

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

RAUL SÉRGIO BARTH

Design and Implementation of a Flight Recommendation Engine

Graduation Project.

Prof. Dr. Fernanda Lima Kastensmidt

Advisor

Porto Alegre, June, 2014

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitora de Graduação: Prof. Sérgio Roberto Kieling Franco

Diretor Instituto de Informática: Prof. Luís da Cunha Lamb

Coordenador ECP: Prof. Marcelo Götz

Bibliotecário-Chefe do Instituto de Informática: Alexander Borges Ribeiro

ACKNOWLEDGEMENTS

I would like to thank all the people that made this project excellent professional and personal experience.

First of all I would like to thank my mother, Leda Marisa Barth, who always gave me all the necessary support I needed to grow up and be where I am. Thank you for being a dedicated and exceptional woman and mother. Also I would like to thank my godfather, Sérgio Ivo Barth, for being like a father to me, my godmother Ione and for my second mother Jane Cunha, and my girlfriend Mira Cunha Machado, for always be with me, believing in me and supporting me throughout my entire academic and personal life.

Generally, I also would like to thank all the Amadeus AS2 team for the good moments and eventual support. Finally, I would like to thank my advisor, Fernanda Lima Kastensmidt, for the support and guidance during this project.

Table of Contents

ACKNOWLEDGEMENTS	3
1. INTRODUCTION	9
1.1. Company Introduction	10
1.2. Project Organization and My Contribution	10
1.3. Organization of this work	11
2. RELATED WORK	12
2.1. Collaborative Filtering	12
2.2. Content-based Filtering	13
2.3. Hybrid Recommender Systems	13
3. CONTEXT OF THE PROJECT	16
3.1. Motivation	16
3.2. Problem Statement	16
3.3. System Overview	17
3.4. Functional Example of the System	18
4. IMPLEMENTATION OF THE PROTOTYPE	21
4.1. Architecture	21
4.2. Offline Mode	22
4.2.1. Overview of the Offline Mode	22
4.2.2. Data Characteristics	22
4.2.2.1. Sources	22
4.2.2.2. Extraction	23
4.2.2.2.1. Features: the relevant information	23
4.2.2.2.2. Features Types	24
4.2.2.3. Data Normalization	27
4.2.2.4. Storage	28
4.2.3. Wikipedia Distance: destination characteristics	29
4.2.3.1. Cosine Distance	30
4.3. Online Mode	32
4.3.1. Overview of the Online Mode	32

4.3.2.	Flights: representing them as vectors.....	33
4.3.3.	Similarity Computation	33
4.3.3.1.	Weighted Euclidean Distance.....	34
4.4.	Recommendations	36
5.	RESULT ANALYSIS.....	37
5.2.	Drawbacks	41
5.3.	Improvements.....	41
5.4.	Amadeus Integration.....	41
6.	CONCLUSION AND FUTURE WORK.....	45
	REFERENCES.....	46
	APPENDIX A: CODE.....	49
	APPENDIX B: PROJECT DESCRIPTION (TG1).....	50

List of Figures

Figure 1 - Flights: vectors in space.....	17
Figure 2 - Environment Overview.....	18
Figure 3 - Functional Example: environment overview.....	19
Figure 4 - Functional Example: flights seen as vectors.....	19
Figure 5 - Functional Example: similarity computation.....	20
Figure 6 - Flight Recommender Engine: General Structure.....	21
Figure 7 - Offline Mode Overview.....	22
Figure 8 - Normalization Equation.....	28
Figure 9 - Normalization Example.....	28
Figure 10 - Database: tables.....	29
Figure 11 - Wikipedia Feature: dictionary of words.....	30
Figure 12 - Wikipedia Feature: building city vector.....	31
Figure 13 - Wikipedia Feature: cosine distance.....	31
Figure 14 - Online Mode Overview.....	32
Figure 15 - Building flight vector.....	33
Figure 16 - Cosine Distance.....	34
Figure 17 - Euclidean Distance.....	34
Figure 18 - Weighted Euclidean Distance: configurable features weights.....	35
Figure 19 - Example Similarity Computation.....	35
Figure 20 - List of Similarities.....	36
Figure 21 - Prototype Interface.....	37
Figure 22 - Results: example 1.....	38
Figure 23 - Results: example 2.....	38
Figure 24 - Results: example 3.....	389
Figure 25 - Results: example 4.....	389
Figure 26 - Integrated Result: search page.....	42
Figure 27 - Integrated Result: recommended flights.....	43
Figure 28 - Integrated Result: trip map and destination map.....	44

List of Tables

Table 1 - Recommendations Systems Comparative Table.....	14
Table 2- Origin-Destination Feature.....	25
Table 3 - Origin-Destination-Period Feature.....	25
Table 4 - Origin-Destination-Period Feature.....	26
Table 5 - Destination Feature	26
Table 6 - Destination-Period Feature	27
Table 7 - Destination-Period Feature	27

RESUMO

O uso de sistemas de recomendação tem provado aumentar as vendas na maioria das plataformas de e-commerce. Esse tipo de sistema busca prover ao usuário mais opções, mais itens que possam ser de seu interesse, dando a ele maior conforto e praticidade, e também provendo à companhia maior chance de aumentar a venda de seus produtos e serviços. Esse projeto é inspirado em Amazon Recommendations Systems, mas com uma abordagem diferente.

Esse projeto foi desenvolvido na Amadeus Company, uma provedora líder de soluções de TI criada pela Air France, Iberia, Lufthansa, e SAS como um Sistema Global de Distribuição em 1987 e tem buscado cobrir a indústria de turismo e viagem. O sujeito desse projeto consiste na criação de um protótipo que providencia destinos de viagens alternativos para os clientes da Amadeus.

Na primeira parte desse relatório é apresentado o contexto desse projeto, e então são apresentadas a motivação e o ambiente de desenvolvimento, bem como a análise que foi feita para solucionar o problema e a modelagem do sistema, chamado Flight Recommender Engine. Ao final, é explicado a integração do protótipo no sistema da Amadeus.

Alguns detalhes específicos sobre a implementação e avaliação de resultados do projeto não podem ser abordados nesse documento, visto que são de cunho confidencial em contrato assinado com a empresa Amadeus Sas, Sophia-Antipolis, França.

Palavras-Chave: similaridade, distância euclidiana, ferramenta de recomendação, features

ABSTRACT

The use of recommendation systems has proven to increase sales on most e-commerce platforms. This type of system aims to provide to the user more options, more items that he could be interested in, giving to him more comfort and practicality, and also providing to the company a greater chance to increase its products and services sells. This project is inspired in Amazon Recommendations Systems, but with a different approach.

This project was developed in Amadeus Company, a leading IT solutions provider created by Air France, Iberia, Lufthansa, and SAS as a Global Distribution System in 1987 and has evolved to cover the whole travel and tourism industry. The subject of this project consists in creating a prototype of an intelligent application which provides alternative destinations to Amadeus clients, based in some business intelligence information, as well as some unstructured information from web, aiming to recommend the best alternative destinations to the user.

In the first part of this report, the context of this project is presented, and then the motivation and the environment, as well as the analysis that was made to solve the problem and the modeling of the design and the Flight Recommendation Engine itself in a theoretical and technical analysis. At the end, is explained the integration of this prototype in the Amadeus System.

Some specific details about the implementation and evaluation of project results cannot be approached in this document, because they are part of a confidential agreement with the company Amadeus SAS, Sophia-Antipolis, France.

Keywords: similarity, Euclidean distance, recommendation engine, features

1. INTRODUCTION

In a constantly evolving and competitive environment, selling platforms, services, and travel merchants in particular, companies are faced to the challenge of offering to their customer relevant content and options. On a longer term, customers who find the offered content relevant are more likely to use the service again. The recommendation services are very important nowadays, as for the companies who want to sell the more products and services, keeping the clients satisfied and aggregating new ones, as for the users who want the most comfort and quickly service as possible.

This project was created based on that, which the subject consists in providing flights with alternative destinations and periods to the clients. More specifically, the subject consists in creating a prototype of an intelligent application that uses the similarity computation approach, taking into account some business intelligence information, as well as some unstructured information from web, aiming to recommend the best alternative destinations to the client. This project was developed as the subject of an internship in the Amadeus Company, a leading IT solutions provider. This company was initially created by Air France, Iberia, Lufthansa, and SAS as a Global Distribution System in 1987 and has evolved to cover the whole travel and tourism industry. Its first goal was to enable these airlines to have a common structure of distribution of airs segments and group together the whole offer of sale of plane ticket in order to better serve the travel distribution needs.

There are similar approaches in the recommendations systems field, however, there are not so much ones in the airlines area. Also, different of others recommendations engines, the recommendation engine developed in this project does not take into account - due to the fact that the environment where it must work does not support - the user profile and users rates, being the result modeled according to what the company wants to show to its clients. This is the first recommendation prototype in Amadeus that takes into account so many different features to calculate and analyze the similarity between the products that, in this scenario, are the flights. The items used by the recommendation engine are very deeply described and the prototype uses flights information that were mined from the company logs and databases, as number of passengers, flights prices and destination characteristics, providing good alternative destination to the users. All these characteristics make the result easily moldable by Amadeus or by TAM, according to what they want to provide as options to the client, what is considered the greatest advantage of this prototype.

Also, the prototype uses the notion of recommendations algorithm, having a huge mathematical part, making use of Euclidean Distance and Cosine similarity applied to an industrial problem. The development of this engine required a huge data extraction and analysis, as well as a considerable mathematical approach.

Some specific details about the implementation and evaluation of project results cannot be approached in this document, they are part of a confidential agreement with the company Amadeus SAS Sophia-Antipolis, France.

1.1. Company Introduction

In 1990's, Amadeus established a strong presence in the world through the opening of regional and national offices and introducing itself into the online travel market as well. Nowadays, its solutions are used by airlines, railway companies, travel agencies, hotel chains, airports, and car rental companies. The company acts both as a worldwide network connecting travel providers and travel agencies through a highly effective processing platform for the distribution of travel products and services (through our Distribution business), and as a provider of a comprehensive portfolio of IT solutions which automate certain mission-critical business processes, such as reservations, inventory management and operations for travel providers (through our IT solutions business).

Amadeus is located in several countries, having commercial representations that cover more than 190 countries. The most important are Corporate Headquarters and Marketing in Madrid, Development in Sophia Antipolis, and Operations in Erding (Germany). This project is being developed in the Amadeus Sophia-Antipolis, France, from February, 11, to August, 9.

1.2. Project Organization and My Contribution

All the project was developed from scratch, as an internship subject in the Amadeus Sas company, located in Sophia-Antipolis, France, being part of a Double Degree program between Grenoble Institute of Technology - INP - and the Federal University of Rio Grande do Sul - UFRGS. There were no other recommendation engines similar to the one developed in Amadeus, so all the project steps were done during the internship period, since the specification documentation, approaching the problem and modeling the solution, passing through the software modeling, the implementation itself and the integration in the TAM Airlines test page. It was developed in 6 months, being the most part of this time dedicated mainly to the solution modeling and to the implementation and coding.

The project was developed using the Agile Methodology, in the Scrum model, with a huge communication between the author of this report and his company manager, having weekly meetings to analyze the deadlines that were stipulated, as well as to solve current problems and doubts during the development. To build this prototype, were put into practice strong knowledge of software engineering, fault tolerance techniques, programming standards and Structured Query Language (SQL).

Since the beginning some deadlines were stipulated, due to the fact that at the end of 6 months of working, a final product should be delivered. As that, the project was divided in five big steps: specification, solution modeling, implementation, testing, and integration. In the first step was done the specification of the project, in other words, the problem was analyzed and some solutions were provided. In the second step, the solution of the problem was chosen and modeled, and a new documentation was created with this subject. In the implementation step consists of the code implementation, as well as the technology that would be used.

The project was developed using Java programming language and some scripts were done in Python responsible for the data mining in the Amadeus logs and databases. Apache Cassandra Database was used as database. Most part of the company uses Oracle, but after an analysis of the possible databases to be used, Cassandra was chosen. This is an open source distributed database management system and its goal is a well handling of large amount of data across many commodity servers, with a high availability and with no single point of failure. As the prototype, in the future, must work with a high amount of data in all over the world, this database offers robust support for clusters spanning multiple datacenters and with a high level of performance. The tests were done gradually, as the prototype was developed using Agile Methodology. They were done in each step of the project, using Unitary Tests, Integration Tests, Operational Tests, Functional Tests, Performance Tests and Interface Tests. In the last step of the project, the prototype was integrated in the TAM Airlines test page and some final tests were performed. To that be possible, was used some code in JavaScript, PHP and CSS, integrating the prototype with TAM.

In a general saying, a huge mathematical approach was used, including some similarity computation mechanisms as Euclidean Distance and Cosine Distance.

1.3. Organization of this work

In this document is presented the Flight Recommender Engine, developed in Amadeus Sas, a software development company of the airline and tourism areas. In the first part of this report, the context of this project is presented, what were the reasons to it be developed, as well as the motivation and the environment. Later, is presented the solution analysis made to solve the problem and the modeling of the design. At the end, is presented the Flight Recommender Engine itself in a theoretical and technical analysis, and the integration of the prototype in the TAM Airlines test page.

2. RELATED WORK

There are several approaches related to recommendation engines, the collaborative filtering, content-based filtering and hybrid recommender systems, for example. Each one of them is composed by different sub-types of algorithms, which will not be approached here. After the explanation of each one of these three main types, a table is shown comparing them and the algorithm used by the prototype developed in this project.

2.1. Collaborative Filtering

The collaborative filtering has its methods based in users behaviors preferences or activities and predicting what users will like most, based on their similarity to other users. One of the most famous examples is the Amazon Recommender System, which uses the item-to-item collaborative filtering. This method is used to make automatic predictions - filtering - about the interests of the users, collecting information and preferences of several users. This collaborative filtering is done supposing that if a certain person A has the same opinion as a person B on an issue, then, on a different issue, A is more likely to have B's opinion. It means that this type of algorithm requires having user profile.

This recommendation method requires, usually, the user participation - i.e. the profile analysis - and algorithms capable of match people with similar interests. Basically, in this method, the user expresses his opinion by rating the items in the system, then the systems matches the user's ratings against other users and finds users with similar interests. With these similar users. In other words, the system looks for users that share the same rating patterns with the actual user and then it uses the ratings from those like-minded users that were found to calculate a prediction for the actual user.

One of the most known collaborative filtering is the one used in Amazon, called item-based collaborative filtering. This recommendation system uses the idea that who bought a product x also bought a product, and it proceeds in an item-centric manner. As explained in a paper made from Amazon developers, called *Amazon.com Recommendations - Item to Item Collaborative Filtering [bibliography]*, this type of recommendation algorithm is computationally expensive. It is $O(MN)$ in its worst case, being M the number of costumers and up to N items for each one of them. As the average costumer vector is sparse, the performance tends to be closer to $O(M+N)$, what represents the final performance of the algorithm. As the number of costumers and items increase, the algorithm encounters severe performance and scaling issues. The use of clusters can increase the performance and reduce these issues.

There are three types of collaborative filtering, the memory-based, model-based and hybrid. These types will not be approached in this report. As mentioned before, this recommendation algorithm is used by *Amazon* combined with clusters usage. It is one of the most used recommendation algorithm.

2.2. Content-based Filtering

This recommendation method is based on the item description and a profile of the preferences of the user. The items are described by keywords and a user profile is built, indicating the type of items that this user likes. Basically, the algorithm of this type of method try to recommend items that are similar to those that a user liked in the past, i.e., the algorithm compares items previously rated by the user with candidate items, and the best matches are recommended. As the Collaborative-Filtering, this method also algorithm requires having user profile. To make the abstraction of the items features in the system, an item presentation algorithm is used. In this case is used the tf-idf representation, that represents the items as vectors in the space. This method is also used in this project, meaning “Frequency-inverse document frequency”, and is a numerical statistic responsible for reflecting how important a word is to a document in a collection.

The system uses a model of the user preference and a history of the user interaction to create a user profile. The system also creates a content-based profile of users based on a weighted vector of item features. These weights are important because they denote how important each feature is to the user and can be computed from individually rated content vectors using several techniques. These weights can be decreased or increased, based in the user opinion, as a like or dislike button, for example. In this type of recommendation algorithm, if the user has only few purchases, it scales and performs well, however, when the number of purchases increase, it is impractical to base a query on all the items

2.3. Hybrid Recommender Systems

The hybrid recommender system is a method that combines the collaborative-filtering and the content-based filtering, and can be more effective in some cases. This method can be implemented in different ways, using the other two methods. The first way to do that is adding content-base capabilities to a collaborative-based approach. Also, it can be done by making content-based and collaborative-based predictions individually and then combining them, or by unifying the approaches into one model. There are several studies that proving that this method can provide more accurate recommendations than the other two pure methods. A famous example of system that uses this method is Netflix, that make recommendations by offering movies that share similarities with films that a user has rated highly - content-based filtering - as well as by comparing the watching and searching habits of similar users - collaborative filtering.

The term *hybrid recommender system* describes recommender system which combines multiple recommendation methods to produce the recommendations.

Below, it is shown a table comparing these three recommendations approaches mentioned, as well as the one used in this project. It is important to say that the user profile

was not used in the project because the environment where it should work does not use it as well

	Flight Recommender Engine	Collaborative Filtering	Content-based Filtering	Hybrid Recommender System
User Profile		X	X	X
User Rating		X	X	X
Users Connection (by products rating)		X	X	X
Item Understanding	X			
Deep Description of Items	X			
Key words describing Items	X		X	X
Items Profile (features)	X			
Weighted Features	X		X	X
Resizable Similarity (number of features)	X			
Result Modeling	X			
Tf-Idf representation	X		X	X
Cosine Similarity	X	X		X
Euclidean Distance	X			
Computationally Expensive		X		
Good Scalability	X	With clusters	With clusters	With clusters

Table 1 - Recommendations Systems Comparative Table

Based in the *Table 1*, we can analyze the differences between all the approaches of dation systems. The approach used to develop this project focus in the item profile, where the items, or more specifically the flights, are very well described by some key words and features. Each one of the flights are described by their characteristics, as the number of passengers, destination characteristic, distance between origin and destination and flight

time. As the environment where the prototype was developed to work does not use user profile, the prototype does not use either.

Even though Collaborative Filtering appears as a good alternative to a recommendation system, being used even more nowadays, the main obstacle between this approach and the environment where the prototype works is that the environment does not support the user profile analysis and rating.

3. CONTEXT OF THE PROJECT

3.1. Motivation

The technological progress and the constant improvement of the online technologies are giving the opportunity to the companies to increase their markets and sells, as well as giving to the users even more facilities, options and comfort. The recommendation services are very important nowadays, as for the companies who want to sell the most products as possible, keeping the clients satisfied and aggregating new ones, as for the users who want the most comfort and quickly service as possible, not forgetting the quality.

For that reason the Flight Recommender Engine was developed, to provide to airline companies users more flights options that serve their demands. In order to obtain a full integration experience, and make the prototype as a final product, at the end, the Flight related with the process of booking, searching and displaying of flights.

3.2. Problem Statement

The challenge of this project is the development of a prototype, which aims at recommending the best destinations to the user. The meaning of best destinations, in the approach of this internship, is alternative destinations and periods. The Flight Recommender Engine does that using some business intelligence information as geolocation, flight distance, flight time, itinerary popularity and number of passengers that used to take this flight, and also unstructured information from web, as Wikipedia. In total, there are 26 different types of information, describing each one of the flights in the database and the user flight.

It is clear that, usually, a user that searches for a flight is decided where to go, but sometimes it is not true. Some Amadeus booking logs had been analyzed and was verified that, commonly, there are flights that are booked but not purchased. This analysis shows that the user is not always sure about his choice and sometimes he changes his opinion. And that is what the Flight Recommender Engine is intended to cover. Based on the fact that sometimes the user is not sure about where to go, the prototype developed in this project aims to provide to this user good alternative destinations and periods, making him chooses a flight, decreasing the case of not finalized bookings, increasing the sales.

The challenge, then, is to find the most similar flights to the user flight. The approach used is to see all these flights as vectors in the space, and then, using the Weighted Euclidean Distance, calculate the similarity between them. As there are 26 different information types, then, there are 26 dimensions in the space where the flights are. The figure below illustrates how the prototype sees all the flights in the space.

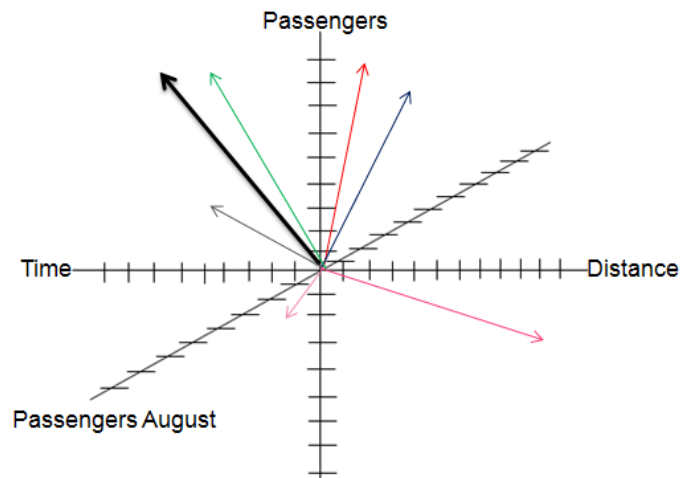


Figure 1 - Flights: vectors in space

As there is one information type that is the flight distance, we will have one dimension in the space that represents this information, and the same thing for all the information types. Therefore, as there are 26 different information types characterizing each one of the flights, the space where the flights are represented have 26 dimensions.

3.3. System Overview

The prototype has as input the origin, the destination and the period of the user flight. Having this data, the prototype accesses the database and gets all the data related to this specifically flight. As was mentioned before, all the flights in this scope are seen as vectors in an N-dimension space, so then, the problem is basically to find the nearest neighbors of the user flight. To be able to follow this approach, the prototype builds a vector, which represents the user request, composed by the data retrieved in the database. The same thing, then, is done for all the flights in the database. At the end, the Flight Recommender Engine calculates the similarity between the user flight and all the database flights, finding the most similar, and recommending them to the user.

Therefore, the prototype's output is the K-best flights, where K is the number of recommended flights shown to the user, composed by the flight origin, flight destination and the flight month. The recommended flights are combined with the user flight booking and the, shown to the user. The process, in a general illustration, is shown in the figure below.

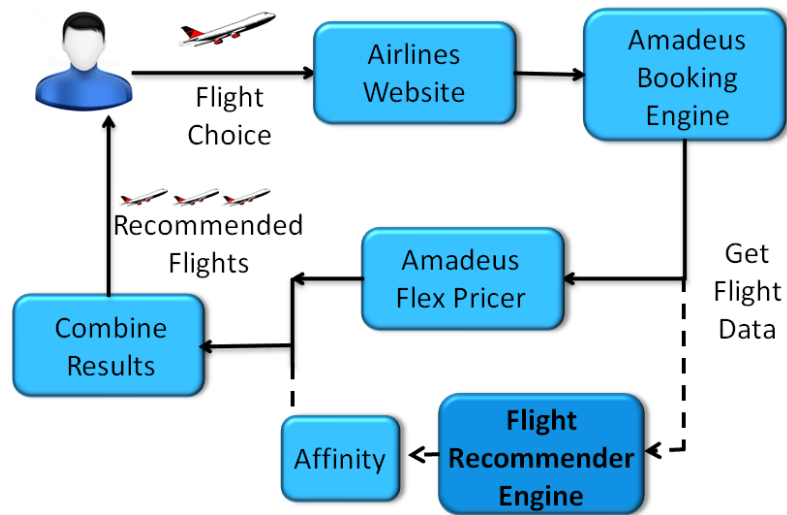


Figure 2 - Environment Overview

In the airlines company website, the user searches for a flight. The result of this choice passes through the Amadeus Booking Engine, responsible for pre-booking this flight with its respective characteristics, such as origin, destination, date and number of passengers, in other words, it is responsible for all booking-dependent functionalities. After that, the result is sent to another tool, called Flex Pricer, which gives to the user the cheapest price for a period of time and the price of different class in a date. The Flight Recommender Engine works in the same level as the Flex Pricer. It means that the prototype receives the user flight data given by the Booking Engine and processes that, obtaining the best recommendations. After that, the result is sent to Affinity Shopper, which is used to get the real prices of the recommended flights. Then, the result of the Flex Pricer are combined with the results of the Flight Recommender Engine, and shown to the user.

3.4. Functional Example of the System

The input is the flight chosen by the user, also called User Query. This Query is the flight chosen by the user, for example, a flight from Amsterdam (AMS) to Madrid (MAD) departing in August 9 and returning in August 21. That is the input of the Flight Recommender Engine. Having the input, the prototype accesses the database, aiming to get all the data related to this flight. The next figure illustrates the general environment.

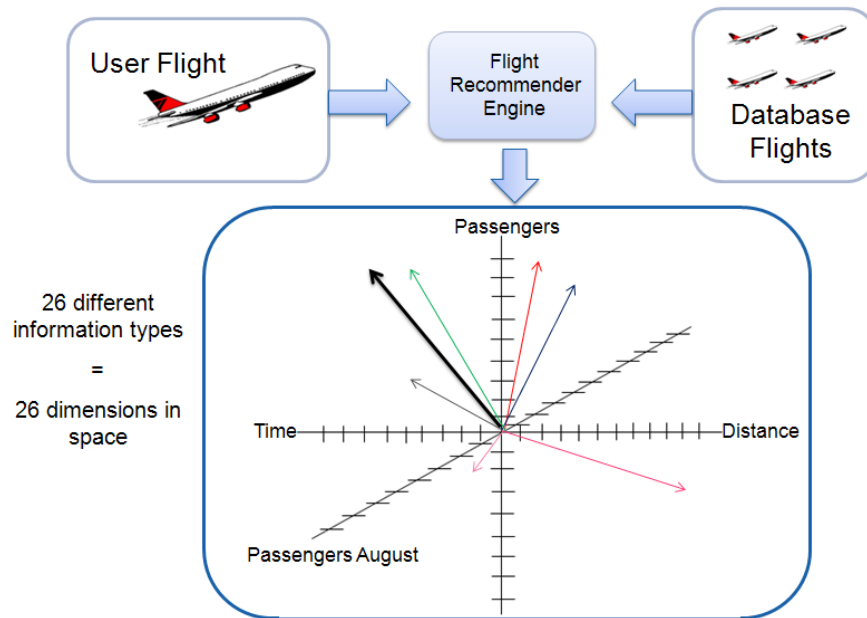


Figure 3 - Functional Example: environment overview

In one hand we have the user flight, that is the input of the prototype, and in another hand we have all the flights in the database. The prototype sees all these flights as vectors in the space. In the *Figure 4* there are the user flight and the database flights, all of them represented as vectors in the space, based in the information of each one of the flights, as distance, passengers, passengers August and flight time.

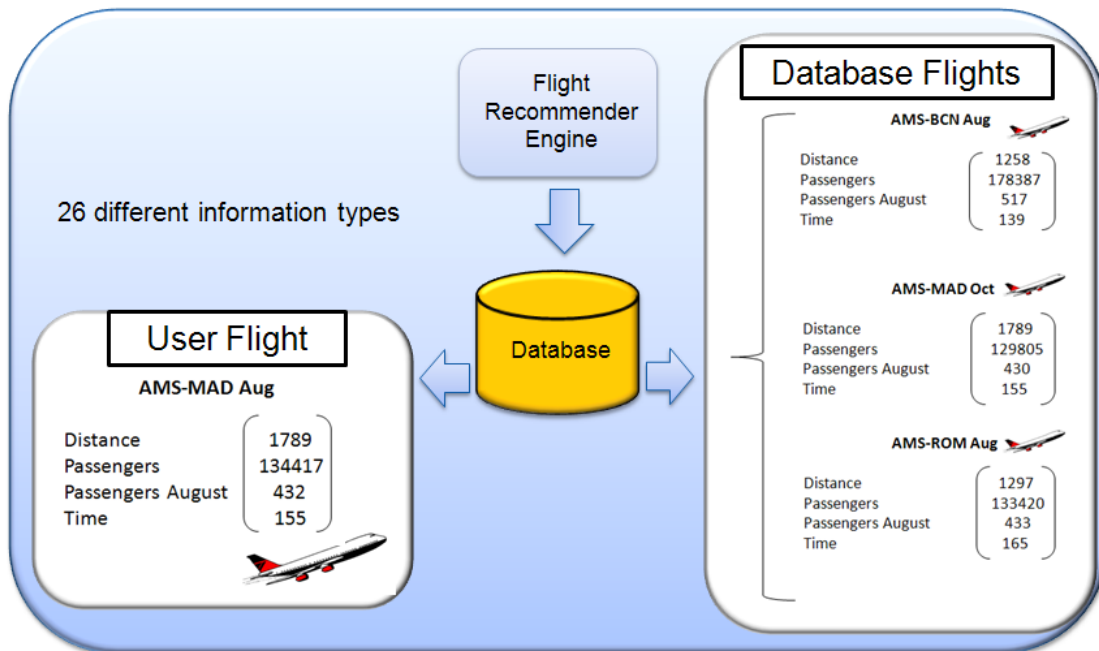


Figure 4 - Functional Example: flights seen as vectors

From now on, the challenge consists in calculate the similarity between the user flight and all the database flights, aiming to find the nearest neighbors. That is done using the Weighted Euclidean Distance.

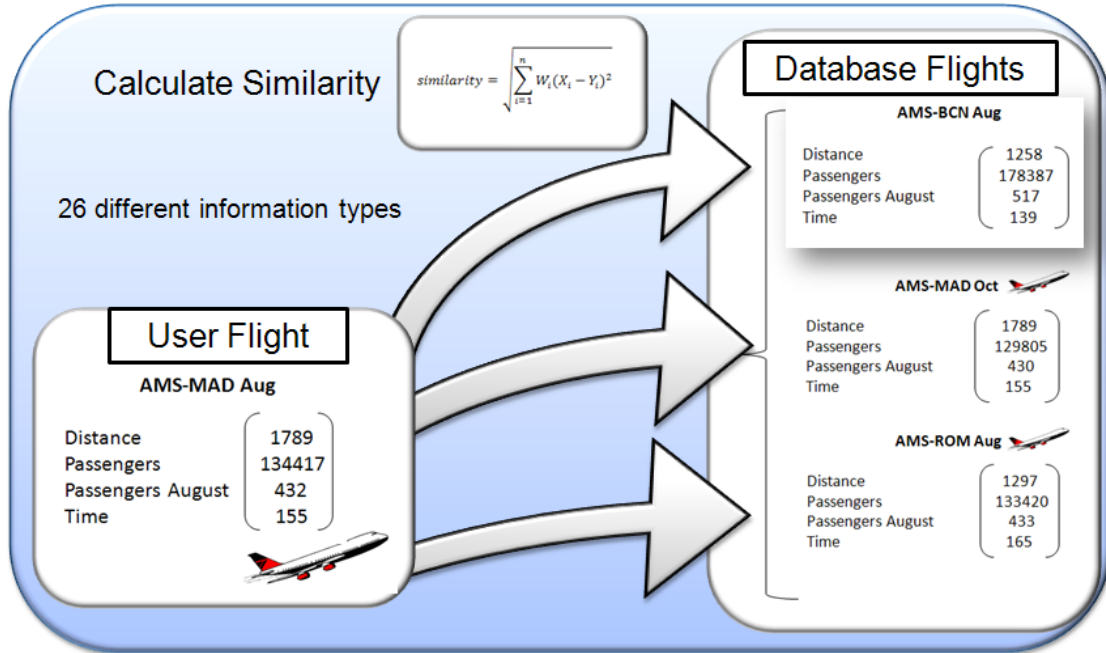


Figure 5 - Functional Example: similarity computation

4. IMPLEMENTATION OF THE PROTOTYPE

4.1. Architecture

The *Figure 7* represents all the process done by the prototype, since the recovery of the user input request, passing through the similarity computation, and ending in the recommended flights. The prototype can be divided into two modes, the offline mode and the online mode. The offline mode is responsible for all the pre-process, as the data extraction, data normalization and data storage in database. The online mode is related to the online process, responsible for getting the user flight, transforming it and all the database flights in vectors represented by their respective features, and responsible for the similarity computation between all of the flights and the user flight, managing the results and recommending the best destinations and periods.

The general structure of the prototype is shown below:

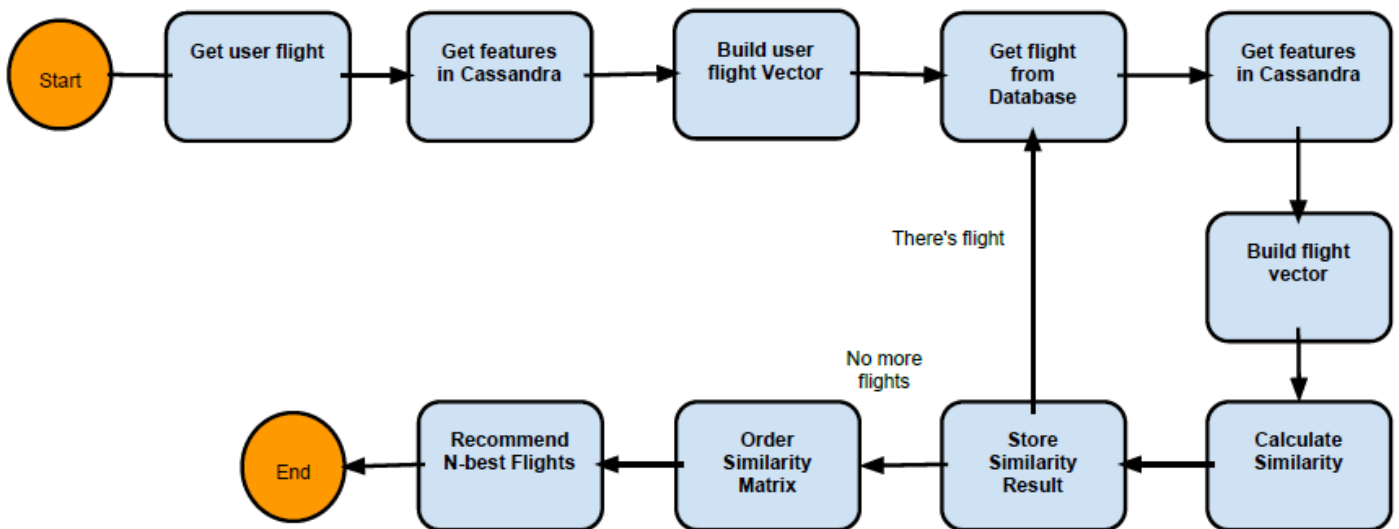


Figure 6 - Flight Recommender Engine: General Structure

This figure represents a general overview of how the prototype works. First of all, it gets the user flights, the input of the prototype. Then, the prototype gets in the database the respective features of this flight and builds a vector with them, representing the flight in the space. After that, for each one of the flights in database, the prototype gets their respective features, builds their vectors, and calculates the similarity between each one of them and the user flight. This process is done till there is no more flights in database to be analyzed. After that, with all the similarities computed, they are ordered and the most similar flights are recommended to the user.

4.2. Offline Mode

4.2.1. Overview of the Offline Mode

The offline mode is the pre-processing mode of the prototype. It approaches all the tasks related to the data needed by the prototype, more specifically, the data of all flights. The first part of the offline mode is the data extraction, followed by the data normalizing and then, by the data storage. Following, a general figure which show the structure of this mode.

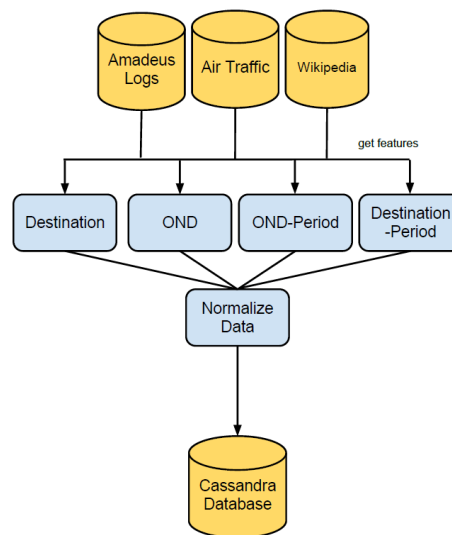


Figure 7 - Offline Mode Overview

4.2.2. Data Characteristics

The first part of the offline mode is referred about the data that was used. In this section will be explained all the steps related to the used data, as the sources from where it was obtained, how the extraction was done and how the data was managed and stored.

4.2.2.1. Sources

The data needed by the prototype is all the data types that characterize the flights, as the departure time, arrival time, distance between origin and destination, and so many others. These data were obtained from three different sources, shown below:

- Amadeus Logs:

Around 400Gb of data was analyzed. From that source, two different information types were obtained. The first one is the number of requests that were made for a certain

trip (origin-destination), the second one is the average stay duration, which is the total stay duration (number of days between departure and return) divided by the number of requests for a certain origin-destination. All the data from Amadeus Logs was needed to be parsed aiming to extract just the relevant information.

- Air Traffic Amadeus Tool

The Air Traffic is an Amadeus tool. This tool was used to search flights data, considering itineraries which have as origin a European city or as destination an European city (World-Europe and Europe-World). From these itineraries, were obtained data as the number of airlines that support each route, the flights time, distance between cities, and so many others. That is the most important data source used in the project. All the information from Air Traffic Amadeus was obtained in CSV files and was needed to be parsed aiming to extract just the relevant information for the prototype.

- Internet Data

The internet data was used to obtain the characteristics of all the cities in the database. This data were obtained from Wikipedia. Explaining breathily – it will be deeply explained later – the prototype gets all the texts in Wikipedia related to each one of the cities in the database. Then, two cities can be compared taking into account their Wikipedia texts. Using this source, we can recommend destination which have the same characteristics as the user destination.

4.2.2.2. Extraction

This step of the offline mode consists in figure out what is the relevant information for the prototype. First of all will be presented the relevant information that were extracted, and then they will be characterized.

4.2.2.2.1. Features: the relevant information

As mentioned before, the data needed to the prototype is the information about the flights. Nine different information types were taken into account, each one of them is considered as one feature of the flight, representing one dimension in the space. The types are listed below, and after they will be detailed, resulting in a total of 26 different information types.

- Airlines: number of airlines that take a certain route (origin-destination)
- Distance: distance between origin-destination
- Time: flight time, time spent by the airplane to go from the origin to destination.
- Frequency: number of flights that take a certain route (origin-destination)
- Passengers: number of passengers that take certain route (origin-destination)

- Itineraries: number of different itineraries that the flight can take to go from the origin to the destination (different scales and airports)
- Requests: number of times that the route was requested in the Amadeus site
- Average Stay: number of days between Departure-Return divided by the number of requests for a certain route

The features above are called as general features, and they are divided into four types:

- O&D (Origin-Destination): the features of this type are related to the whole trip, in other words, to the origin and the destination of the flight. As example, can be mentioned the distance between the origin and the destination and the time spent by the flight to go from that origin to that destination. Is important to say that the values of these features are related to the whole year.
- O&D-Period (Origin-Destination-Period): this type is almost the same as the first one, but it has a relation to the flight period. While the O&D type values are related to the whole year, the values of O&D-Period are related to each month of the year. As an example can be mentioned the number of passengers that take a certain route in August and the number of passengers that take the same route in January.
- Destination: the features of this type are related just to the destination of the flight. As examples can be mentioned the number of passengers that goes to this destination by year and the number of times that this destination was searched in the Amadeus site.
- Destination-Period: this type is similar to the last one, except to the fact the period, in this case, is each month of the year, as the O&D-Period type. AS example, there are the number of passengers that goes to this destination in September.

4.2.2.2.2. Features Types

The relevant extracted data, now called features, are divided into four types. In this section, all of them will be approached. The first is the O&D type. This type represents the features of Origin-Destination, and takes into account the whole year as period. The features presented in this type and an example with four trips is listed below:

- Airlines total: number of airlines that take a certain route.
- Distance (Km): distance between the origin and destination.
- Time total (Min): total time to go from origin to destination.
- Frequency total: frequency of flights that take a certain route.
- Passengers total: number of passengers that take a certain route.
- Itineraries total: number of different itineraries to go from the origin to destination.
- Requests total: number of request of a certain route.

- Average Stay total (Days): average stay duration - days between departure and return.

Table 2- Origin-Destination Feature

Origin	Destination	Airlines	Distance	Time	Frequency	Passengers	Itineraries	Requests	Av. Stay
Paris	Nice	121	1160	84	8949	910568	62	1345	6
London	Paris	142	316	65	8139	339876	71	1329	5
Madrid	Rome	54	1334	142	1687	133976	28	847	6
Rome	Berlin	237	1168	195	3798	133175	112	984	4

The second type to be approached is the O&D-Period. This type represents the features of Origin-Destination by period, in other words, by month. It takes into account the months of the year as period and their meaning are the same as the features presented before. The features presented in this type and an example with four trips in the months of November and January are listed below:

- Airlines January
- Time January
- Frequency January
- Passengers January
- Itineraries January

Table 3 - Origin-Destination-Period Feature

Origin	Destination	Airlines November	Time November	Frequency November	Passengers November	Itineraries November
Paris	Nice	16	82	989	105358	9
London	Paris	22	63	1432	54943	10
Madrid	Rome	6	143	673	13598	4
Rome	Berlin	25	204	1213	12465	14

Table 4 - Origin-Destination-Period Feature

Origin	Destination	Airlines January	Time January	Frequency January	Passengers January	Itineraries January
Paris	Nice	15	81	1720	186952	8
London	Paris	45	65	2689	119756	22
Madrid	Rome	26	145	872	72034	12
Rome	Berlin	50	206	760	20210	38

Above are represented the features for January and for November. The database will have these different features for each one of the year's months. As an example, we will take into account the "passengers" feature. For the flights from London to Paris, in the month of January, we can have 119820 passengers that take this route, while for the same route, in the month of November, this feature has the value of 54938 passengers.

The third features type is the Destination. This type represents the features just of Destination, and takes into account the whole year as period.

- Airlines total
- Time total
- Frequency total
- Passengers total
- Itineraries total
- Requests total
- Average Stay total

Table 5 - Destination Feature

Destination	Airlines	Time	Frequency	Passengers	Itineraries	Requests	Av. Stay
Nice	37145	210	2178029	45097	21069	6452	3
Paris	95681	243	18925997	1824506	310638	13274	6
Rome	65248	99	5788235	2039450	21689	9284	4
Berlin	54189	140	3314859	920007	47452	38699	4

The last type is the Destination-Period. This type represents the features of Destination by period, in other words, by month. It takes into account the months of the year as period.

- Airlines January
- Time January
- Frequency January
- Passengers January
- Itineraries January

Table 6 - Destination-Period Feature

Destination	Airlines November	Time November	Frequency November	Passengers November	Itineraries November
Nice	2183	254	24967	189648	754
Paris	5248	441	68549	1441005	3100
Rome	3599	725	38850	474507	2027
Berlin	3143	798	31837	259266	1751

Table 7 - Destination-Period Feature

Destination	Airlines January	Time January	Frequency January	Passengers January	Itineraries January
Nice	7634	1074	90238	468056	4374
Paris	1968	424	241036	504967	1112
Rome	1399	650	139238	200910	8122
Berlin	1053	877	104404	893842	5906

4.2.2.3. Data Normalization

From the begging until now was explained about the data, how and from where it was obtained and how it was characterized. Before storage the data, however, is necessary to normalize it. According to the Oxford Dictionary of Statistical Terms (entry for normalization of scores), normalization is done with the objective of:

“... Adjusting values measured on different scales to a notionally common scale, often prior to averaging.” (Oxford Dictionary of Statistical Terms)

That is the case of this project approach, the normalization is done for the flights be able to be considered in the same dimension space. The flights relevant information was extracted, but as can be realized, they are in different metrics. As example, we can mention that the distance between two cities is represented in kilometers and the flight time in minutes. In this project, the normalization is done using the follow equation:

$$X_{i, 0 \text{ to } 1} = \frac{X_i - X_{\text{Min}}}{X_{\text{Max}} - X_{\text{Min}}}$$

Figure 8 - Normalization Equation

This equation represents that for each feature value i , it will be normalized between 0 and 1, where X_{Min} is the minimum value of the range and X_{Max} is the maximum value of the range. To understand how it is used in the project, a simple example is given below:

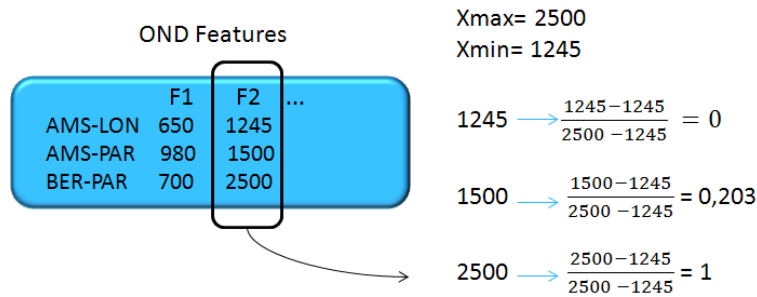


Figure 9 - Normalization Example

4.2.2.4. Storage

After normalize the features, we are able to store them in the database. The database chosen for this project is the Cassandra Database and, how this project was developed in Java, the Hector API is the tool chosen to make the connection and operation with the database.

The Apache Cassandra Database is an open source distributed database. Some goals of this database are that it was designed to handle very large amounts of data and it provides a high scalability and high availability without compromising performance. This project works with huge amounts of data, around 1.300.000 flights. Due to that, it is important to work with a database that knows how to handle with this huge quantity, and also, the fact that more infinite possibilities of news features can be added to the flights, what represents a clear necessity of scalability. There is also Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages. Hector API is a high level Java client for Apache Cassandra currently in use on a number of production systems some of which have node counts into the hundreds. We can mention as Hector's features, the high level, simple object oriented interface to Cassandra, failover behavior on the client side and connection pooling for improved performance and scalability.

Below is shown the structure of the database, composed by six tables.



Figure 10 - Database: tables

There are four tables representing the four types of features, one table with the result of the Wikipedia Similarity for all the cities combinations, and one table called “flights” which was built with all the data from the four types of features tables. It was done to facilitate the searching process.

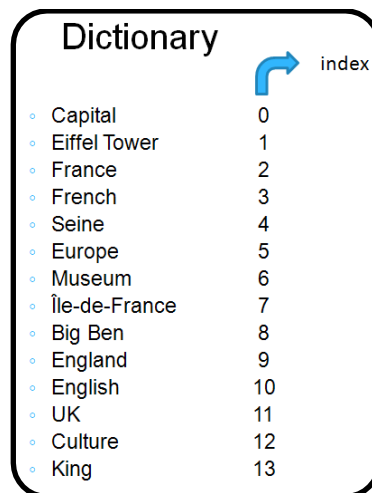
4.2.3. Wikipedia Distance: destination characteristics

The Wikipedia distance is another type of feature that is treated in a different way. All the cities, or most of them, have a page in the Wikipedia where there is the text referent to this city, which is used to describe the city, giving characteristics of history, politic, weather and culture. Realizing that, all the cities in the database can be characterized based

in their Wikipedia texts. The motivation is that, for the destination chosen by the user, we can obtain a real description and characterization of this city. So then, the prototype could search and recommend destinations similar to the user destination, in other words, destination whose Wikipedia text is similar to the user destination Wikipedia text. A XML file with all Wikipedia content can be downloaded. This file, containing 40Gb of information, was parsed, taking off the XML tags, other language words and terms and English stop word, and another file was created just with the cities and their texts used in this project. The technical way of how this comparison and how the similarity of two cities based in their Wikipedia texts are done is explained below.

4.2.3.1. Cosine Distance

In a general explanation, the prototype gets all the words from all the cities and builds a dictionary with the each word and a index. The index is given sequentially, and the used of that will be explained after. Below, there is a sample example with a dictionary containing some words of the cities Paris and London.



Dictionary	
Capital	0
Eiffel Tower	1
France	2
French	3
Seine	4
Europe	5
Museum	6
Île-de-France	7
Big Ben	8
England	9
English	10
UK	11
Culture	12
King	13

Figure 11 - Wikipedia Feature: dictionary of words

Now, the prototype has a dictionary with all the cities words. The next step is the cosine computation itself. In this case, now, the cities are seen as vectors in a dimension space, then, they must be represented by vectors. These vectors are built as following: the prototype gets the city and its text, calculates the frequency of each word in the text, and searches this word in the dictionary, getting its respective index. After that, the frequency of the word is put in the vector in the position represented by its index in the dictionary. An example is shown below for Paris and London:

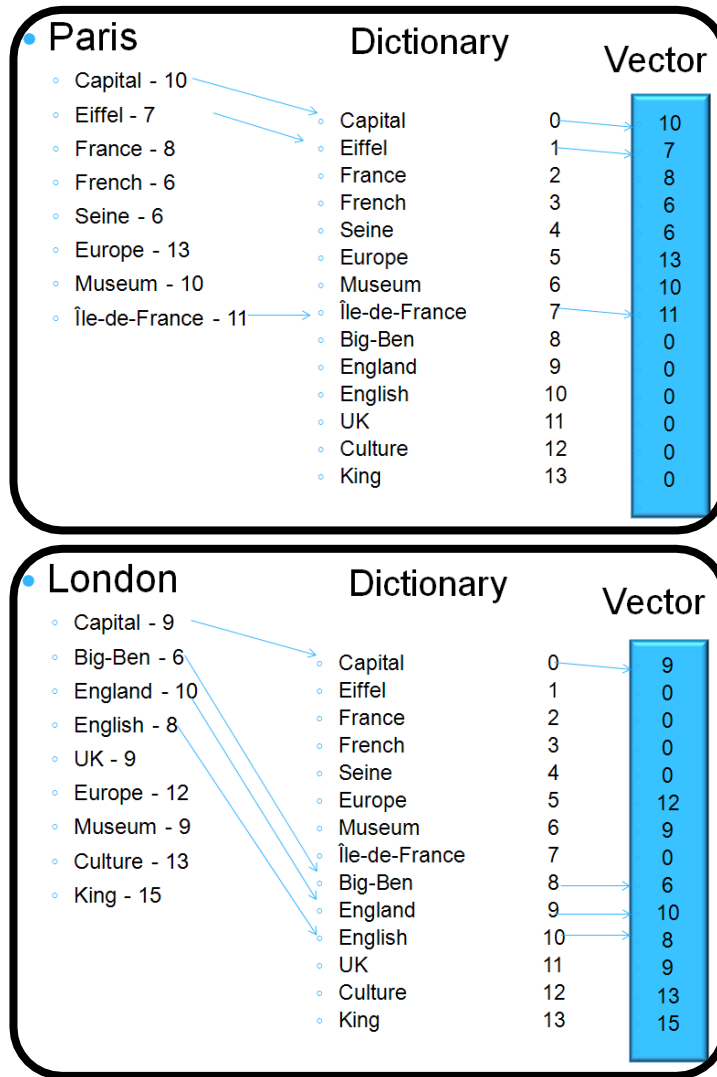


Figure 12 - Wikipedia Feature: building city vector

In this way, it is perceptible that all the cities vectors will have the same size as the dictionary size. For the words that the city does not contain, it is put 0 in the respective vector position. Having the cities represented as vectors, the similarity is calculated using the Cosine approach, represented by the equation:

$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

Figure 13 - Wikipedia Feature: cosine distance

It is important to mention that the Wikipedia Similarity result is another feature in the database. In other words, it represents just one flight feature, related to the destination. The challenge then was to stipulate the importance of this feature in the result, related to the other features. That is because the Wikipedia feature is one of the most important feature, but it is just one feature, against another 25 features taken into account. The weight of Wikipedia Similarity was analyzed and stipulated very carefully, aiming to have the best results.

4.3. Online Mode

4.3.1. Overview of the Online Mode

This mode depends on the data provided by the user, it means the flight chosen by the user. While the offline mode is related basically to the data – extraction, normalization and storage – the online mode is related to the recommendation process itself. The next sections will show how the flights are transformed into vectors, how the similarity between them is calculated and how the recommendations are done to the user. Following, a general figure which show the structure of this mode.

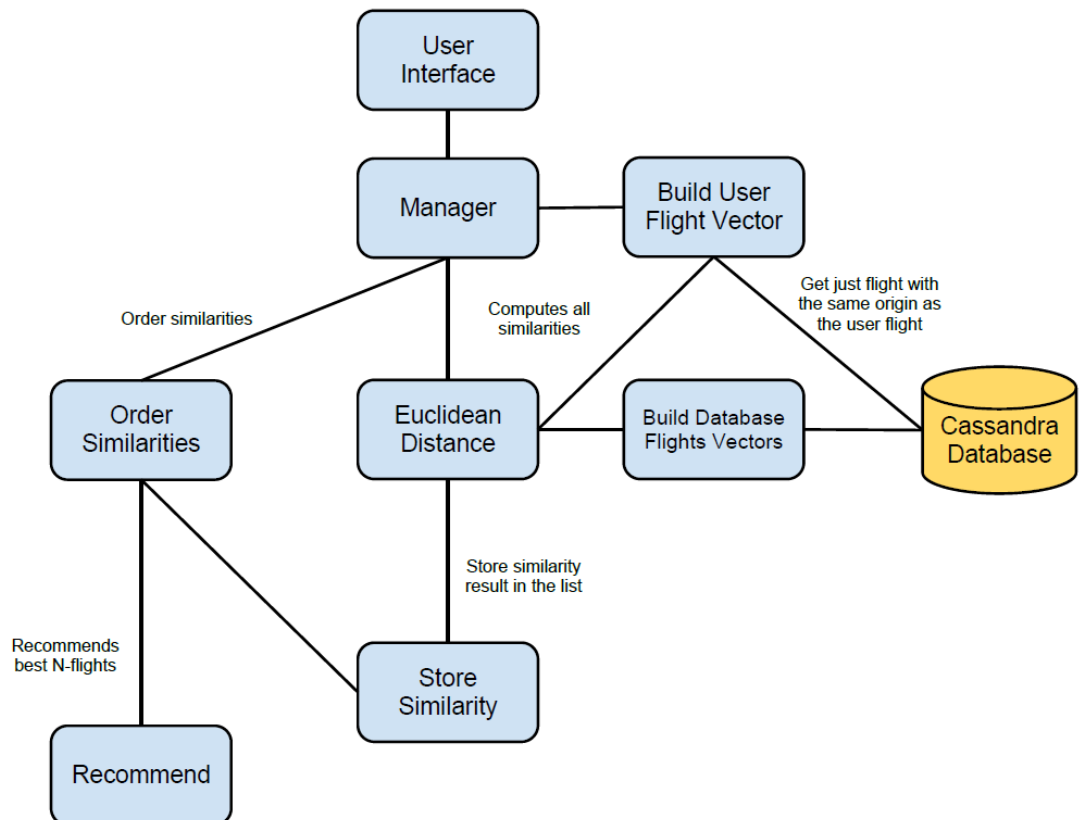


Figure 14 - Online Mode Overview

4.3.2. Flights: representing them as vectors

The reason for transforming the flight in vectors is that they must be seen as in an N-dimension space, so then, the prototype is able to calculate the most similar, in other words, the nearest neighbors of the user flight. For each one of the flight, the prototype get the information of the flight, and then, get the respective features in the database. A vector representing this flight is build, composed of all the features related to this flight. An example is shown below:

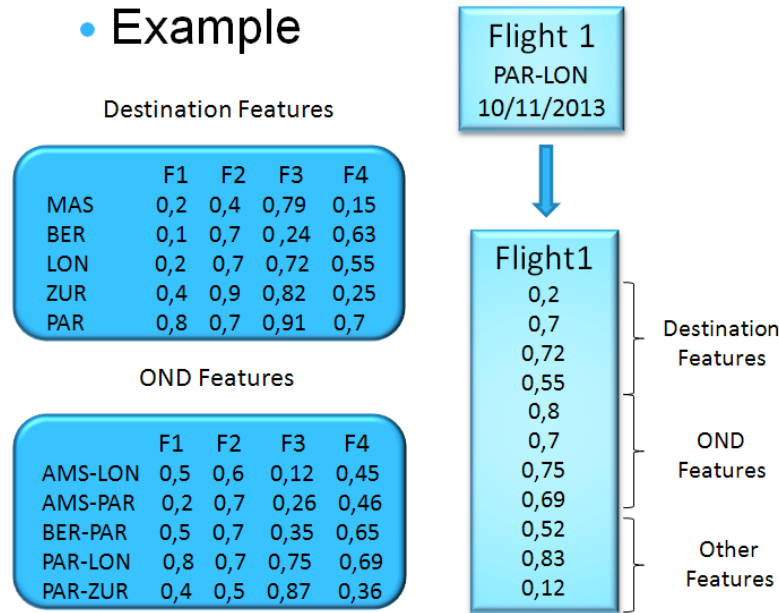


Figure 15 - Building flight vector

This process is done for all the flights in database and the user flight. After finishing the building of one vector, the prototype calculates the similarity of this vector (flight) with the user vector. The result of this calculation is stored in a list, always keeping the flight id.

4.3.3. Similarity Computation

In this section will be explained the similarity computation that is done by the prototype, as well as how it is done and what are the results. The challenge consists in imaging the flights as in a dimension space, each flight represented by its respective vector, composed by the flight features. Similarity computation aims to find the nearest neighbors of the user flight, in other words, the flights that are most similar to the user flight. This is the base for the recommendations indeed.

4.3.3.1. Weighted Euclidean Distance

Two approaches were analyzed to compute the similarity, the first of them is the Cosine Similarity which was not used, and the second and used one is the Euclidean Distance. Both will be explained below, as well as the reasons why each one was or was not used.

The cosine treats two vectors as unit vectors by normalizing them, giving a measure of the angle between the two vectors. It is possible to use the cosine approach to calculate the similarity between two vectors, named Cosine Similarity. However, despite the fact that this approach provides an accurate measure of similarity, it does not regard the magnitude. That is the reason that this approach was discarded, because the vector magnitude is an important factor while considering the similarity computation.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

Figure 16 - Cosine Distance

The second one, and very similar with the one that was chosen is the Euclidean Distance. The Euclidean Distance, also called Euclidean Metric, is the ordinary distance between two points, in this case, represented by two vectors. Using the Euclidean formula as distance, the dimension space becomes a metric space.

$$\text{sim}(i, j) = d_{ij} = \sqrt{\sum_{k=1}^n (X_i - Y_i)^2}$$

Figure 17 - Euclidean Distance

The Weighted Euclidean Distance was chosen instead of the normal Euclidean Distance, because using this type of computation, we are able to choose, if necessary, what features are more important in the moment of the search. This approach was chosen to facilitate the tests, as well as the weights used in the features could be analyzed more deeply, aiming to find the best ones to be used, giving better recommendations in the real final product. So, the main idea was use the Weighted Euclidean Distance, because in the feature, the importance of each one of the features can be an Amadeus criterion or an airline criterion, based on what they would prefer to provide to the client. The equation is

very similar, just with the weight added to each one of the dimensions, represented by w in the equation below:

Weighted Euclidean Distance

$$similarity = \sqrt{\sum_{i=1}^n W_i (X_i - Y_i)^2}$$

Figure 18 - Weighted Euclidean Distance: configurable features weights

Below is shown a sample example of the similarity computation. In this example, there are two flights, one from Amsterdam to Nice and another from Amsterdam to Toulouse. They are represented by four features and all the features are considered as having the same weight 1. This example is just to illustrate how the computation is done and does not have real features values.

Amsterdam-Nice	Amsterdam-Toulouse
February	March
0,54	0,78
0,74	0,45
0,41	0,35
0,98	0,97

Figure 19 - Example Similarity Computation

Below there is the computation done for the similarity between the two flights, using the Weighted Euclidean Distance.

$$\text{similarity} = \sqrt{(0,54 - 0,78)^2 + (0,74 - 0,45)^2 + (0,41 - 0,35)^2 + (0,98 - 0,97)^2} = 0,3813135$$

4.4. Recommendations

The result of the similarity between two flights is the result of the Weighted Euclidean Distance between them. There will be F different results, where F is the number of the flights in database. As soon as a similarity is calculated, the result is stored in a list, always keeping the respective flight id.

Flight ID	Flight 1	Flight 2	Flight 3	Flight 4
Similarity	0,8754	0,7412	0,3267	0,1485

Figure 20 - List of Similarities

It is important to say that all the similarities results are obtained taking into account each one of the flights vectors and calculate their similarities with the user flight. After the step of the similarities computations, the prototype has all the similarities, between the user flight and each one of the flights in the database. The lower similarity result is, the more its respective flight is similar to the user flight. It means that to find the K-best flights to be recommended is necessary, then, to find the K-lowest similarities. First of all the prototype orders the similarity list. Having the similarities results list ordered, the prototype is able to recommend the flights to the user. To make this ordination were tried some different algorithms as the Quicksort, Heapsort and Selection Sort. The first one, despite of be faster, having a complexity of $O(n^2)$ in the worst case, $O(n \log n)$ in the middle case and $O(n \log n)$ in the best case, this algorithm presented some problems of stack overflow because of its recursion. The Heapsort algorithm, having a complexity of $O(n \log n)$ in the worst case, also presented stack overflow problem despite of its recursion.

To make this list ordering is used the selection sort algorithm. This algorithm presents a complexity of $O(n^2)$ both in the worst and best cases. It is easy to realize that this algorithm has a worse complexity than the other two mentioned. However, this algorithm didn't present any error like the other ones, because it does not use the recursion approach. Furthermore, the execution time of the Selection Sort had been analyzed and was realized that the time is not a crucial point for the whole prototype execution time. The similarity list ordering is done always keeping the respective flights id. After order the similarities list, the prototype is able to make the recommendations. It gets the flights identifications of the K-lowest similarities and displays the result to the user.

5. RESULT ANALYSIS

In this chapter, the implementation decisions made and the final results achieved in the program will be analyzed, judging what the benefits were and what the drawbacks were. Also, future improvements will be commented and finally an updated Gantt chart will be presented for what was really followed.

5.1. Benefits

The result of the Flight Recommendation Engine aims to provide the best alternative destinations and periods to the users. However, as mentioned before, the real result is just obtained after this result is passed through the Affinity tool, which gives them their real prices. As the prototype would be integrated to the Amadeus system, an interface was created just to facilitate the tests and the analysis of the results before this integration. There is the main window, which is composed by the origin, destination and date of the user flight, and by a field to display the results, and there is a settings window. The settings window is able in the tests phase. With that we are able to change the features weights, aiming to realize which one is most important, if it gives plausible results, and for example, to check which features are more important in the similarity computation. As mentioned before, the weights of the features can be an Amadeus criterion or an Airline criterion in the feature, modeling the results according to what Amadeus and the Airline Company want to provide to the user. The output of the prototype is the recommended flights, composed by their origin, destination and month.

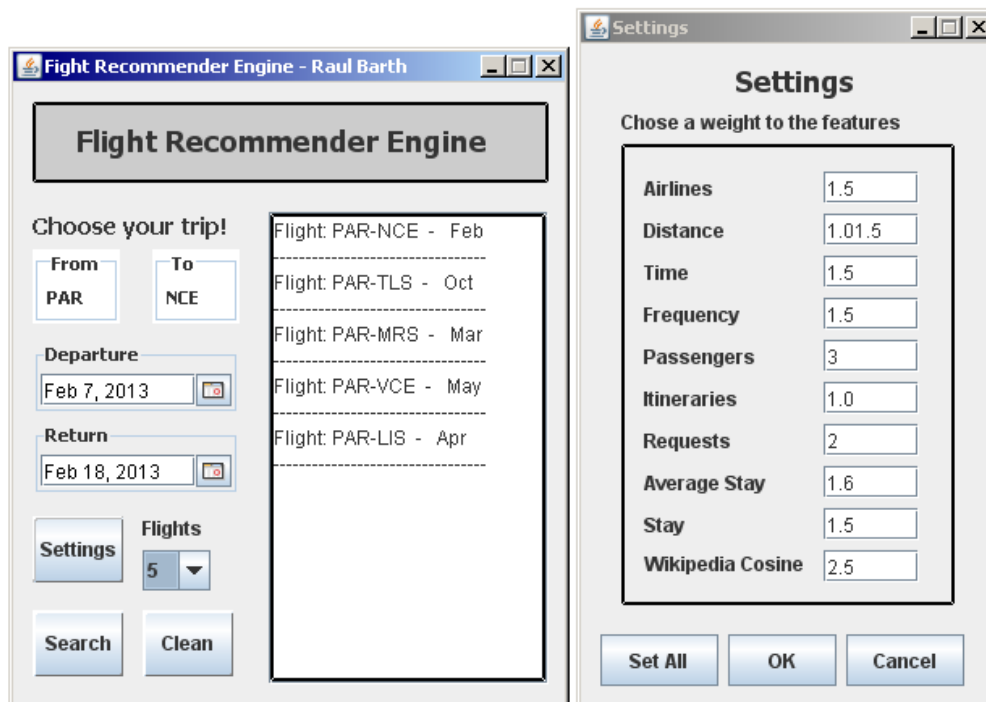


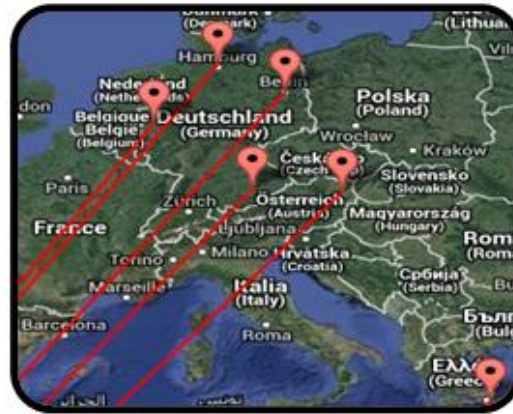
Figure 21 - Prototype Interface

Below, are shown two examples of recommended flights displayed for the user:



- Nice Côte d'Azur (NCE)
- Toulouse (TLS)
- Marseille (MRS)
- Venice (VCE)
- Lisbon (LIS)

Figure 22 - Results: example 1



- Munich (MUC)
- Berlin (BER)
- Hamburg (HAM)
- Vienna (VIE)
- Düsseldorf (DUS)
- Athens (ATH)

Figure 23 - Results: example 2



- Berlin (BER)
- Budapest (BUD)
- Hamburg Airport (HAM)
- Bruxelles National (BRU)
- Munich International Airport (MUC)
- Barcelona (BCN)

Figure 24 - Results: example 3



- Nice Côte d'Azur International Airport (NCE)
- Bucharest (BUH)
- Bordeaux Airport (BOD)
- Marseille Provence Airport (MRS)
- Bergen (BGO)
- Toulouse Airport (TLS)
- Budapest (BUD)

Figure 25 - Results: example 4

The subject of this project can be compared with Amazon Recommendation System. However, there are relevant differences between the two approaches. Amazon Recommendation Systems uses the Collaborative-Filtering, taking into account the user profile and the user rating for products. Meanwhile, this project does not take into account the user profile, due to the fact that it is not taken into account by the environment where the prototype must work. The results provided by the prototype were very useful, achieving the objectives and satisfying Amadeus and TAM desires and requirements.

The initial tests were done using around 25 different input flights, changing the features weights and analyzing the results. For each one of the inputs, were tested 20 different types of configuration. That results in 500 different tests. As mentioned before, as the prototype provides alternative destination, there is not just one good result, what force the evaluation method to be done manually or conceptually. Each one of the input flights gave as result 10 recommended flights, totalizing 5000 recommended flights. All the recommended flights were analyzed, and at first, for results that presented 40% of noisy cities, their respective configurations were discarded. From 20 initial features weights configuration, 5 were discarded. In a second approach, another 25 different flights were taken, and results that presented 20% of noisy cities were discarded, resulting from 15 to 6 acceptable configurations. In the third and last approach, all the 50 used flights were taken, and results with 10% of noisy cities were discarded, resulting in 5 acceptable configurations. So at the end, there were 5 acceptable configurations. That makes possible the configuration be an Amadeus criterion or an Airline Company criterion, based in what type of result they want to provide to the client. That was the only possible way, as discussed internally, to evaluate the prototype. A deeper analysis of the result will be a subject of another internship.

The configuration chosen in this project is shown below, represented by the feature name and its weight.

Airlines	1	Distance	1	Time	1.5	Frequency	1.5	Passengers	2
Itineraries	1	Requests	2	Stay	1.5	Wikipedia	2.5		

The time of the algorithm execution answered very well when increasing the number of the flights in the database. If we had used a Collaborative Filtering without clusters, for example, the time would be considerably greater. The final version of the prototype, working in the TAM Airlines test page operated with a response time of around 4.3 seconds, what can be considered a great response time, taking into account it is within the normal response time of the flight booking page.

As mentioned before in this report, the main goal of this prototype in the area and in the environment where it operates is that it has the results moldable, in other words, changing the features weighs, we change the result of the recommendations. Based on that, the airline company can chose what type of recommendations it wants to provide to its user. For some internal and confidential reasons, sometime it is better to provide flights based more in the popularity of the flights and less in the distance between the origin and the

destination. We also have some cases that the recommendation modeling can depend on where the user is, or when the flight search was done.

5.2. Drawbacks

A difficult found in this project was the evaluation method. As the objective of the prototype is to provide alternative destinations to the user, it means that there is not just one correct result, in other words, there is not a mechanism to analyze if the result is correct or not, because there is not just one possible correct result. However, we are able to analyze and discard wrong results, analyzing what we can call as *noisy* cities that appear in the result, and based on the implausible cities that appear in a result, we are able to discard this result and the settings that were configured for it. The tests were done changing the features weights and observing carefully the respective results obtained for each configuration. Some examples of noisy cities were found when, for a flight from Amsterdam to Nice was recommended a very small city in the east of China, and also, for a flight from Paris to New York, was recommended a very small city in the north of Russia. Therefore, the test phase was done like that, changing the features weights, analyzing the results carefully, and discarding configurations that provided implausible cities, till was found a good configuration that provides plausible results.

5.3. Improvements

The mainly improvement to be done in this prototype is related to the features weights. As mentioned before, the weight of each one of the features is configurable, but it is done in an offline mode, being and Amadeus criterion or an airline criterion. This improvement can be done, for example, testing this prototype in the market, and using some user profile. Its works can be analyzed and for a certain type of flight chosen by the user, we can realize if the user changed his opinion, and then, analyze what recommended flight was chosen. The user profile is not taken into account by the prototype because the environment where it works does not use user profile, in other words, it does not require that the user be logged to search for a flight. That is an important difference between this type of recommendation engine and Amazon recommendations, that takes into account the user profile, as well as the user rating for the products. Another good improvement is related to the database performance. That could be done pre-processing the similarity between all the flights in database, and then grouping them in similar types.

5.4. Amadeus Integration

This section is designated to explain how and why the prototype was integrated to Amadeus systems. The prototype was integrated in the TAM test page, a webpage developed in Portuguese by an Amadeus team. In the TAM webpage, the user searches for a flight, the result of this booking pass throw Amadeus booking engine, is processed by Flex Pricer and also processed by the Flight Recommender Engine. The TAM page where the user chooses the flight and the combination of the two results are shown below:



- CHANGE COUNTRY
- OUTROS SITES TAM
- TAM EM UM CLIQUE

TAM FIDELIDADE

CONHEÇA CADASTRE-SE

Número Fidelidade

Assinatura Eletrônica

LOGIN

Esquedi meu nº fidelidade

Esquedi minha assinatura eletrônica

Buscar no site da TAM

BUSCAR



INSTITUCIONAL

SERVIÇOS

TAM FIDELIDADE

EXPERIÊNCIA

TAM TIPS

CONTATO

SEARCH CHECKIN RETRIEVE CARTÃO DE CRÉDITO

Ida e Volta Somente Ida Várias cidades

[Busca Preço](#) [Busca Pontos](#)

De: Rio de Janeiro (RIO)

Para: Milan - Malpensa (MXP)

Data da Partida: 08/08/2013 Data do Retorno: 14/08/2013

Adultos: 1 Crianças: 0 (2 a 11 anos) Bebês de Colo: 0 (0 a 23 meses)

[Mais de 9 passageiros?](#)

Utilizar pontos Multiplus?

Classe de Serviço: Econômica

Código Promocional

COMPRE SEU BILHETE

CONHEÇA O PARAÍSO ILHAS DE PARATY

DE: SÃO PAULO PARA: RIO DE JANEIRO

R\$ 200^{,00}

- 1
- 2
- 3
- 4

Figure 26 - Integrated Result: search page

YOU MAY ALSO CONSIDER

From: Rio Janeiro To: Milan	Month: September	Price: 4320,99	BOOK	TRIP MAP	DESTINATION MAP
From: Rio Janeiro To: Rome	Month: August	Price: 3699,55	BOOK	TRIP MAP	DESTINATION MAP
From: Rio Janeiro To: Amsterdam	Month: August	Price: 3650,50	BOOK	TRIP MAP	DESTINATION MAP
From: Rio Janeiro To: Madrid	Month: July	Price: 7514,98	BOOK	TRIP MAP	DESTINATION MAP

IDA
De: **Rio de Janeiro (RIO)** Para: **Milan (MXP)** Quinta-feira, 08 de Agosto de 2013

Tarifas de ida com preço similar.

1.734,59 → Seg-05-Ago	1.734,59 → Ter-06-Ago	1.734,59 → Qua-07-Ago	1.903,82 → Qui-08-Ago	2.908,07 → Sex-09-Ago	2.547,34 → Sáb-10-Ago	2.309,08 → Dom-11-Ago
--------------------------	--------------------------	--------------------------	----------------------------------------	--------------------------	--------------------------	--------------------------

+ Voos com troca de aeronave (8)

Legenda: Operado pela TAM Operado por LAN Operado por Cia. aérea membro da Star Alliance
 Operado por Cia. aérea parceira Últimos assentos disponíveis. **+1** Chegada no próximo dia

RETORNO
De: **Milan (MXP)** Para: **Rio de Janeiro (RIO)** Quarta-feira, 14 de Agosto de 2013

Tarifas de volta com preço similar.

2.963,74 → Dom-11-Ago	2.707,67 → Seg-12-Ago	2.496,13 → Ter-13-Ago	2.257,88 → Qua-14-Ago	2.055,25 → Qui-15-Ago	2.117,60 → Sex-16-Ago	2.309,09 → Sáb-17-Ago
--------------------------	--------------------------	--------------------------	----------------------------------------	--------------------------	--------------------------	--------------------------

Figure 27 - Integrated Result: recommended flights

In the bottom, we have the result of the searched flight, with the range of the prices and days, to the departure and return. The Flight Recommender Engine result was added above that, in this case, with four recommended destinations. The recommendations contain the flight origin, flight destination, flight month and flight price. Furthermore, the user can also see the trip map and the destination map of each one of the recommendations, as shown below:

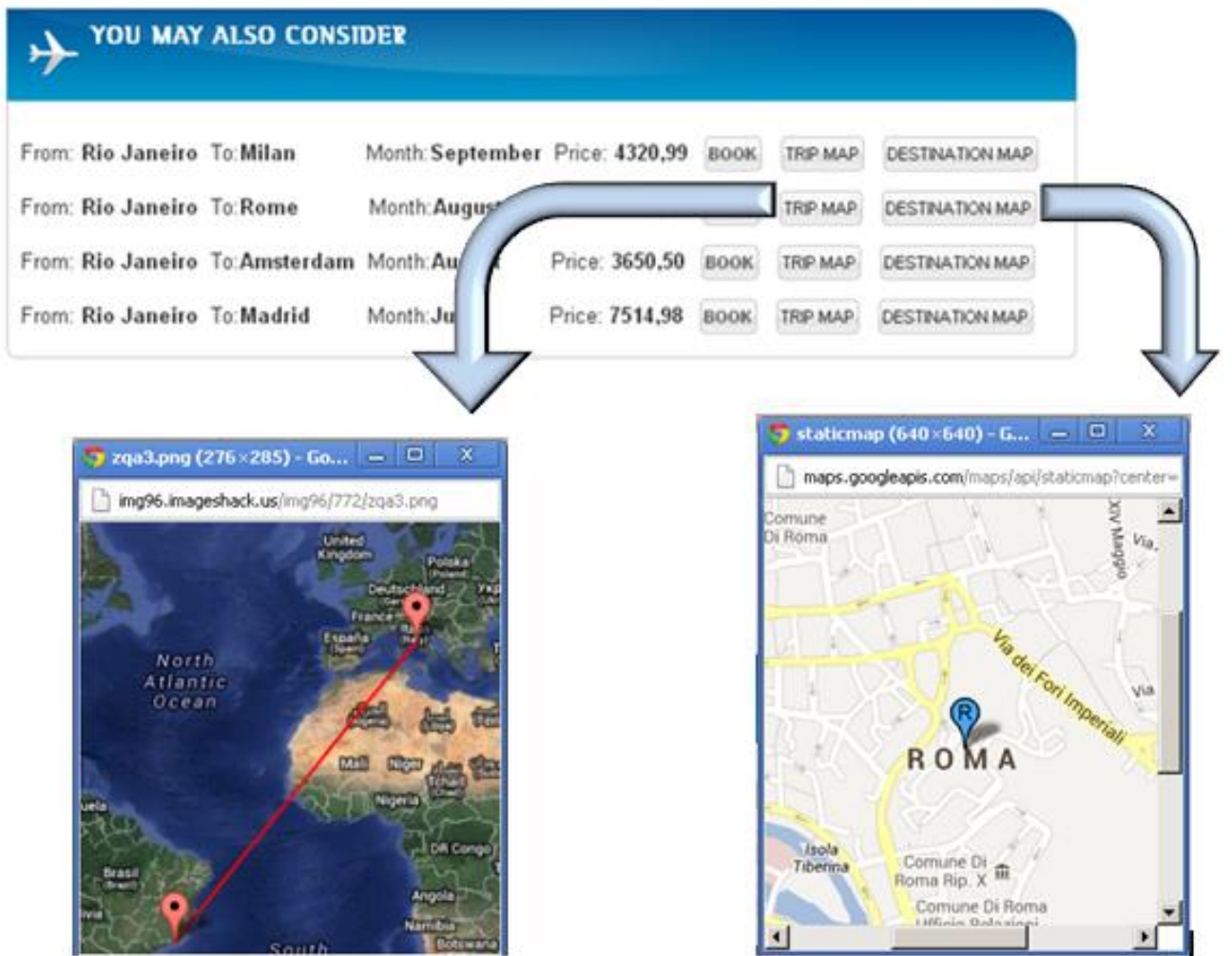


Figure 28 - Integrated Result: trip map and destination map

6. CONCLUSION AND FUTURE WORK

In this project was designed and developed a Flight Recommender Engine, using techniques present in the literature, as *top N-recommendations algorithms*, *Similarity Computation*, *Cosine Similarity* and *Euclidean Distance*. The first part of this project was dedicated to the problem understanding and to reading and thinking about some possible solutions. A lot of papers and thesis were read, aiming to understand a lit bit more about the world of recommendations and the problem itself. After the problem was completely understood, a modeling of the problem and the solution approach was made. The second part was dedicated to the implementation, as explained in this thesis.

One weak point in the development of this project was the fact that the data extraction took a lot of time, more than previously expected. But that did not prejudice the ongoing of the prototype development. In other hand, the extraction was made in a huge quantity of data, around 400Gb, what made this delay not so important, taking into account that as more data was analyzed, more flights information was gotten. That contributed a lot to the correctness of the results.

The prototype presented a good performance, even with a huge amount of data to be processed and analyzed. This performance depends, for sure, on what kind of tools we are using to develop the prototype, as the used database and platforms, as well as the amount of data.

A future work expected for this project is the use of user profile. As explained in the section of related work, there are several methods of recommendations that use the user profile. Using that, the engine developed in this project would be able to recommend flights based on user ratings, for example. However, that depends on how the company where the prototype will be integrated work, if it uses user profile or not.

REFERENCES

- [1] Amazon Recommendations – Item-to-Item Collaborative Filtering. Greg Linden, Brent Smith, Jeremy York. 2003
- [2] Travel Recommendation Engine Internship Amadeus – Imad El Ghalbzouri
- [3] Data Normalization and Standardization. <http://www.benetz Korn.com/2011/11/data-normalization-and-standardization/>
- [4] Apache Cassandra Database. <http://cassandra.apache.org/>
- [5] Hector API - A high level Java client for Apache Cassandra database - <http://hector-client.github.io/hector/build/html/index.html>.
- [6] Selection Sort Algorithm - analysis of the algorithm and its complexity http://pt.wikipedia.org/wiki/Selection_sort.
- [7] Air Traffic Amadeus
- [8] Wikipedia Database XML file. <http://dumps.wikimedia.org/>
- [9] Euclidean Distance. http://en.wikipedia.org/wiki/Euclidean_distance
- [10] Chapter 4: Measures of distance between examples: Euclidean <http://www.econ.upf.edu/~michael/stanford/maeb4.pdf>
- [11] Cosine Similarity. http://en.wikipedia.org/wiki/Cosine_similarity
- [12] Discussion of Similarity Metrics: Cosine Similarity - http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/sphilip/cos.html
- [13] Amadeus internal documentation
- [14] http://en.wikipedia.org/wiki/Collaborative_filtering
- [15] Item-Based Collaborative Filtering Recommendation Algorithms - Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
- [16] Recommender Systems: An Introduction – Jannach, Dietmar; Zanker, Markus; Felfernig, Alexander; Friedrich, Gerhard
- [17] Recommender Systems Handbook – Ricci, Francesco; Rokach, Lior; Shapira, Bracha; Kantor, Paul.
- [18] Collaborative Filtering Recommender Systems (Foundations and Trends(r) in Human-Computer Interaction) - Michael D. Ekstrand, John T. Riedl, Joseph A. Konstan.
- [19] Computation: Computability, Similarity, and Duality – Hong Jia Wei

- [20] Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures – Gomaa, Hassan
- [21] The Art of Software Modeling - Benjamin A. Lieberman
- [22] Data Mining: Concepts and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems) - Micheline Kamber, Jiawei Han
- [23] <http://www.ibm.com/developerworks/>
- [24] Dodge, Y (2003) The Oxford Dictionary of Statistical Terms, OUP. ISBN 0-19-920613-9 (entry for normalization of scores)
- [25] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010.
- [26] Robin Burke , Hybrid Web Recommender Systems
- [27] Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; Riedl, J. T. (January 2004). "Evaluating collaborative filtering recommender systems"
- [28] Takács, G.; Pilászy, I.; Németh, B.; Tikk, D. (March 2009). "Scalable Collaborative Filtering Approaches for Large Recommender Systems"
- [29] Scalable collaborative filtering using cluster-based smoothing. Authors: Gui-Rong Xue, Chenxi Lin, Qiang Yang
- [30] A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce - Zan Huang, Daniel Zeng and Hsinchun Chen
- [31] Item-Based Collaborative Filtering Recommendation Algorithms - Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
- [32] Analysis of Recommendation Algorithms for E-Commerce - Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl

APPENDIX A: CODE

```
static Logger logger = Logger.getLogger("AccessCassandra.class");
private static String clusterName = "FlightRecommender";
private static String columnFamilyName = "flights";
public static CassandraService cs;
public static CassandraService wikiBase;
public static CassandraHostConfigurator cassandraHostConfigurator = new CassandraHostConfigurator();
```

```
logger.getRootLogger().setLevel(Level.INFO);
cassandraHostConfigurator.setHosts("localhost:9160");
cs = new CassandraService(clusterName, cassandraHostConfigurator, keyspaceName, columnFamilyName());
try{
    cs.getUserFlight(date, onUser, idFlight, columnFamilyName);
} catch (Exception e) {System.out.println("No other flights available"); System.out.println(e.getMessage());}
```

Figure 29 - Database Access

```
CqlQuery<String,String,String> cqlQuery = new CqlQuery<String,String,String>(keySpace, StringSerializer.get(),
    StringSerializer.get(), StringSerializer.get());
cqlQuery.setQuery("SELECT * FROM flights WHERE origin = '"+userOrigin+"'");
QueryResult<CqlRows<String,String,String>> result = cqlQuery.execute();
```

```
CqlQuery<String,String,String> cqlQuery = new CqlQuery<String,String,String>(keySpace, StringSerializer.get(),
    StringSerializer.get(), StringSerializer.get());
cqlQuery.setQuery("select * from flights WHERE onid = '"+userOrigin+"'-"+userDest+"' and "
    + "monthFlight = '"+date+"'");
QueryResult<CqlRows<String,String,String>> result = cqlQuery.execute();
```

Figure 30 - Getting Flights from Database

```
        difference += weight * (Math.pow(flightONE.get(i) - flightTWO.get(i), 2.0));
    }
} catch (Throwable ex) {
    System.out.println("Features Missing");
}
return (1-Math.sqrt(difference));
```

Figure 31 - Similarity Computation

APPENDIX B: PROJECT DESCRIPTION (TG1)

Design and Implementation of a Flight Recommendation Engine

Raul Sérgio Barth

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)

Porto Alegre - RS - Brazil

raulbarth@gmail.com

Abstract. *The use of recommendation systems has proven to increase sales on most e-commerce platforms. This type of system aims to provide to the user more options, more items that he could be interested in, giving to him more comfort and practicality, and also providing to the company a greater chance to increase its products and services sells. This project is inspired Amazon Recommendations Systems, but with a different approach. This project was developed in Amadeus Company, a leading IT solutions provider created by Air France, Iberia, Lufthansa, and SAS as a Global Distribution. The subject of this project consists in creating a prototype of an intelligent application, which provides alternative destinations and periods to Amadeus clients. In the first part of this report, the context of this project is presented, and then the motivation and the environment, as well as the analysis that was made to solve the problem and the modeling of the design and the Flight Recommendation Engine itself in a theoretical and technical analysis.*

Resumo. *O uso de sistemas de recomendação tem provado aumentar as vendas na maioria das plataformas de e-commerce. Esse tipo de sistema busca prover ao usuário mais opções, mais itens que possam ser de seu interesse, dando a ele maior conforto e praticidade, e também provendo à companhia maior chance de aumentar a venda de seus produtos e serviços. Esse projeto é inspirado em Amazon Recommendations Systems, mas com uma abordagem diferente. Esse projeto foi desenvolvido na Amadeus Company, uma provedora líder de soluções de TI criada pela Air France, Iberia, Lufthansa, e SAS como um Sistema Global de Distribuição. O sujeito desse projeto consiste na criação de um protótipo que providencia destinos e períodos de viagens alternativos para os clientes da Amadeus. Na primeira parte desse relatório é apresentado o contexto desse projeto, e então são apresentadas a motivação e o ambiente de desenvolvimento, bem como a análise que foi feita para solucionar o problema e a modelagem do sistema, chamado Flight Recommender Engine.*

1. Introduction

In a constantly evolving and competitive environment, selling platforms, services, and travel merchants in particular, companies are faced to the challenge of offering to their customer relevant content and options. On a longer term, customers who find the offered content relevant are more likely to use the service again. The recommendation services are very important nowadays, as for the companies who want to sell the more products and services, keeping the clients satisfied and aggregating new ones, as for the users who want the most comfort and quickly service as possible.

This project was created based on that, which the subject consists in providing flights with alternative destinations and periods to the clients. More specifically, the subject consists in creating a prototype of an intelligent application that uses the similarity computation approach, taking into account some business intelligence information, as well as some unstructured information from web, aiming to recommend the best flights to the client.

The purpose of this project is to provide to the user other flights options that could be of his interest. It is done using the notion of recommendations algorithm, having a huge mathematical part, making use of Euclidean Distance and Cosine similarity applied to an industry problem. The development of this engine required a huge data extraction and analysis, as well as a considerable mathematical approach.

2. Context

2.1. Motivation

The technological progress and the constant improvement of the online technologies are giving the opportunity to the companies to increase their markets and sells, as well as giving to the users even more facilities, options and comfort. The recommendation services are very important nowadays, as for the companies who want to sell the most products as possible, keeping the quality and keeping the clients satisfied and aggregating new ones, as for the users who want the most comfort and quickly service as possible..

For that reason the Flight Recommender Engine was developed, to provide to airline companies users more flights options that serve their demands. In order to obtain a full integration experience, and make the prototype as a final product, at the end, the Flight Recommender Engine was integrated in the TAM Airlines page, using some Amadeus tools related with the process of booking, searching and displaying flights.

2.2. Problem Statement

The challenge of this project is the development of a prototype which aims at recommending the best destinations to the user. The meaning of best destinations, in the approach of this internship, is alternative destinations and periods. The Flight Recommender Engine does that using some business intelligence information as geolocation, flight distance, flight time, itinerary popularity and number of passengers that used to take this flight, and also unstructured information from web, as Wikipedia. In total, there are 26 different types of information, describing each one of the flights in the database and the user flight.

It is clear that, usually, a user that searches for a flight is decided where to go, but sometimes it is not true. Some Amadeus booking logs had been analyzed and was verified that, commonly, there are flights that are booked but not purchased. This analysis shows that the user is not always sure about his choice and sometimes he changes his opinion. And that is what the Flight Recommender Engine is intended to cover. Based on the fact that sometimes the user is not sure about where to go, the prototype developed in this project aims to provide to this user good alternative destinations and periods, making him chooses a flight, decreasing the case of not finalized bookings, increasing the sales.

The challenge, then, is to find the most similar flights to the user flight. The approach used is to see all these flights as vectors in the space, and then, using the Weighted Euclidean Distance, calculate the similarity between them. As there are 26 different information types, then, there are 26 dimensions in the space where the flights are. The figure below illustrates how the prototype sees all the flights in the space.

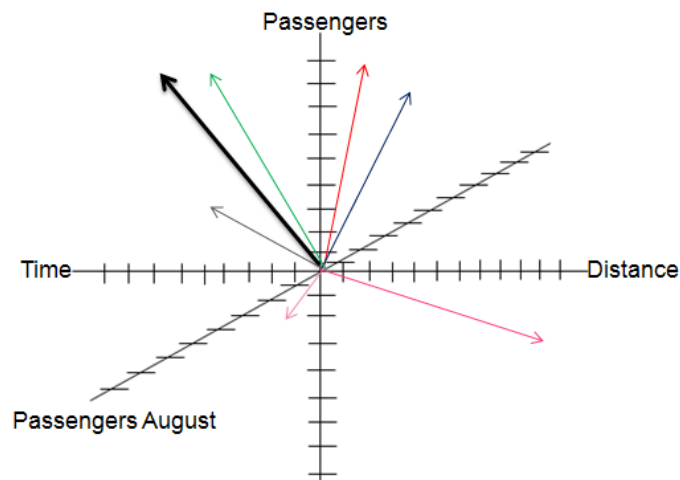


Figure 32 - Flights: vectors in space

As there is one information type that is the flight distance, we will have one dimension in the space that represents this information, and the same thing for all the information types. Therefore, as there are 26 different information types characterizing each one of the flights, the space where the flights are represented have 26 dimensions.

3. Related Work

There are several approaches related to recommendation engines, the collaborative filtering, content-based filtering and hybrid recommender systems, for example.

3.1. Collaborative Filtering

The collaborative filtering has its methods based in users behaviors preferences or activities and predicting what users will like most, based on their similarity to other users. One of the most famous examples is the Amazon Recommender System, which uses the item-to-item collaborative filtering. This method is used to make automatic predictions - filtering - about the interests of the users, collecting information and preferences of several users. This collaborative filtering is done supposing that if a certain person A has the same opinion as a person B on an issue, then, on a

different issue, A is more likely to have B's opinion. It means that this type of algorithm requires having user profile.

This recommendation method requires, usually, the user participation - i.e. the profile analysis - and algorithms capable of match people with similar interests. Basically, in this method, the user expresses his opinion by rating the items in the system, then the systems matches the user's ratings against other users and finds users with similar interests. With these similar users. In other words, the system looks for users that share the same rating patterns with the actual user and then it uses the ratings from those like-minded users that were found to calculate a prediction for the actual user.

One of the most known collaborative filtering is the one used in Amazon, called item-based collaborative filtering. This recommendation system uses the idea that who bought a product x also bought a product, and it proceeds in an item-centric manner. There are three types of collaborative filtering, the memory-based, model-based and hybrid. These types will not be approached in this report.

3.2. Content-based Filtering

This recommendation method is based on the item description and a profile of the preferences of the user. The items are described by keywords and a user profile is built, indicating the type of items that this user likes. Basically, the algorithm of this type of method try to recommend items that are similar to those that a user liked in the past, i.e., the algorithm compares items previously rated by the user with candidate items, and the best matches are recommended. As the Collaborative-Filtering, this method also algorithm requires having user profile. To make the abstraction of the items features in the system, an item presentation algorithm is used. In this case is used the tf-idf representation, that represents the items as vectors in the space. This method is also used in this project, and will be explained latter.

The system uses a model of the user preference and a history of the user interaction to create a user profile. The system also creates a content-based profile of users based on a weighted vector of item features. This weights are important because they denote how important each feature is to the user and can be computed from individually rated content vectors using several techniques. These weights can be decreased or increased, based in the user opinion, as a like or dislike button, for example.

3.3. Hybrid Recommender Systems

The hybrid recommender systems is a method which combines the collaborative-filtering and the content-based filtering, and can be more effective in some cases. This method can be implemented in different ways, using the other two methods. The first way to do that is adding content-base capabilities to a collaborative-based approach. Also, it can be done by making content-based and collaborative-based predictions individually and then combining them, or by unifying the approaches into one model. There are several studies that proving that this method can provide more accurate recommendations than the other two pure methods. A famous example of system that uses this method is Netflix, that make recommendations by offering movies that share similarities with films that a user has rated higly - content-based filtering - as well as by comparing the watching and searching habits of similar users - collaborative filtering. The term *hybrid recommender system* describes recommender system which combines multiple recommendation methods to produce the recommendations.

4. Project

4.1. Development Setup

The platform to be used in this project is Windows. There is no a specifically reason for this choice, but all the company uses Windows Platform, so it could be easier to solve possible problems and to integrate the application with the whole system.

The Flight Travel Recommender will be developed in Java, using the Eclipse Tool. Some of scripts could be developed in Python, to help in the offline step, in the data recovery. Some of database language will also be needed, to build all the information needed to create the flights vectors.

The application will be developed in Eclipse. A database will also be needed as written before, and, for that, maybe the Cassandra Database will be chosen. At the end, the Git Tool is used to save and protect the project.

4.2. General Specification

This project aims to provide to the user other flights options that could be of his interest. It will be done using the N-top recommendations algorithm, having a huge mathematical part, using, for example, the Euclidean Distance to calculate the similarity between all the flights, and then, to recommend the N best flights.

In general, this project can be divided into two parts, one that will be made in an offline mode, in other words, this part is done just once, and the other that will be made in an online mode, which means that this part is done in real time, in the same instant that the user is choosing his flight .

In the offline mode are the extraction of the needed data, and also the building of the database tables that will be used after. In the offline mode, for example, will be done the data recovery, the extraction of the relevant information, and will be built the database tables with the data that will be used to create the vectors of all flights. The mainly part of the offline step is the recovering of the flights features that are responsible for the building of the vectors of each one of the flights. These features can be, for example, the total population of the destination, the number of visitors that go to the destination city, the flight price, the flight time, the quantity of demand for the city, the average of stay in the city, city size, and number of flights departed from the city. It is important say that all these features will be normalized before be used to build the flights vector. That normalization will be done to the features become a value between 0 and 1, so then, the similarity result will also have a value between 0 and 1.

After this first application in the offline mode, we will have almost all the useful data to make the recommendations, the only data that must be recovery yet is the flight chosen by the user. That is the reason of the second part is done in an online mode. This step is responsible for recovering the user flight and calculate it similarity with all the others flights from the database, aiming to find the top N-recommendations.

Given the current flight, in other words, the flight chosen by the user, the engine must recommend the best N-flights to the user. It is clear that these recommended flights must be as similar to the current flight as possible. To do that, all the flights will be considered as vectors in the space on N dimensions, so then, we can calculate the similarity between all of them using the Euclidean Distance, which one is shown below:

$$sim(i, j) = d_{ij} = \sqrt{\sum_{k=1}^n (X_{ik} - X_{jk})^2}$$

Figure 2: Euclidean Distance

A huge quantity of data will be analyzed to build the flights vectors, and then the similarity between the current flight and all the flights from the database will be calculated, making a vector of similarity. After that, this vector will be ordered in decreasing order, and the flights that have the N-biggest similarities will be taken and recommended to the user.

Following, we can see a simplified example of some flights from the database:

Flight 1		Flight 2		Flight 3		Flight 4	
10/11/13	Date	10/11/13	Date	10/11/13	Date	10/11/13	Date
PAR	Origin	PAR	Origin	NIC	Origin	PAR	Origin
LON	Distance	LON	Distance	POR	Distance	NIC	Distance
1400	Price	1600	Price	1500	Price	1450	Price

Figure 3: Flights from database

Taking the respective features, for each one of the flights above and using a normalization equation, we will have these flights represented as the following vectors:

Flight 1	Flight 2	Flight 3	Flight 4
10112013	10112013	10112013	10112013
0,78	0,78	0,27	0,78
0,45	0,45	0,16	0,27
0,75	0,86	0,79	0,76

Figure 4: Flights Vectors

The same will be done with the flight chosen by the user, and then, the Euclidean Distance will be used to calculate the distance, it means the similarity, between the user flight and all the other flights. And at the end, the biggest similarities values will be taken and the respective flights recommended to the user.

The results display was not taken into account yet, but that is also a important part of the project. It is necessary to know where to show the results, in other words, the recommended flights, to the user.

It is important to say that this was a very simplified example, as of flights, as of vectors building. In the database will be manipulated a very high quantity of data. Just in the data extraction that is being made in the log files, what can be considered a database, around 400Gb of data is being analyzed. More data will be analyzed, from other files and sources.

4.3. System Overview

The prototype has as input the origin, the destination the period of the user flight. Having this data, the prototype accesses the database and gets all the data related to this specifically flight. As was mentioned before, all the flights in this scope are seen as vectors in an N-dimension space, so then, the problem is basically to find the nearest neighbors of the user flight. To be able to follow this approach, the prototype builds a vector which represents the user request, composed by the data retrieved in the database. The same thing, then, is done for all the flights in the database. At the end, the Flight Recommender Engine calculates the similarity between the user flight and all the database flights, finding the most similar, and recommending them to the user.

Therefore, the prototype's output is the K-best flights, where K is the number of recommended flights shown to the user, composed by the flight origin, flight destination and the flight month. The recommended flights are combined with the user flight booking and the, shown to the user. The process, in a general illustration, is shown in the figure below.

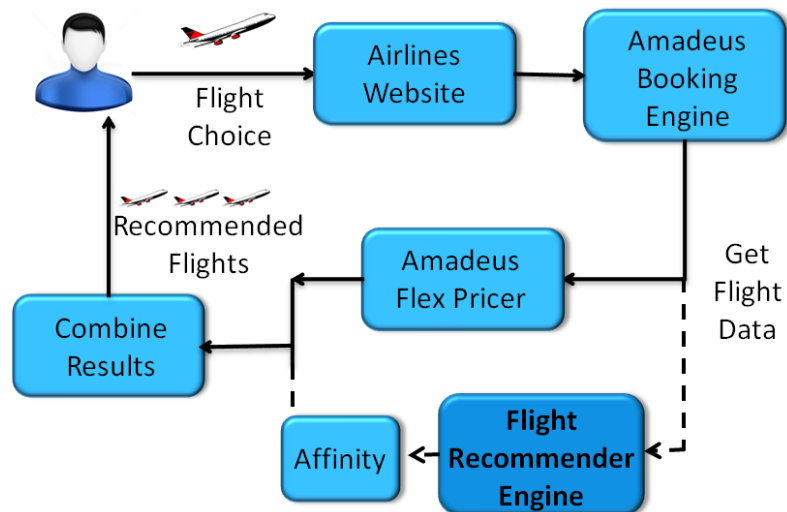


Figure 5 - Environment Overview

In the airlines company website, the user searches for a flight. The result of this choice passes through the Amadeus Booking Engine, responsible for pre-booking this flight with its respective characteristics, as origin, destination, date and number of passengers, in other words, it is responsible for all booking-dependent functionalities. After that, the result is sent to another tool,

called Flex Pricer, which gives to the user the cheapest price for a period of time and the price of different class in a date. The Flight Recommender Engine works in the same level as the Flex Pricer. It means that the prototype receives the user flight data given by the Booking Engine and processes that, obtaining the best recommendations. After that, the result is sent to Affinity Shopper, which is used to get the real prices of the recommended flights. Then, the result of the Flex Pricer are combined with the results of the Flight Recommender Engine, and shown to the user.

4.4. Architecture

The *Figure 7* represents all the process done by the prototype, since the recovery of the user input request, passing through the similarity computation, and ending in the recommended flights. The prototype can be divided into two modes, the offline mode and the online mode. The offline mode is responsible for all the pre-process, as the data extraction, data normalization and data storage in database. The online mode is related to the online process, responsible for getting the user flight, transforming it and all the database flights in vectors represented by their respective features, and responsible for the similarity computation between all of the flights and the user flight, managing the results and recommending the best destinations and periods.

The general structure of the prototype is shown below:

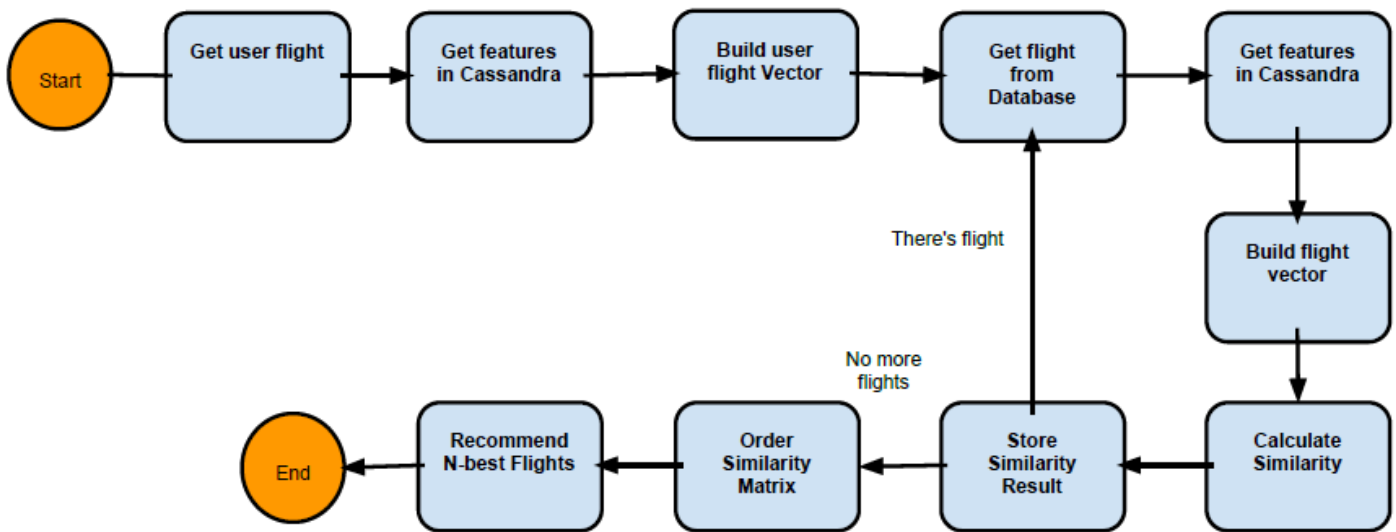


Figure 6 - Flight Recommender Engine: General Structure

This figure represents a general overview of how the prototype works. First of all, it gets the user flights, the input of the prototype. Then, the prototype gets in the database the respective features of this flight and builds a vector with them, representing the flight in the space. After that, for each one of the flights in database, the prototype gets their respective features, builds their vectors, and calculates the similarity between each one of them and the user flight. This process is done till there is no more flights in database to be analyzed. After that, with all the similarities computed, they are ordered and the most similar flights are recommended to the user.

4.5. Data

The data needed by the prototype is all the data types that characterize the flights, as the departure time, arrival time, distance between origin and destination, and so many others. These data were obtained from three different sources, shown below:

- Amadeus Logs:

Around 300Gb of data was analyzed. From that source, two different information types were obtained. The first one is the number of requests that were made for a certain trip (origin-destination), the second one is the average stay duration, which is the total stay duration (number of days between departure and return) divided by the number of requests for a certain origin-destination. All the data from Amadeus Logs was needed to be parsed aiming to extract just the relevant information.

- Air Traffic Amadeus Tool

The Air Traffic is an Amadeus tool. This tool was used to search flights data, considering itineraries which have as origin a European city or as destination an European city (World-Europe and Europe-World). From these itineraries, were obtained data as the number of airlines that support each route, the flights time, distance between cities, and so many others. That is the most important data source used in the project. All the information from Air Traffic Amadeus was obtained in CSV files and was needed to be parsed aiming to extract just the relevant information for the prototype.

- Internet Data

The internet data was used to obtain the characteristics of all the cities in the database. This data were obtained from Wikipedia. Explaining breathily – it will be deeply explained later – the prototype gets all the texts in Wikipedia related to each one of the cities in the database. Then, two cities can be compared taking into account their Wikipedia texts. Using this source, we can recommend destination which have the same characteristics as the user destination

4.6. Flight Features

Nine different information types were taken into account, each one of them is considered as one feature of the flight, representing one dimension in the space. The types are listed below, and after they will be detailed, resulting in a total of 26 different information types.

- Airlines: number of airlines that take a certain route (origin-destination)
- Distance: distance between origin-destination
- Time: flight time, time spent by the airplane to go from the origin to destination.
- Frequency: number of flights that take a certain route (origin-destination)
- Passengers: number of passengers that take certain route (origin-destination)
- Itineraries: number of different itineraries that the flight can take to go from the origin to the destination (different scales and airports)
- Requests: number of times that the route was requested in the Amadeus site
- Average Stay: number of days between Departure-Return divided by the number of requests for a certain route

The features above are called as general features, and they are divided into four types:

- O&D (Origin-Destination): the features of this type are related to the whole trip, in other words, to the origin and the destination of the flight. As example, can be mentioned the distance between the origin and the destination and the time spent by the flight to go from that origin to that destination. Is important to say that the values of these features are related to the whole year.

- O&D-Period (Origin-Destination-Period): this type is almost the same as the first one, but it has a relation to the flight period. While the O&D type values are related to the whole year, the values of O&D-Period are related to each month of the year. As an example can be mentioned the number of passengers that take a certain route in August and the number of passengers that take the same route in January.
- Destination: the features of this type are related just to the destination of the flight. As examples can be mentioned the number of passengers that goes to this destination by year and the number of times that this destination was searched in the Amadeus site.
- Destination-Period: this type is similar to the last one, except to the fact the period, in this case, is each month of the year, as the O&D-Period type. AS example, there are the number of passengers that goes to this destination in September.

4.7. Similarity Computation

In this section will be explained the similarity computation that is done by the prototype, as well as how it is done and what are the results. The challenge consists in imaging the flights as in a dimension space, each flight represented by its respective vector, composed by the flight features. Similarity computation aims to find the nearest neighbors of the user flight, in other words, the flights that are most similar to the user flight. This is the base for the recommendations indeed.

4.7.1. Weighted Euclidean Distance

Two approaches were analyzed to compute the similarity, the first of them is the Cosine Similarity which was not used, and the second and used one is the Euclidean Distance. Both will be explained below, as well as the reasons why each one was or was not used.

The cosine treats two vectors as unit vectors by normalizing them, giving a measure of the angle between the two vectors. It is possible to use the cosine approach to calculate the similarity between two vectors, named Cosine Similarity. However, despite the fact that this approach provides an accurate measure of similarity, it does not regard the magnitude. That is the reason that this approach was discarded, because the vector magnitude is an important factor while considering the similarity computation.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

Figure 7 - Cosine Distance

The second one, and very similar with the one that was chosen is the Euclidean Distance. The Euclidean Distance, also called Euclidean Metric, is the ordinary distance between two points, in this case, represented by two vectors. Using the Euclidean formula as distance, the dimension space becomes a metric space.

$$sim(i, j) = d_{ij} = \sqrt{\sum_{k=1}^n (X_i - Y_i)^2}$$

Figure 8 - Euclidean Distance

The Weighted Euclidean Distance was chosen instead of the normal Euclidean Distance, because using this type of computation, we are able to choose, if necessary, what features are more important in the moment of the search. This approach was chosen to facilitate the tests, as well as the weights used in the features could be analyzed more deeply, aiming to find the best ones to be used, giving better recommendations in the real final product. So, the main idea was use the Weighted Euclidean Distance, because in the feature, the importance of each one of the features can be an Amadeus criterion or an airline criterion, based on what they would prefer to provide to the client. The equation is very similar, just with the weight added to each one of the dimensions, represented by w in the equation below:

Weighted Euclidean Distance

$$similarity = \sqrt{\sum_{i=1}^n W_i (X_i - Y_i)^2}$$

Settings	
Chose a weight to the features	
Airlines	1.0
Distance	1.0
Time	3.5
Frequency	1.0
Passengers	4
Itineraries	1.0
Average Stay	5
Stay	1.0
Wikipedia Cosine	10

Buttons: Set All, OK, Cancel

Figure 9 - Weighted Euclidean Distance: configurable features weights

Below is shown a sample example of the similarity computation. In this example, there are two flights, one from Amsterdam to Nice and another from Amsterdam to Toulouse. They are represented by four features and all the features are considered as having the same weight 1. This example is just to illustrate how the computation is done and does not have real features values.

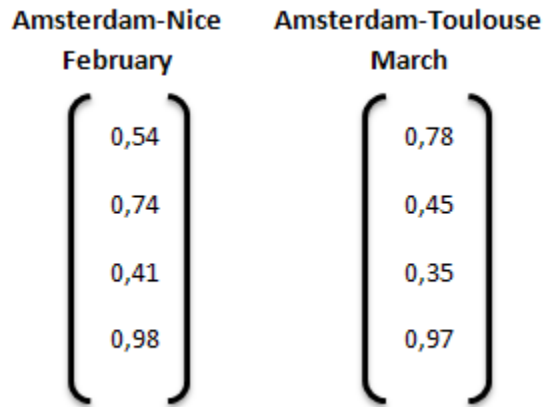


Figure 10 - Example Similarity Computation

Below there is the computation done for the similarity between the two flights, using the Weighted Euclidean Distance.

$$similarity = \sqrt{(0,54 - 0,78)^2 + (0,74 - 0,45)^2 + (0,41 - 0,35)^2 + (0,98 - 0,97)^2} = 0,3813135$$

4.8. Recommendations

The result of the similarity between two flights is the result of the Weighted Euclidean Distance between them. There will be F different results, where F is the number of the flights in database. As soon as a similarity is calculated, the result is stored in a list, always keeping the respective flight id.

Flight ID	Flight 1	Flight 2	Flight 3	Flight 4
Similarity	0,8754	0,7412	0,3267	0,1485

Figure 11 - List of Similarities

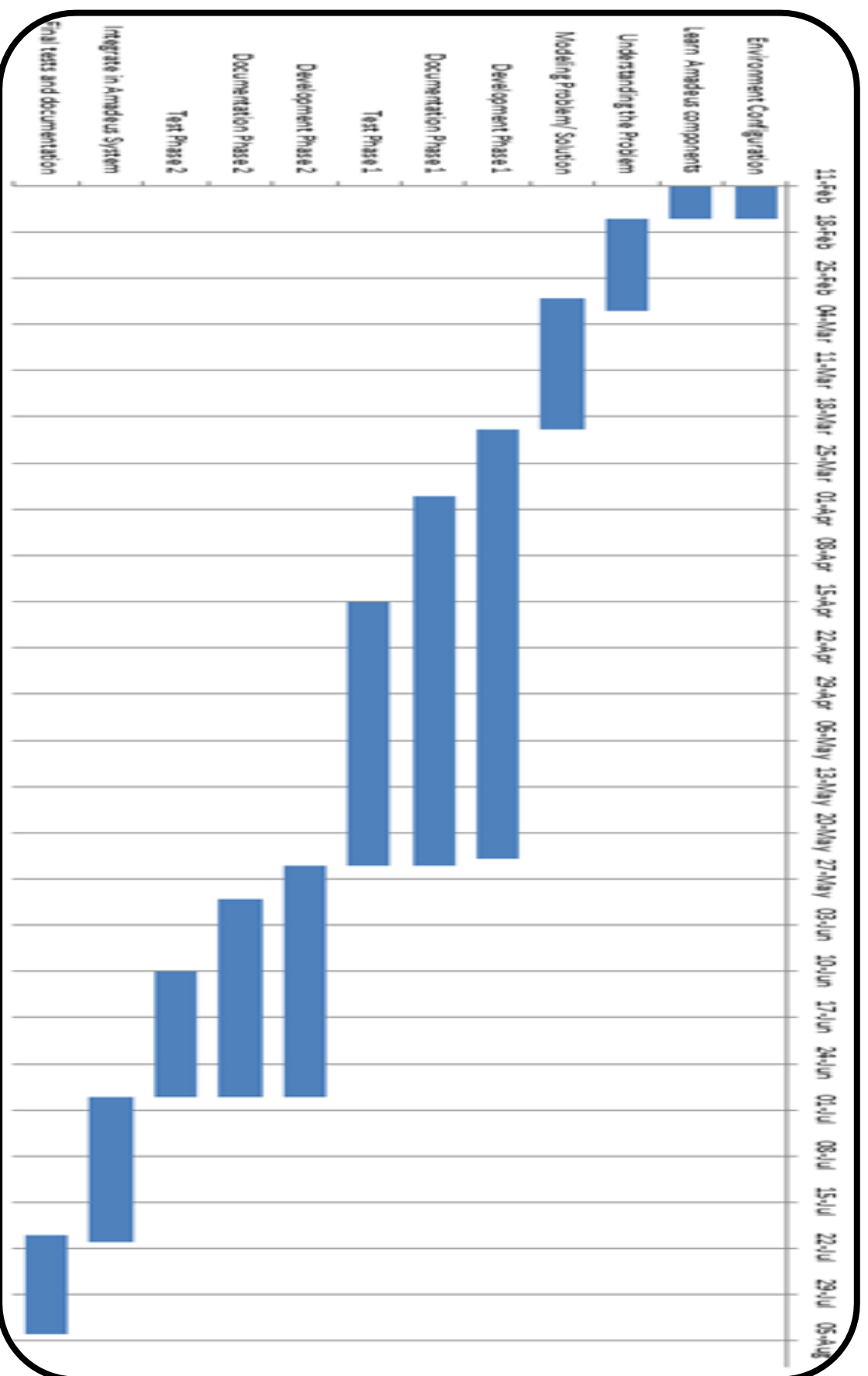
It is important to say that all the similarities results are obtained taking into account each one of the flights vectors and calculate their similarities with the user flight. After the step of the similarities computations, the prototype has all the similarities, between the user flight and each one of the flights in the database. The lower similarity result is, the more its respective flight is similar to the user flight. It means that to find the K-best flights to be recommended is necessary, then, to find the K-lowest similarities. First of all the prototype orders the similarity list. Having the similarities results list ordered, the prototype is able to recommend the flights to the user. To make this ordination were tried some different algorithms as the Quicksort, Heapsort and Selection Sort. The first one, despite of be faster, having a complexity of $O(n^2)$ in the worst case, $O(n \log n)$ in the middle case and $O(n \log n)$ in the best case, this algorithm presented some problems of stack overflow because of its recursion. The Heapsort algorithm, having a complexity of $O(n \log n)$ in the worst case, also presented stack overflow problem despite of its recursion.

To make this list ordering is used the selection sort algorithm. This algorithm presents a complexity of $O(n^2)$ both in the worst and best cases. It is easy to realize that this algorithm has a worse complexity than the other two mentioned. However, this algorithm didn't present any error like the other ones, because it does not use the recursion approach. Furthermore, the execution time

of the Selection Sort had been analyzed and was realized that the time is not a crucial point for the whole prototype execution time. The similarity list ordering is done always keeping the respective flights id. After order the similarities list, the prototype is able to make the recommendations. It gets the flights identifications of the K-lowest similarities and displays the result to the user.

5. Chronogram

Below is the initial planning of the project. This planning includes all the steps to be followed during the development, starting with the understanding of the problem, the applications already developed and the recommendation algorithms, the modeling step, which includes model the problem and the solution, and two implementation steps. In the first one will be implemented the engine, starting from scratch, while in the second one some upgrades and improvements will be done. In the final step, the final tests will be done, and also the user guide and the final report



6. Conclusion

In this project will be designed and developed a Flight Recommender Engine, using techniques present in the literature, as top N-recommendations algorithms, Similarity Computation, and Euclidean Distance. In the first, a lot of papers and thesis were read, aiming to understand a lit bit more about the world of recommendations and the problem. The problem was already completely understood, a modeling of the problem and the solution approach was already made. From now, it is necessary to do the extraction of the needed data, build the respective flights vectors, and implement a form of display the results. The process of data extraction is being done, and it will take a considerable time, taking into account that as more flights features we have, more precise will be the similarity computation, and then, the recommended flights.

In conclusion, I am certain that this project will be developed carefully and the final results will be satisfactorily good. This is an important project, that aims the development of a prototype in a branch of technology that is constantly growing.