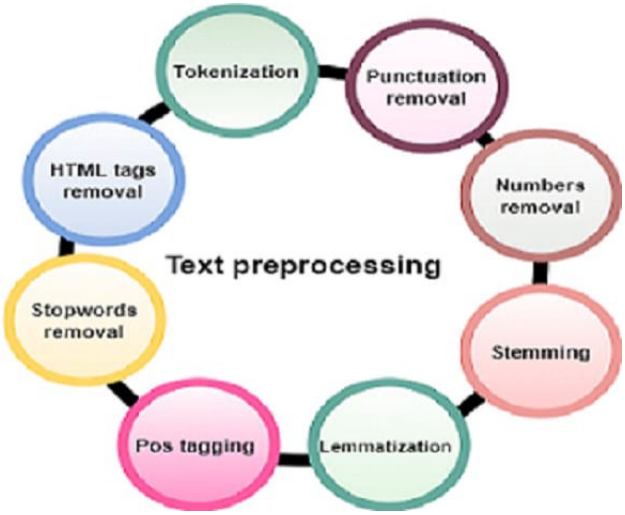

Figure 3.3. Data preprocess flow [62]

Tokenization

Given a character sequence and a defined document unit, *tokenization* is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. Here is an example [47] of tokenization:

- input: *Friends, Romans, Countrymen, lend me your ears;*
- output tokens: Friends | Romans | Countrymen | lend | me | your | ears.

These tokens are often loosely referred to as terms or words, but it is sometimes important to make a type/token distinction. A *token* is an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing. A *type* is the class of all tokens containing the same character sequence. A *term* is a (perhaps normalized) type that is included in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens, for instance, they could be semantic identifiers in a taxonomy, but in practice in modern IR systems they are strongly related to the tokens in the document. However, rather than being exactly the tokens that appear in the document, they are usually derived from them by various normalization processes.

Removing punctuation

This step involves removing all punctuation from the text. In Python «string» library, there is a predefined list of punctuation marks. In this project it was used the Natural Language Toolkit [46], which is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries [48].

Stemming and lemmatization

Stemming and Lemmatization are algorithms that are used in Natural Language Processing (NLP) to normalize text and prepare words and documents for further processing in Machine Learning.
Also known as the text normalization stage, in which words are reduced to their root or elemental form. A Porter stemmer package is used for stemming, which reduces inflectional and derived word forms to a common base form. Since documents are going to use different forms of a word, such as *organize*, *organizes*, and *organizing*. Additionally, there are families of derivationally related words with similar meanings, such as *democracy*, *democratic*, and *democratization*. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For example [49]:

- *am, are, is* ⇒ *be*;
- *car, cars, car's, cars'* ⇒ *car*.

The result of this mapping of text will be something like:

- *the boy's cars are different colors the boy car be differ color*.

However, the two words differ in their flavor. *Stemming* usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. *Lemmatization* usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*. If confronted with the token *saw*, stemming might return just *s*, whereas lemmatization would attempt to return either *see* or *saw* depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Linguistic processing for stemming or lemmatization is often done by an additional plug-in component to the indexing process, and a number of such components exist, both commercial and open-source. The most common algorithm for stemming English, and one that has repeatedly been shown to be empirically very effective, is the Porter's algorithm (Porter, 1980) [12]. This algorithm consists of 5 phases of word reductions, applied sequentially. Within each phase there are various conventions to select rules, such as selecting the rule from each rule group that applies to the longest suffix. In the first phase, this convention is used with the following rule group:

Rule	Example
SSFS ⇒ SS	caresses ⇒ caress
IES ⇒ I	ponies ⇒ poni
SS ⇒ S	caress ⇒ caress
SS ⇒	cats ⇒ cat

Many of the later rules use a concept of the *measure* of a word, which loosely checks the number of syllables to see whether a word is long enough that it is reasonable to regard the matching portion of a rule as a suffix rather than as part of the stem of a word.

Stop word removal

The concept of stop words has a long history with Hans Peter Luhn credited with coining the term in 1960 (Luhn 1960 [13]). Stop words are available in abundance in any human language. By removing these words, low-level information will be removed from text in order to give more focus to the important information and because they don't add any value to the analysis. These words have little or no meaning. The Python NLTK library contains a list of English stop words. There are several of them, such as: (i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your).
Stop words can have different roles in a corpus. They can generally be categorized into three groups: *global*, *subject*, and *document* stop words. Global stop words are words that are almost always low in meaning in a given language; these are words such as "of" and "and" in English that are needed to glue text together. These words are likely a safe bet for removal, but they are low in number.
Next up are subject-specific stop words. These words are uninformative for a given subject area. Subjects can be broad like finance and medicine or can be more specific like obituaries, health code violations, and job listings for librarians in Kansas. Words like *bath*, *bedroom*, *entryway* are generally not considered stop words in English, but they may not provide much information for differentiating suburban house listings and could be subject stop words for certain analysis. Usually these words need to be manually constructed. These kinds of stop words may improve performance.

Lastly, we have document-level stop words. These words do not provide any or much information for a given document. These are difficult to classify and won't be worth the trouble to identify. Even if you can find document stop words, it is not obvious how to incorporate this kind of information in a regression or classification task. It was also performed the task of pulling a string of length 'n' from the reviews; in this project, were removed strings with n = 2 or less to keep only the words.

Irrelevant words removal

In addition to eliminating common stop words, eliminating irrelevant terms is a beneficial task in the pre-processing process, which involves eliminating noisy terms such as names of attractions, links, and proper nouns [50].

Removing Hashtags

To clean text from *hashtags*, the # char was removed.

Bug of Words

The bag-of-words model [51] is a way of representing text data when modeling text with machine learning algorithms. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- *A vocabulary of known words*;
- *A measure of the presence of known words.*

Results after Preprocessing

Below it follows two figures with an example of text before the preprocessing steps (Figure 3.4) and after the preprocessing steps (Figure 3.5).

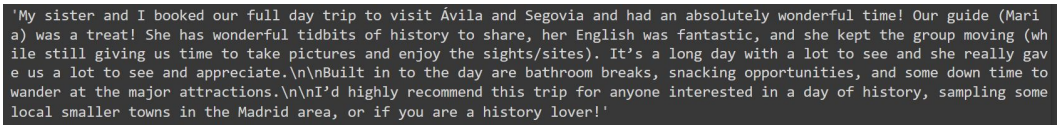

Figure 3.4. Example of a text before preprocessing

Figure 3.5. Example of a preprocessed text

3.2.3 Comparing models

After data was preprocessed and ready to be used, for each review, the pivotal column to be work on, has the most importand words splitted as a list.
Some basic concepts will be addressed below.

Basic Concepts

Topic modeling [54] is a method for unsupervised classification of documents, similar to clustering on numeric data, which finds some natural groups of items (topics) and it provides methods for automatically organizing, understanding, searching, and summarizing large electronic archives.

Latent Dirichlet Allocation (LDA) model

Definition

In natural language processing, Latent Dirichlet Allocation (LDA) is a generative statistical model that explains a set of observations through unobserved groups, and each group explains why some parts of the data are similar. The LDA is an example of a topic model. In this, observations (e.g., words) are collected into documents, and each word's presence is attributable to one of the document's topics. Each document will contain a small number of topics [54].

Terminology

Here follows some main terms definintion, such as *words*, *documents* and *corpus* [55].

- A *word* is the basic unit of discrete data, defined to be an item from a vocabulary indexed by $\{1, ..., V\}$, words are represented by using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus, using superscripts to denote components, the v-th word in the vocabulary is represented by a V-vector w such that $w^v = 1$ and $w^u = 0$ for $u \neq v$;
- A *document* is a sequence of N words denoted by w $= w_1, w_2, ..., w_N$ where w_n is the n-th word in the sequence;
- A *corpus* is a collection of M documents denoted by D $= \{w_1, w_2, ..., w_M\}$.

LDA Mallet model

Mallet module allows both LDA model estimation from a training corpus, using an (optimized version of) collapsed gibbs sampling.
Gensim library provides a wrapper to implement Mallet's LDA from within Gensim itself.

Mallet includes sophisticated tools for document classification: efficient routines for converting text to "features", a wide variety of algorithms (including Naïve Bayes, Maximum Entropy, and Decision Trees), and code for evaluating classifier performance using several commonly used metrics.
In addition to classification, MALLET includes tools for sequence tagging for applications such as named-entity extraction from text. Algorithms include Hidden Markov

```
[(0,
  '0.075*"mora" + 0.066*"tour" + 0.041*"madrid" + 0.029*"wanted" + 0.022*"bit" + 0.022*"hours" + 0.021*"end" + 0.019*"dare" + 0.018*"booked" + 0.018*"feel" + 0.016*"thought" +
0.016*"tourist" + 0.015*"things" + 0.015*"time" + 0.015*"wife" + 0.014*"thing" + 0.013*"happy" + 0.013*"couple" + 0.011*"friend" + 0.010*"week"'),
 (1,
  '0.125*"bus" + 0.025*"hop" + 0.025*"stop" + 0.015*"city" + 0.014*"back" + 0.014*"stops" + 0.014*"buses" + 0.013*"found" + 0.013*"route" + 0.013*"ticket" + 0.011*"commentary" +
0.011*"money" + 0.010*"day" + 0.009*"times" + 0.009*"tickets" + 0.009*"full" + 0.009*"people" + 0.008*"traffic" + 0.008*"bought" + 0.008*"open"'),
 (2,
  '0.180*"food" + 0.098*"madrid" + 0.068*"history" + 0.067*"local" + 0.032*"cultura" + 0.030*"places" + 0.026*"jorge" + 0.025*"delicious" + 0.021*"restaurants" +
0.015*"knowledge" + 0.014*"eat" + 0.011*"andres" + 0.011*"learned" + 0.011*"spots" + 0.010*"daniel" + 0.010*"drinks" + 0.010*"devour" + 0.008*"pedro" + 0.008*"alfonso" +
0.007*"foods"'),
 (3,
  '0.169*"day" + 0.077*"trip" + 0.063*"toledo" + 0.037*"full" + 0.031*"beautiful" + 0.025*"driver" + 0.020*"worth" + 0.015*"comfortable" + 0.015*"half" + 0.015*"town" +
0.013*"time" + 0.013*"cathedral" + 0.013*"lunch" + 0.012*"architecture" + 0.011*"monuments" + 0.009*"view" + 0.009*"weather" + 0.008*"views" + 0.008*"nice" + 0.007*"oscar"'),
 (4,
  '0.059*"art" + 0.058*"museum" + 0.057*"prado" + 0.028*"history" + 0.024*"tours" + 0.023*"knowledge" + 0.017*"paintings" + 0.016*"hernan" + 0.014*"maria" + 0.014*"life" +
0.013*"artists" + 0.012*"satt" + 0.012*"works" + 0.011*"piece" + 0.009*"work" + 0.008*"background" + 0.008*"visit" + 0.008*"highlights" + 0.008*"articolo" +
0.007*"collection"'),
 (5,
  '0.082*"show" + 0.059*"good" + 0.023*"flamenco" + 0.022*"dinner" + 0.021*"amazing" + 0.020*"small" + 0.019*"dancers" + 0.015*"performance" + 0.015*"passion" +
0.013*"experience" + 0.011*"place" + 0.011*"staff" + 0.010*"felt" + 0.009*"feel" + 0.009*"outstanding" + 0.009*"performers" + 0.008*"warm" + 0.008*"stage" + 0.008*"put" +
0.008*"service"'),
 (6,
  '0.439*"tour" + 0.104*"guida" + 0.080*"history" + 0.036*"made" + 0.028*"arantxa" + 0.023*"explained" + 0.022*"loved" + 0.018*"knowledge" + 0.014*"knew" + 0.014*"highly" +
0.011*"spain" + 0.011*"gave" + 0.010*"explaining" + 0.007*"terrific" + 0.006*"detail" + 0.006*"appreciated" + 0.005*"enjoyable" + 0.004*"detailed" + 0.004*"engaging" +
0.004*"attention"'),
 (7,
  '0.064*"group" + 0.057*"night" + 0.049*"evening" + 0.046*"tapas" + 0.041*"people" + 0.040*"food" + 0.033*"bars" + 0.030*"drinks" + 0.030*"drink" + 0.025*"fun" + 0.024*"great"
+ 0.023*"places" + 0.021*"james" + 0.020*"bar" + 0.017*"eat" + 0.015*"madrid" + 0.015*"met" + 0.013*"host" + 0.012*"restaurants" + 0.011*"variety"'),
 (8,
  '0.172*"knowledgeable" + 0.110*"guida" + 0.087*"tour" + 0.080*"excellent" + 0.070*"friendly" + 0.038*"highly" + 0.036*"extremely" + 0.032*"questions" + 0.029*"funny" +
0.028*"helpful" + 0.020*"enjoyable" + 0.019*"professional" + 0.018*"personable" + 0.016*"entertaining" + 0.015*"pleasant" + 0.012*"engaging" + 0.012*"incredibly" +
0.012*"answered" + 0.012*"answer" + 0.011*"recommended"'),
```


Figure 3.6. Mallet model - textual example of word importance in each topic

Models, Maximum Entropy Markov Models, and Conditional Random Fields. These methods are implemented in an extensible system for finite state transducers.
The MALLET topic modeling toolkit contains efficient, sampling-based implementations of Latent Dirichlet Allocation, Pachinko Allocation, and Hierarchical LDA.
In the Figure 3.6, as an example, there are shown the top 10 important words for each topic in *TripAdvisor* context for *Madrid*, while in Figure 3.7, there are visual representations for the 10 most important words in each of the 20 topics.

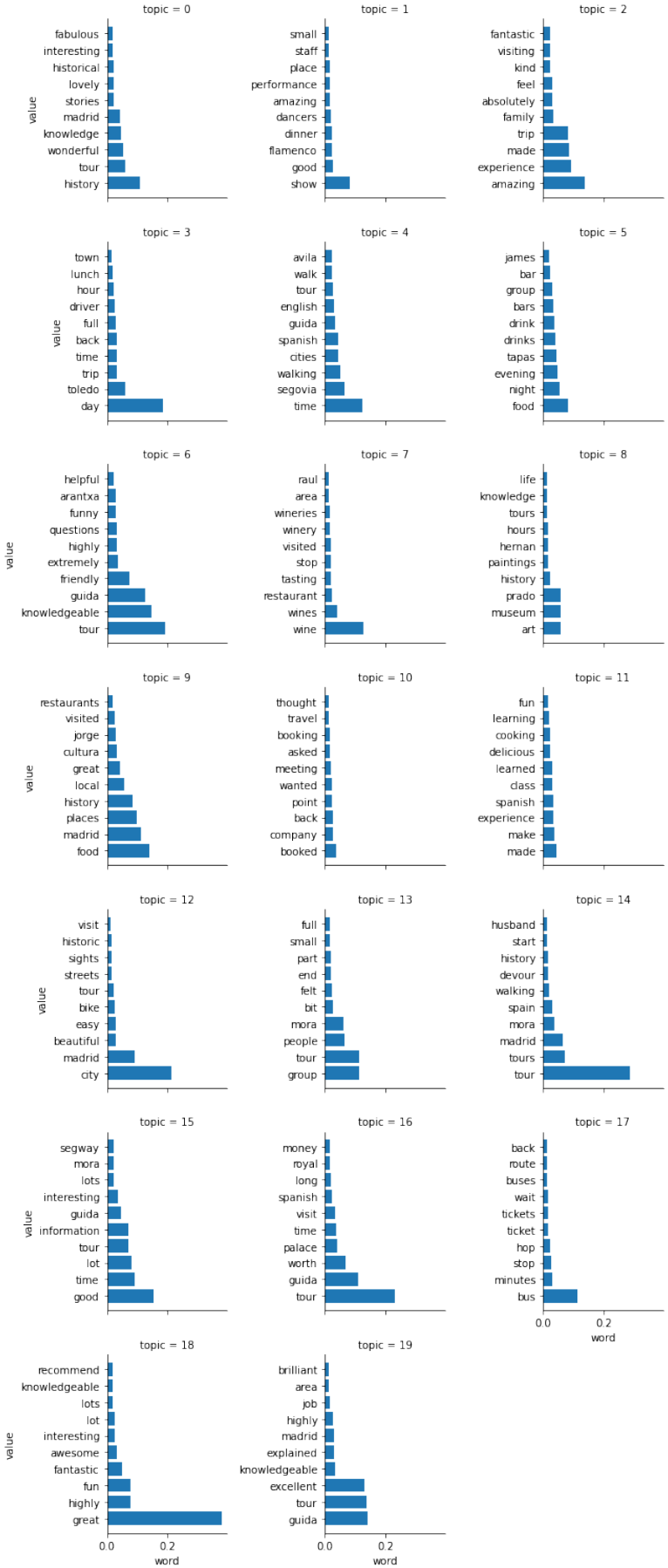

Figure 3.7. Mallet model - visual example of word importance in each topic

Latent Semantic Indexing (LSI) model

Latent Semantic Indexing, also known as latent semantic analysis [56], is a mathematical practice that helps classify and retrieve information on particular key terms and concepts using singular value decomposition (SVD).
Singular Value Decomposition (SVD) of a matrix is a factorization of that matrix into three matrices. It has some interesting algebraic properties and conveys important geometrical and theoretical insights about linear transformations.
When analysing a string of words, LSI removes conjunctions, pronouns, and common verbs, also known as stop words. This isolates the words which comprise the main 'content' of a phrase.
These words are then placed in a Term Document Matrix (TDM), which is a 2D grid that lists the frequency that each specific word (or term) occurs in the documents within a data set. Weighing functions are then applied to the TDM. A simple example is classifying all documents that contain the word with a value of 1 and all that don't with a value of 0. When words occur with the same general frequency in these documents, it is called co-occurrence.
The following figures are two examples of words in the topics, the first one is a textual representation while the second one is a more understandable visual representation. Using SVD allows us to approximate the patterns in word usage across all documents. The SVD vectors produced by LSI predict meaning more accurately than analysing individual terms. Ultimately, LSI can use the relationships between words to better understand their sense, or meaning, in a specific context.
In the Figure 3.8, as an example, there are shown the most important words for each topic in *TripAdvisor* context for *Madrid*, while in Figure 3.9, there are visual representations for the 10 most important words in each of the 20 topics.

Figure 3.8. LSI model - textual example of word importance in each topic

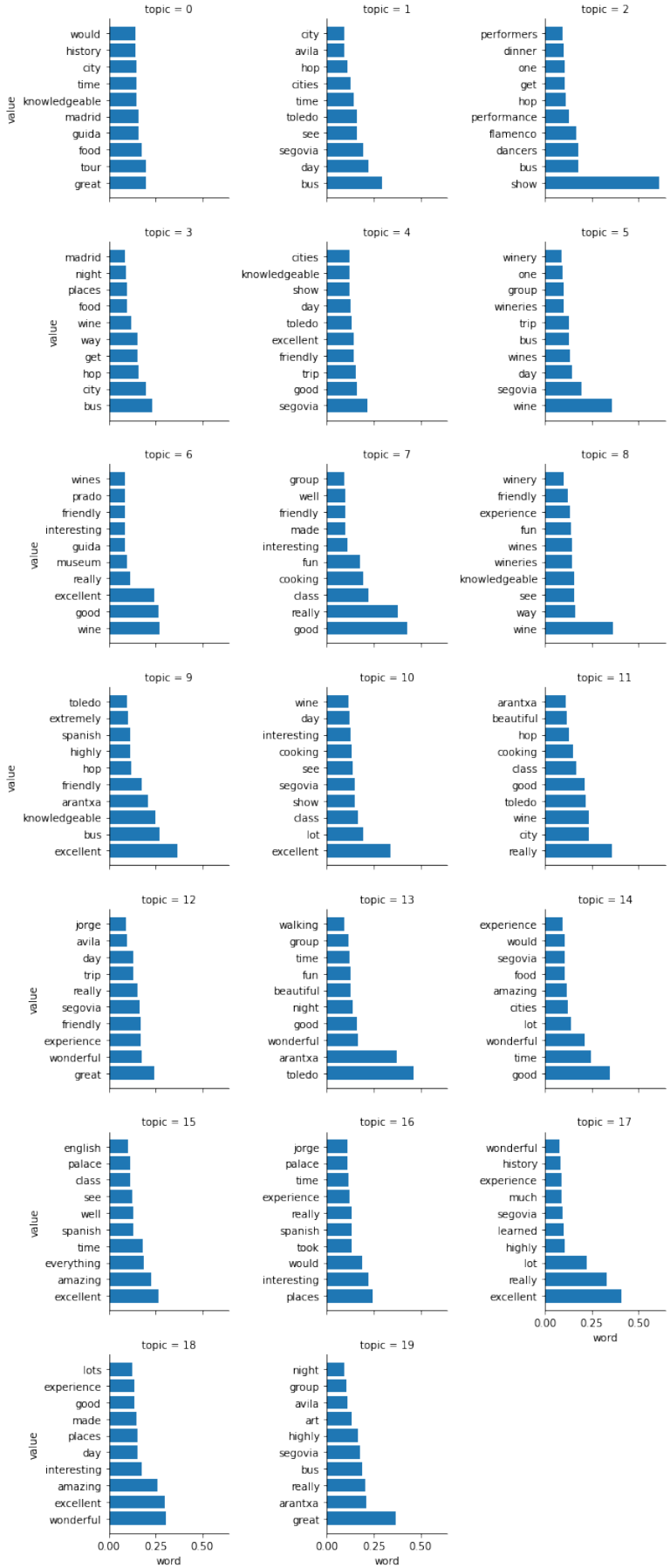

Figure 3.9. LSI model - visual example of word importance in each topic

Models' scores comparison

To compare models, they were used two different measures obtained by each one of the presented models, *perplexity* and *coherence* [57]:

- *perplexity*: perplexity measure is a statistical measure of how well a probability model predicts a sample and is one of the intrinsic evaluation metric, and is widely used for language model evaluation. It captures how surprised a model is of new data it has not seen before, and is measured as the normalized log-likelihood of a held-out test set;
- *coherence*: is used for assessing the quality of the learned topics. Coherence score measures the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference. For this reason was chosed coherence score to compare models with different number of topics. Higher the topic coherence, the topic is more human interpretable.

Following tables are an example of results based on coherence score for *Madrid* city in *TripAdvisor* context of each model, based on number of topics:

Number of topics	Coherence Score
10	0.5010791842
15	0.5557138915
20	0.4977870038
30	0.4971665614
40	0.4557507582

Table 3.1. LDA model

Number of topics	Coherence Score
10	0.5232462021
15	0.4893962586
20	0.4894383027
30	0.4508120032
40	0.4443470839

Table 3.2. Mallet model

Number of topics	Coherence Score
10	0.4446341388
15	0.4051284653
20	0.3804685314
30	0.3592382276
40	0.3385757743

Table 3.3. LSI model

Once the scores were obtained after the grid-search, the best one was chosen as the model for the Recommender System, LDA model in this case.

3.2.4 Building the System

After the best model was set based on the best coherence score, LDA model was used in the System and it was possible to start performing several steps in order to obtain the recommendations for each user that gave a feedback, a certain number of recommendations (five in this case).

Datasets

For each context, datasets were divided by city:

- *Barcelona*;
- *Berlin*;
- *Madrid*;
- *Rome*;
- *London*.

In the Figure 3.13 is shown an example of the first rows in the TripAdvisor Madrid dataset. It is possibile to see the columns contained in it:

- *userName*: the account name of the user;
- *userUrl*: the url to the user profile;
- *reviewDate*: the date user gave the review;
- *userText*: the text user wrote as a description of the review;
- *review*: the vote user gave to the attraction;
- *host_id*: the id of the host;
- *review_id*: the id of the review;
- *userText_processed*: the processed text after preporcessing step.

userName	userUrl	reviewDate	usertext	review	host_id	review_id	usertext_processed
Rori F	https://www.tripadvisor.com/Profile/Greyhoundm...	September 9, 2022	My sister and I booked our full day trip to vi...	5.0	TZ917HUF8HP1	CF5D3AGFD1PEB5GIR7	['sister', 'booked', 'full', 'day', 'trip', 'v...
Maps516996	https://www.tripadvisor.com/Profile/Maps516996	July 3, 2022	My daughters and I thoroughly enjoyed our full...	5.0	TZ917HUF8HP1	X9ENIHJ8DJZ37R8PLM	['thoroughly', 'full', 'day', 'tour', 'avila',...
Diana M	https://www.tripadvisor.com/Profile/dianam770	February 28, 2022	Very nice tour of Avila and Segovia. The guide...	4.0	TZ917HUF8HP1	P7Z22GGTQUX1LXMMX0	['tour', 'avila', 'segovia', 'guida', 'knowled...
Dantecass	https://www.tripadvisor.com/Profile/Dantecass	October 30, 2022	I'll start with the "cons" because I felt ther...	3.0	TZ917HUF8HP1	NIWFY6KEORG7U3OW9U	['start', 'cons', 'felt', 'mora', 'cons', 'pro...
neillay51	https://www.tripadvisor.com/Profile/neillay51	October 30, 2022	The trip was not well organised. The microphon...	2.0	TZ917HUF8HP1	MC8LA0P1WZH36OK01R	['trip', 'well', 'microphone', 'receiving', 'd...

Figure 3.10. First rows of the TripAdvisor Madrid dataset

For each city, the core feature was the description of each user review, each of which in the bug of word format, ready to be worked on.

Gensim LDA method parameters

Once the LDA model was chosen, to build it, it was used the *LdaMulticore* method of the Gensim library, using all CPU cores to parallelize and speed up model training. The mentioned method needs some parameters to be specified, such as:

- *corpus*;
- *id2word*;
- *number of topics*;
- *passes*;
- *random state*.

```
lda_model = gensim.models.LdaMulticore(corpus = corpus,
                                       id2word = id2word,
                                       num_topics = num_topics,
                                       passes = num_passes,
                                       random_state = random_state)
```

Figure 3.11. Gensim function to build LDA model and its parameters

If the *corpus* was already addressed earlier, it is useful to talk about the other parameters and on performed steps: for *id2word*, which is a dictionary of the format *(int, str)*, is the mapping from word IDs to words. It is used to determine the vocabulary size.
For the suitable number of topics to choose, it was performed a grid-search approach for this *hyperparameter* (a parameter which is defined before the algorithm is run): *Grid search* is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on a the specific parameter values of a model.
The numbers used for the grid-search were 10, 15, 20, 30 and 40 number of topics and, at the end, each model with different number of topics was compared each other based on their *coherence* and *perplexity*; a good number of topics minimizes the perplexity compared to other numbers of topics.

model	perplexity	coherence
topics_10_nr_passes_15	-6.992077	0.521622
topics_20_nr_passes_15	-7.109424	0.498731
topics_40_nr_passes_15	-7.366013	0.461510

Figure 3.12. Example of Coherence and Perplexity scores for a model with different number of topics

The *passes* parameter represents the number of how many times the algorithm is supposed to pass over the whole corpus. In this model, it was set to 15.

The *random state* parameter serves as a seed (needed for reproducibility).

Once the models were ran and the number of topics was fine-tuned, the model with the higher coherence score was chosen to work with.

Below there is a visual representation example, through pyLDAvis package, of the obtained topics and relative words. Figure 3.13 is a representation of the model topics: each bubble represents a topic. The larger the bubble, the more prevalent or dominant the topic is. Good topic model will be fairly big topics scattered in different quadrants rather than being clustered on one quadrant. Also, the further the bubbles are away from each other, the more different they are.
In Figure 3.14, blue bars represent the overall frequency of each word in the corpus. If no topic is selected, the blue bars of the most frequently used words will be displayed, while red bars, as in Figure 3.15, give the estimated number of times a given term was generated by a given topic. In this example, there are about 19,000 of the word *tour*, and this term is used about 1,500 times within topic 2. The word with the longest red bar is the word that is used the most in that topic.

For each city of the contexts, they were generated plots for topics and words.

Topic sentiment analysis

Sentiment analysis aims to identify opinions, emotions, and evaluations expressed in natural language. The main objective is to predict the sentiment orientation (i.e., positive, neutral, or negative) by analysing the sentiment or opinion words and expressions in sentences and documents; this approach involves analysing extracted topic words through a lexicon-based sentiment analysis approach to identify the emotional state associated with each dimension.

Figure 3.13. Example of a LDA model with 20 topics

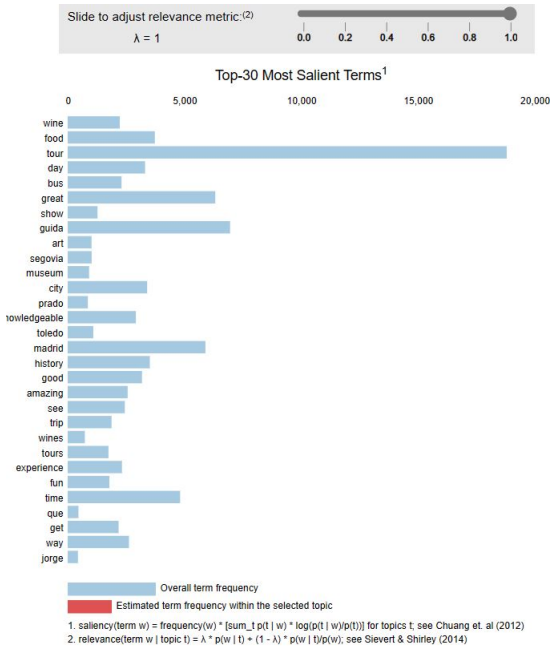

Figure 3.14. Example of a LDA model words related to the 20 topics

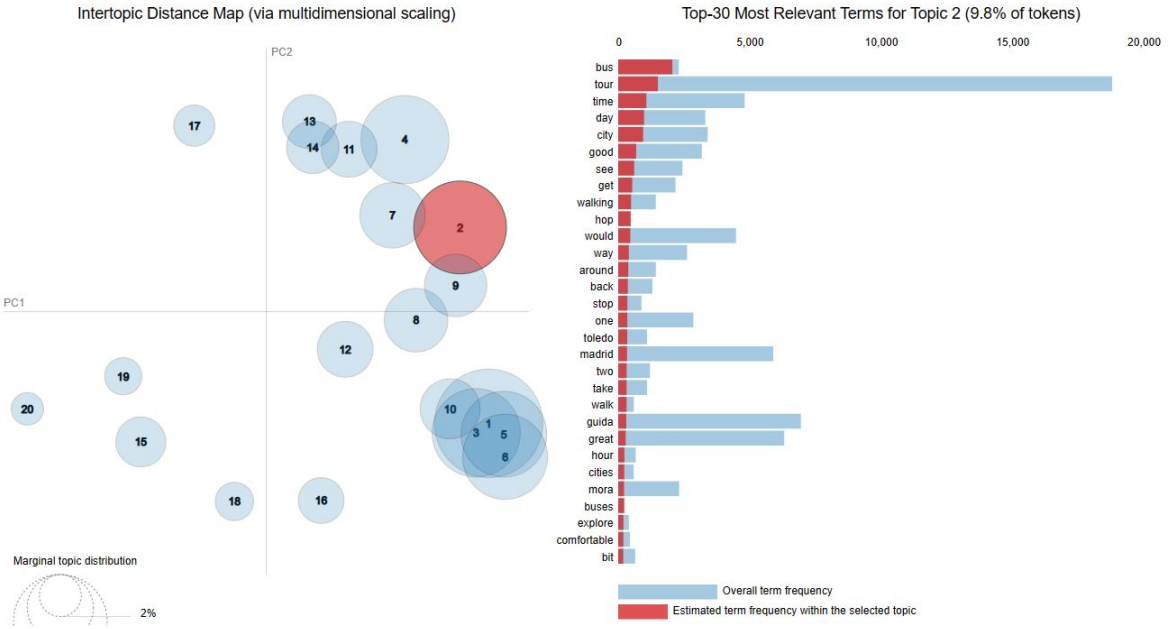

Figure 3.15. Topic 2 selected and its related words

Once the topics were obtained, it began the process to build the Recommender System, in order to provide recommendations for each user.
The main idea was to find, for each user, the most similar 5 places/attractions to recommend the user could like. They were taken into consideration the *top-k* most significant words of each user, with *k=5*. Also, they were taken into consideration descriptions with at least 3 significant words, in order to avoid short descriptions or the ones with not significant meaning.

Creating the System

The system was made up by different steps:

- *vectorization*;
- *tf-idf*;
- *recommendations*.

Vectorization

Vectorization is a classic approach of converting input data from its raw format (i.e. text) into vectors of real numbers which is the format that ML models support. This approach has been there ever since computers were first built, it has worked wonderfully across various domains, and it's now used in NLP. In Machine Learning, vectorization is a step in feature extraction. The idea is to get some distinct features out of the text for the model to train on, by converting text to numerical vectors.

Tf-idf

The *tf-idf* stands for term frequency–inverse document frequency and it is a popular approach in text mining and information retrieval and it allows to get a number representation of how important a word is across a set of documents. TF-IDF defines importance of a term by taking into consideration the importance of that term in a single document, and scaling it by its importance across all documents.
The formula can be defined as:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{3.1}$$

where t is the term, d the document and D the set of documents. The term frequency is defined as:

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3.2}$$

Term frequency tells how many times does a word appear in a document among the number of times all words appear in that document, while the inverse document frequency is:

$$idf(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3.3}$$

Inverse document frequency tells how common (or uncommon) is a word among all the documents.
TF-IDF is a popular approach used to weigh terms for NLP tasks because it assigns

a value to a term according to its importance in a document scaled by its importance across all documents in your corpus, which mathematically eliminates naturally occurring words.

3.2.5 Recommendations

At this point the Recommender System gives, for each user were taken into account the most top-5 significant words (the words that have the higher weight in the document, based on tf-idf score) and it was possible to merge all the previously mentioned techniques to ran the Recommender System and to obtain recommendations.
So at the end, each user has a list of 5 host id attractions (in the case of TripAdvisor context) or places (for AirBnb one), having the format as shown in the next Figure 3.16:

userName	recommended_host_id
Rori F	['XYF21MJK6I92', 'DGL5NIEXC4R0', 'BXP67Z8AO2OU', 'AYKQQTCXPM2H']
Maps516996	['XAR4XINAHGJO', 'XAR4XINAHGJO', 'RYC0VLYC9RM8', 'RYC0VLYC9RM8']
Diana M	['TZ917HUF8HP1', 'XKTGVCJZWW21', '4DHEKK03Z3B7', 'XKTGVCJZWW21']
Dantecass	['YT6IWEYXLJZ6', 'XYF21MJK6I92', 'V3XL8ZLBZWC0', '2BGXTN0Q1F9E']
neillay51	['BXP67Z8AO2OU', 'XYF21MJK6I92', '2BGXTN0Q1F9E', 'TZ917HUF8HP1']

Figure 3.16. Example of the results obtained by the Recommender System using a LDA model in the TripAdvisor context

The column *recommended host id* represents the ID of the attractions/places the user specified in the column *userName* could like.
Then, having these IDs as a source, there were printed the titles of the attraction (as shown in the next Figure 3.17).

```
                    Full-Day Toledo Tour with Cathedral from Madrid
Madrid Walking Tour and The Royal Palace with Skip the Line Tickets
    Toledo, Segovia, Optional Avila: Majesty of Medieval Spain Tour
                                Segway Ride in the Old City of Madrid
```

Figure 3.17. Example of the resulting attraction titles

Once the recommendations were obtained, to have a complete overview, they were calculated the most-3 recommended attractions.
In Figure 3.18) is showed a bar plot where is possible to see the frequencies of the attractions: the most recommended attraction, with id *TZ917HUF8HP1* and title *Full Day Tour Ávila and Segovia from Madrid with Tickets to Monuments Included*, was suggested 9197 times, while id *BL0BFMAJVCCB* 6101 times, meaning that about 50% less then the first one.
In Table 36 is possible to see the names of the attraction given its host_id.

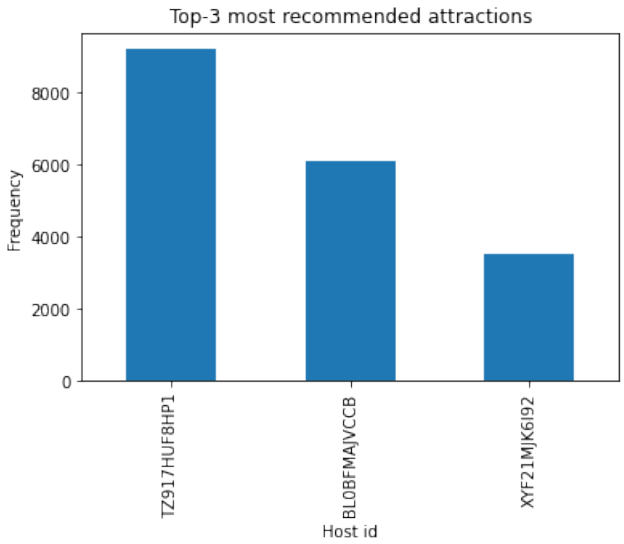

Figure 3.18. Frequencies of top-3 most recommended attractions for *TripAdvisor - Madrid*

Host_id	Title
TZ917HUF8HP1	Full Day Tour Ávila and Segovia from Madrid with Tickets to Monuments Included
BL0BFMAJVCCB	Madrid Tapas and Wine Tasting Tour
XYF21MJK6I92	Full-Day Toledo Tour with Cathedral from Madrid

Table 3.4. Example of attractions' names

3.3 Item-item collaborative filtering Recommender System

As [53] a comparative System, it was build an item-item collaborative filtering System, belonging to the family of item-based Recommender Systems.
Item-item collaborative filtering is one kind of recommendation method which looks for similar items based on the items users have already liked or positively interacted with. It was developed by Amazon in 1998 and plays a great role in Amazon's success. Rather than matching the user to similar customers, item-to-item collaborative filtering matches each of the user's purchased and rated items to similar items, then combines those similar items into a recommendation list. So at its core item-item is all about finding items similar to the ones user has already liked.
The very first step is to build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.

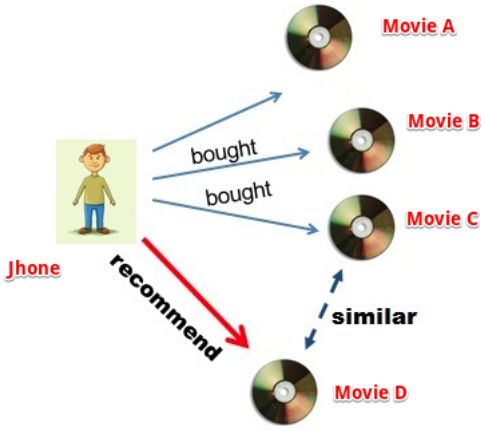

Figure 3.19. Example of item-item technique [63]

In this case, were taken into account not the feedback description as for the topic-model based RS but the ratings: in the Figure 3.20 is represented a plot to check the correlation between the average rating and the number of ratings.
It is possible to see an upward trend from the scatter plot, showing that popular movies get higher ratings. The average rating distribution shows that most attractions in the dataset have an average rating of around 5.5. The number of rating distribution shows that most movies have less than 20 ratings.
The following are the steps used to build the System:

- Calculate item similarity scores based on all the user ratings;
- Identify the top n items that are most similar to the item of interest;
- Calculate the weighted average score for the most similar items by the user;
- Rank items based on the score and pick top n items to recommend.

In the Figure 3.21 there is an example of results obtained with this System: for each *user* there is a list of five recommended *host* ids.

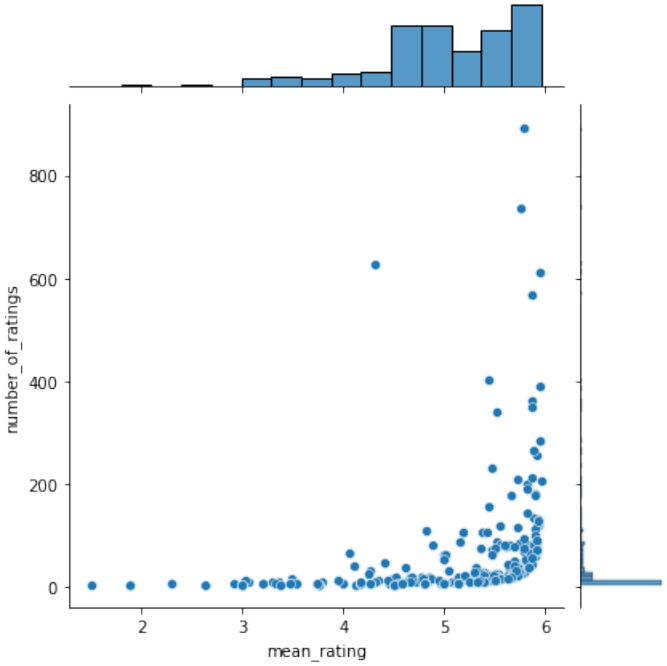

Figure 3.20. Example correlation between the average rating and the number of ratings for Madrid in *TripAdvisor* context.

Also for the item-item Recommender Systems, once recommendations were obtained, the Plot 3.22 shows the most 3 suggested attractions for *Madrid*.
Then getting the title (3.23) from the attraction id, it is possible to see that attraction with id *04TVIK996ZQ9* has the title *Half Day Trip to Toledo from Madrid* and it was suggeste about 2400 times.

userName	recommended_host_id
-Bromley4321-	7NI8IIHDTJFH, 8WLL7GRDWD23, DGL5NIEXC4R0, G4BR...
003marie	0NS6EQSLM3YM, 0UP0UEPM2BX7, 28TAZMP1UV6J, 2ET9...
0713sandela	0NS6EQSLM3YM, 0UP0UEPM2BX7, 28TAZMP1UV6J, 2ET9...
1010195910102012	0UP0UEPM2BX7, 18N7NLWZJ8N2, 2JWIJAGF6BKR, 3ACW...
1010Susan	2IILF0H2456H, 4J5NTKVP9WC8, 5P6YOPJRGXX6, 6MAY...

Figure 3.21. Example of recommended attractions list for users based on item-item collaborative filtering.

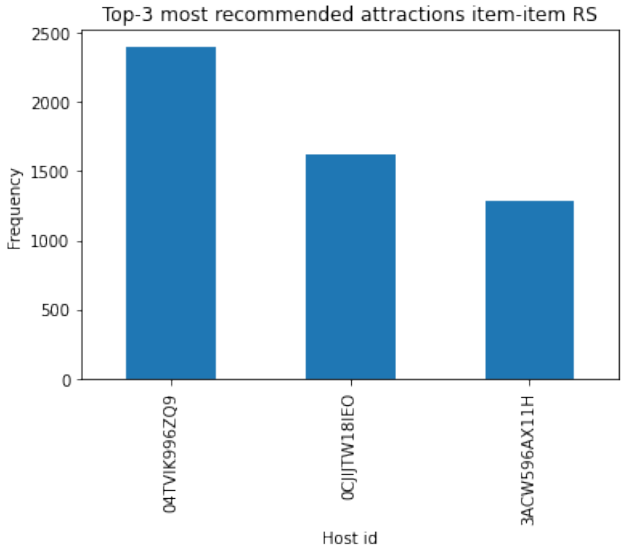

Figure 3.22. Example of the top-3 most recommended attractions for Madrid in *TripAdvisor* context.

```
                                  Half Day Trip to Toledo from Madrid
Madrid Walking Tour and The Royal Palace with Skip the Line Tickets
                 Prado Museum Entrance Ticket with In-App Audio Guide
```

Figure 3.23. Titles of top-3 most recommended attractions for Madrid in *TripAdvisor* context.

3.4 Experiment

The experiment was used to compare the performance of the proposed approach and existing approach. Performance of the Systems are tested on data coming from TripAdvisor. In order to evaluate the performance, the proposed System based on LDA and the item-item collaborative filtering method with k nearest neighbor is examined.

The proposed approach is implemented using the TripAdvisor datasets (one dataset for each city where the attractions are located), which consists of likes users have given to attractions they visited. Then, from original dataset, they were created three different sub-datasets by filtering users that has given at least 5, 10 and 15 likes respectively. Each of this three sub-dataset was divided into train (80%) data and test (20%) data.

Datasets are made of a list of the users' likes, each of which consists of information of the user, such as the username, the date when the like was released, the description and the rating.

In Latent Dirichlet Allocation, similar documents belong to same topic and each topic is a mixture of a certain number of words (in Gensim library, 20 words by default). So in the proposed System, similar reviews belong to same topic as well, based on the words present in a review description that users have given; using the model built on the train set data, in order to check which recommendations to suggest for a user in the test set, it was taken into consideration to which topic the unseen document (review description) is more similar and considering the topics that have at least a probability of 0.5 to be similar. This was done by invoking the Gensim library method and passing as a parameters the unseen document (review description) and the minimum probability, obtaining as a result a list of closest topics. Then, from the most similar topic of the list, predicted rating for the active user was computed based on the given ratings by the other users for reviews with the greatest probability in the selected topic. The predicted rating for a recommendation was needed to perform the MAPE score, which is explained below.

For the comparison between the Systems and measure their accuracy, *Mean Absolute Percentage error (MAPE)* [52] was used: is a metric that defines the accuracy of a forecasting method. It represents the average of the absolute percentage errors of each entry in a dataset to calculate how accurate the forecasted quantities were in comparison with the actual quantities. MAPE is often effective for analyzing large sets of data.

In this case, for each user of the test set, MAPE was calculated through the:

$$\text{MAPE} = \frac{100\%}{n} \sum_{t}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

where At is the actual value and Ft is the forecast value.

To study more in details the performances of the Systems, it was chosen to take into consideration three different kind of users with a specific condition: users that has given at least 5, 10 and 15 ratings respectively and, for each these users, it was created a list of 3, 5 and 10 recommendations and their ratings.

The choice of 5, 10 and 15 amounts was taken in order to better investigate the difference in recommendations' accuracy based on users preferences; when not available users with at least 15 rated attractions, it was chosen a reasonably close amount, e.g. 10 or 12 as available in the datasets.

3.4.1 Experiment results

The experiment was ran on different datasets, belonging to different cities of the context. The detailed MAPE results of the two Systems for Barcelona are shown in Figure (3.24) and in Figure (3.25) below.
Figures show that when a number of recommendations is RM=3 the minimum value of MAPE is 22.71% where rated at least 15 attractions. When a number of recommendations is RM=5, the minimum value of MAPE is 25.02% where rated at least 15 attractions and the minimum value of MAPE is 28.16% where rated at least 15 attractions when RM=10.

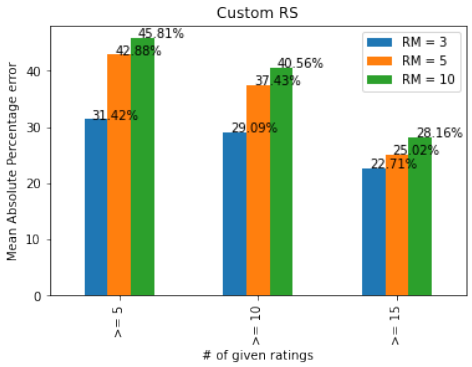

Figure 3.24. Barcelona - Custom RS

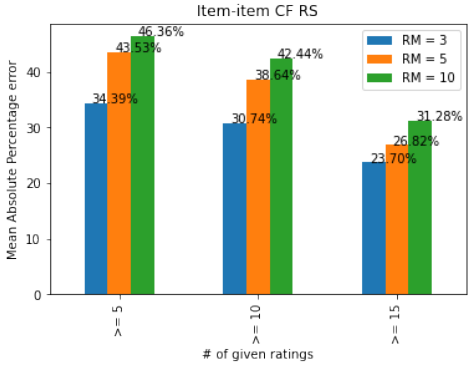

Figure 3.25. Barcelona - Item-item CF RS

MAPE results of the two Systems for Madrid are shown in Figure (3.26) and in Figure (3.27) below. Figures show that when a number of recommendations is RM=3 the minimum value of MAPE is 22.22% where rated at least 12 attractions. When a number of recommendations is RM=5, the minimum value of MAPE is 25.16% where rated at least 15 attractions and the minimum value of MAPE is 28.21% where rated at least 12 attractions when RM=10.

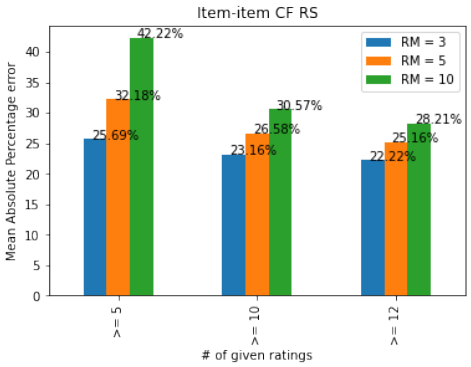

Figure 3.26. Madrid - Custom RS

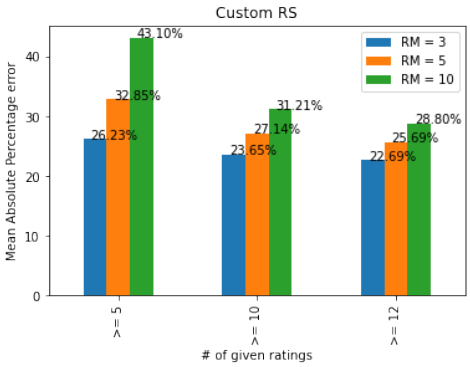

Figure 3.27. Madrid - Item-item CF RS

MAPE results of the two Systems for Berlin are shown in Figure (3.28) and in Figure (3.29) below.
Figures show that when a number of recommendations is RM=3 the minimum

value of MAPE is 25.54% where rated at least 10 attractions. When a number of recommendations is RM=5, the minimum value of MAPE is 31.29% where rated at least 10 attractions. In this case, in the dataset was not present a user which has given more than 10 ratings.

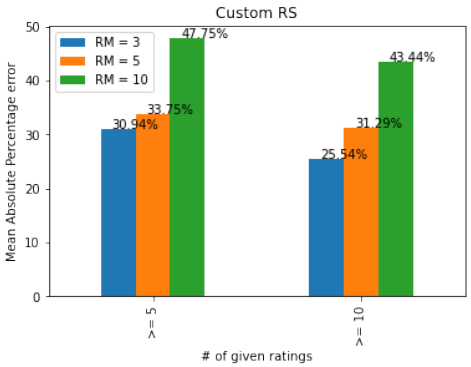

Figure 3.28. Berlin - Custom RS

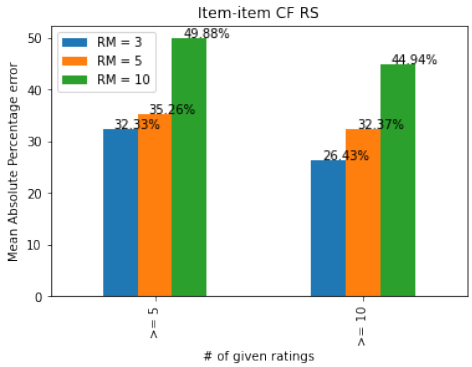

Figure 3.29. Berlin - Item-item CF RS

MAPE results of the two Systems for Rome are shown in Figure (3.30) and in Figure (3.31).
Figures show that when a number of recommendations is RM=3 the minimum value of MAPE is 21.22% where rated at least 15 attractions. When a number of recommendations is RM=5, the minimum value of MAPE is 23.66% where rated at least 15 attractions and the minimum value of MAPE is 36.82% where rated at least 15 attractions when RM=10.

MAPE results of the two Systems for London are shown in Figure (3.32) and in Figure (3.33).
Figures show that when a number of recommendations is RM=3 the minimum value of MAPE is 25.47% where rated at least 15 attractions. When a number of recommendations is RM=5, the minimum value of MAPE is 29.76% where rated at least 15 attractions and the minimum value of MAPE is 33.69% where rated at least 15 attractions when RM=10.

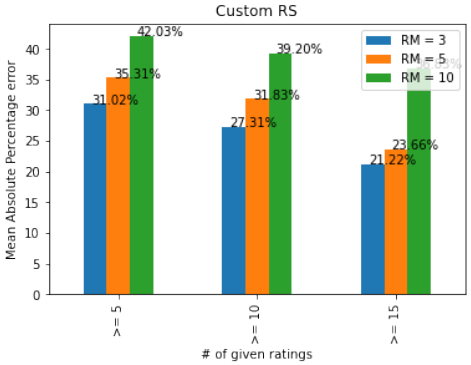

Figure 3.30. Rome - Custom RS

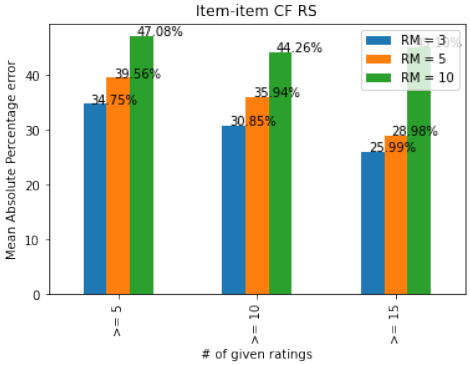

Figure 3.31. Rome - Item-item CF RS

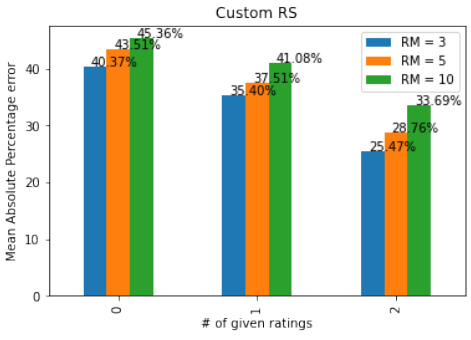

Figure 3.32. London - Custom RS

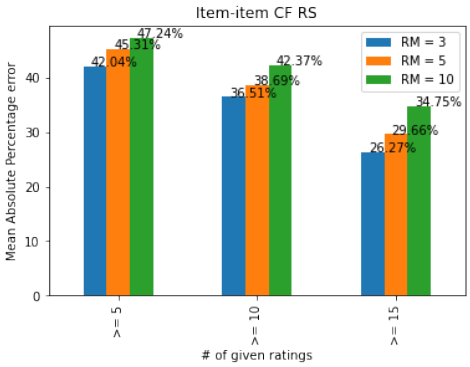

Figure 3.33. London - Item-item CF RS

in addition, using collaborative filtering to generate recommendations is computationally expensive. It is *O(MN)* in the worst case, where M is the number of customers and N is the number of items, since it examines M customers and up to N items for each customer. However, because the average customer vector is extremely sparse, the algorithm's performance tends to be closer to *O(M + N)*.
Scanning every customer is approximately *O(M)*, not *O(MN)*, because almost all customer vectors contain a small number of items, regardless of the size of the catalog. But there are a few customers who have rated a significant percentage of the catalog, requiring *O(N)* processing time. Thus, the final performance of the algorithm is approximately *O(M + N)*.
Data sparsity is seen as a key disadvantage of collaborative filtering. Typically, it can cause the cold start problem that describes the difficulty of collaborative filtering making recommendations when the users or the products are new, since the operation of CF is based on historical data of site interactions between users and items.

3.4.2 Conclusions

After several experiments were performed, in terms of the mean absolute percentage error used to calculate, the best result was found through the proposed Recommender System, for users that rated at least 15 attractions with three attractions recommended to the user. The mean value result of MAPE was 23,43% obtained by the proposed System, while for the item-item CF System the mean value result of MAPE was 25,1%.
Also, it results that growing the number of given ratings by a user, the more accurate are the given recommendations by the Systems, thanks to greater amount of information about the user itself. On the other hand, it could be more difficult to give an accurate recommendation if there are few information, as for example, if the user rated just one attraction.

Bibliography

[1] Introduction. https://link.springer.com/article/10.1007/s42979-021-00592-x.

[2] Introduction. https://ieeexplore.ieee.org/document/8910312. https://link.springer.com/article/10.1007/s42979-021-00592-x

[3] ActiveTopic. https://ieeexplore.ieee.org/document/8910312.

[4] Latent Dirichlet allocation (LDA). https://dl.acm.org/doi/10.5555/944919.944937.

[5] Tourism. https://www.researchgate.net/publication/284725884_TOURISM_IN_SCIENTIFIC_RESEARCH, https://www.mdpi.com/1999-5903/13/1/2.

[6] Non-Personalized Recommender Systems. https://livebook.manning.com/book/practical-recommender-systems/chapter-5/12.

[7] Jack, K., Duclay, F. (2007). Etude de la pertinence de critères de recherche en recherche d'informations sur des données structurées. In PeCUSI, INFORSID (pp. 285-297). Perros-Guirec, France.

[8] User based. https://www.researchgate.net/publication/323156230_User_Participation_in_Collaborative_Filtering-Based_Recommendation_Systems_A_Game_Theoretic_Approach.

[9] Item based. https://realpython.com/build-recommendation-engine-collaborative-filtering/ 'Item-Based Explanations for User-Based Recommendations' by Marius Kaminskas Frederico Durao and Derek Bridge Department of Computer Science Insight Centre for Data Analytics University College Cork Ireland

[10] Model based https://www.researchgate.net/publication/221338420_Designing_Specific_Weighted_Similarity_Measures_to_Improve_Collaborative_

[11] Content based. http://lcandillier.free.fr/publis/ICDM08.pdf.

[12] Porter, Martin F. 1980. An algorithm for suffix stripping. Program 14 (3): 130-137.

[13] uhn, H. P. 1960. "Key Word-in-Context Index for Technical Literature (kwic Index)." American Documentation 11: 288–295. https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.5090110403.

[14] Amazon. https://www.amazon.science/the-history-of-amazons-recommendation-algorithm, https://recostream.com/blog/amazon-recommendation-system, TripAdvisor https://www.tripadvisor.com/engineering/personalized-recommendations-for-experiences-using-deep-learning, https://publications.computer.org/intelligent-systems/2017/10/16/httpswww-computer-orgintelligent-systems20171016tripadvisor-recommendation-algorithm-for-social-media-google/, Netflix https://help.netflix.com/en/node/100639#:~:text=We%20estimate%20the%20likelihood%20that,preferences%20on%20our%20service%2C%20and, https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48, AirBnb https://dl.acm.org/doi/pdf/10.1145/3397271.3401444?casa_token=o5CHZEvHhaIAAAAA:iv_YnTdzBY8LAnydmcRtQ8JmaxGWwFptn4ihQMmFqwG-s-z9cyUF0uRsW9FAFFOh1YfzDmmkA6WE ,

https://medium.com/airbnb-engineering/how-airbnb-uses-machine-learning-to-detect-host-preferences-18ce07150fa3#:~:text=For%20each%20search%20query%20that,prominently%20in%20the%20search%20results, https://medium.com/@alexandra.gg150/how-to-build-a-recommender-system-for-airbnb-in-python-3a92ad500fa5.

[15] Balabanovic Shoham, 1997. https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401.

[16] McNee, S., Riedl, J., Konstan, J. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems.

[17] Herlocker et al., 2000) Herlocker, J., Konstan, J., Riedl, J. (2000). Explaining collaborative filtering recommendations. In ACM Conference on Computer Supported Cooperative Work.

[18] Basu, C., Hirsh, H., Cohen, W. W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In 15th National Conference on Artificial Intelligence (pp. 714–720).

[19] Bilgic, M. (2004). Explanation for Recommender Systems: Satisfaction vs. Promotion. PhD thesis, University of Texas at Austin, Department of Computer Sciences.

[20] Ziegler, C.-N., McNee, S., Konstan, J., Lausen, G. (2005). Improving recommendation lists through topic diversification. In 14th International World Wide Web Conference (pp. 22–32).

[21] Balabanovic Shoham, 1997. https://www.semanticscholar.org/paper/Fab%3A-content-based%2C-collaborative-Section%3A-Systems)-Balabanovic-Shoham/8dd8dd27a73d63135e9f2cfaf39e2b4339b394c8.

[22] Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. In Artificial Intelligence Review, 13(5-6), 393–408.

[23] Polcicova, G., Slovak, R., Navrat, P. (2000). Combining content-based and collaborative filtering. In ADBIS-DASFAA Symposium (pp. 118-12.

[24] Melville, P., Mooney, R., Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In 18th National Conference on Artificial Intelligence (pp. 187-192).

[25] Vozalis, M., Margaritis, K. G. (2004). Enhancing collaborative filtering with demographic data: The case of item-based filtering. In 4th International Conference on Intelligent Systems Design and Applications (pp. 361–366).

[26] Han, E.-H. S., Karypis, G. (2005). Feature-based recommendation system. In 14th Conference of Information and Knowledge Management (pp. 446-452).

[27] Wang, J., de Vries, A.P., Reinders, M.J. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In 29th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 501-508).

[28] https://link.springer.com/article/10.1007/s40747-020-00212-w.

[29] Zhang, Y., Callan, J., Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. In ACM SIGIR'02.

[30] Lam, S. K., Frankowski, D., Riedl, J. (2006). Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. In International Conference on Emerging Trends in Information and Communication Security.

[31] https://towardsdatascience.com/deep-learning-based-recommender-systems-3d120201db7e.

[32] https://towardsdatascience.com/paper-review-neural-collaborative-filtering-explanation-implementation-ea3e031b7f96.

[33] Variational Autoencoders for Collaborative Filtering. Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, Tony Jebara.

[34] https://paperswithcode.com/method/wide-deep.

[35] https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-2-deep-learning-for-recommender-systems/.

[36] https://arxiv.org/pdf/1810.09591.pdf.

[37] https://re.public.polimi.it/retrieve/e0c31c11-c51f-4599-e053-1705fe0aef77/2105.06275v1.pdf.

[38] https://www.python.org/doc/essays/blurb/.

[39] https://pandas.pydata.org/.

[40] https://colab.research.google.com/.

[41] https://en.wikipedia.org/wiki/Project_Jupyter.

[42] https://radimrehurek.com/gensim/intro.html.

[43] https://en.wikipedia.org/wiki/Gensim.

[44] https://pyldavis.readthedocs.io/en/latest/readme.html.

[45] https://www.selenium.dev/

[46] https://www.nltk.org/

[47] https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

[48] https://www.geeksforgeeks.org/python-remove-punctuation-from-string/

[49] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[50] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9672446/

[51] https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[52] https://hcis-journal.springeropen.com/articles/10.1186/s13673-018-0161-6#Tab4

[53] https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d

[54] https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2

[55] https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

[56] https://en.wikipedia.org/wiki/Latent_semantic_analysis

[57] https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

[58] https://www.researchgate.net/figure/Concepts-of-user-based-and-item-based-filtering_fig1_340361119

[59] https://www.bisa.ai/portofolio/detail/MjQ

[60] https://analyticsindiamag.com/a-guide-to-building-hybrid-recommendation-systems-for-beginners/

[61] https://www.hgday360.com/post/web-scraping-extraer-datos-de-sitios-web-din%C3%A1micos

[62] https://basilkjose.medium.com/data-preprocessing-natural-language-competition-processing-dcbbf9d014e8

[63] https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d

[64] European Southern Observatory, http://www.eso.org

@INPROCEEDINGS9693063, author=Kamal, Maryam and Chatzigiannakis, Ioannis, booktitle=2021 International Conference on Innovative Computing (ICIC), title=Influential Factors for Tourist Profiling for Personalized Tourism

Recommendation Systems– A Compact Survey, year=2021, volume=, number=, pages=1-6, doi=10.1109/ICIC53490.2021.9693063 @articlebourg2021enhancing, title=Enhancing shopping experiences in smart retailing, author=Bourg, Lorena and Chatzidimitris, Thomas and Chatzigiannakis, Ioannis and Gavalas, Damianos and Giannakopoulou, Kalliopi and Kasapakis, Vlasios and Konstantopoulos, Charalampos and Kypriadis, Damianos and Pantziou, Grammati and Zaroliagis, Christos, journal=Journal of Ambient Intelligence and Humanized Computing, pages=1–19, year=2021, publisher=Springer Berlin Heidelberg

@articleCHATZIGIANNAKIS2011103, title = Urban pervasive applications: Challenges, scenarios and case studies, journal = Computer Science Review, volume = 5, number = 1, pages = 103-118, year = 2011, issn = 1574-0137, doi = https://doi.org/10.1016/j.cosrev.2010.09.003, url = https://www.sciencedirect.com/science/article/pii/S1574013710000444, author = Ioannis Chatzigiannakis and Georgios Mylonas and Andrea Vitaletti, keywords = Pervasive, Sensor network, RFID, NFC, Participatory, Games, Challenges, Scenarios, abstract = In this work, we discuss various aspects of the application of pervasive technologies inside an urban setting. In the last decade we have seen the emergence of a multitude of closely-related pervasive technologies that have only recently started to materialize on a grand scale, such as wireless sensor networks, RFID and NFC. We discuss the arising research challenges associated with such converging fields and we also provide a survey of the state-of-the-art related application scenaria, which we believe set their near-future applied context. Finally, we provide a more analytic discussion on three discrete systems that belong to this category of applications and give insight to the current state-of-the-art work in this field.