Master’s Degree programme – Second Cycle (D.M. 270/2004) in Informatica – Computer Science

Final Thesis

Destination Monitor design for benchmark evaluation of Tourism Product
Analysis of indicators and impacts

Supervisor
Ch. Prof. Agostino Cortesi

Graduand
Giulia Schiavon
Matriculation Number 826788

Academic Year
2015 / 2016
ACKNOWLEDGEMENT

First of all, I wish to acknowledge my supervisor Agostino Cortesi, without him starting and developing this Master’s Degree Thesis would not have been possible.

I would also like to thank my sister Claudia and my brother-in-law Michele, who welcomed me into their home during the development of this thesis and corrected my mistakes and my terrible English.

I would also thank my boyfriend Alessio, who trusted in me during these two years and helped me in every moment.

I must express my very profound gratitude to my parents, who supported me during my years of study.
Abstract

In this Master’s Degree Thesis we will conjecture an algorithm to define the Appeal in a Destination Monitor context.

The goal consists in identifying a weighted point of view from the customers’ feedback and statistical information. The final result is to be used as a comparison between Tourism Products of the same kind. The algorithm is based on defined indicators and the weighted average, a special counting in which each component is weighted differently.

The last contribution is the design of a database used to save the data retrieved from the Web and from Tourism Product sources, as well.
Table of Content

ABSTRACT ............................................................................................................... V

TABLE OF CONTENT ......................................................................................... XIII

INDEX OF FIGURES ............................................................................................... XVIII

INDEX OF TABLES ................................................................................................... XVIII

INDEX OF CODE .................................................................................................... XX

INTRODUCTION ....................................................................................................... 1

1. DESTINATION MONITOR .................................................................................. 7

1.1 ACTORS IN TOURISM ..................................................................................... 8

1.1.1 TOURISM PRODUCERS ........................................................................... 10

1.1.2 DISTRIBUTORS ......................................................................................... 13

1.1.3 CONSUMERS ............................................................................................. 14

1.2 PRE-EXISTENT DESTINATION MONITORS .............................................. 16

1.3 PURPOSE OF THIS THESIS .......................................................................... 21
2. SERVICE-MANAGER SYSTEM........................................... 23

2.1 DATA RETRIEVAL .................................................... 24

2.1.1 APPEAL............................................................ 27

2.1.2 CRAWLER ......................................................... 28

2.1.3 PAGERANK .......................................................... 32

2.1.4 WORDRELATIONS .............................................. 35

2.2 INDICATORS DEFINITION ......................................... 38

3. MANAGER-CLIENT SYSTEM........................................ 45

3.1 PRODUCT MANAGER-CLIENT RELATIONSHIP ............. 46

3.2 REQUESTS-USER FRIENDLY ...... E R R O R E . I L S E G N A L I B R O N O N È DEFINITO.

3.2.1 VIEW FROM PRODUCT MANAGER’S SIDE ............ 52

3.2.1.1 Keywords .................................................... 55

3.2.2 VIEW FROM THE CLIENT’S SIDE ......................... 58

4. A NEW PROPOSAL..................................................... 61

4.1 INDICATORS DEFINITION ......................................... 62
4.1.1 City ................................................................. 62
4.1.2 Typology ......................................................... 63
4.1.3 Visitors Number .............................................. 64
4.1.4 Period ............................................................ 64
4.1.5 Evaluation Number ........................................... 65
4.1.6 Evaluation Quality ........................................... 65
4.1.7 Area ............................................................... 66
4.1.8 Cost ............................................................... 67
4.1.9 How to Arrive .................................................. 67
4.1.10 PageRank Results ............................................. 67

4.2 Database Structure ............................................... 68
4.2.1 Person .......................................................... 69
4.2.2 Client ........................................................... 71
4.2.3 Manager ......................................................... 71
4.2.4 Review .......................................................... 72
4.2.5 Tourism Product .............................................. 73
4.2.6 How To Arrive ............................................. 76

4.2.7 City ............................................................ 78

4.2.8 Typology ......................................................... 79

4.2.9 Holiday Maker .................................................. 79

4.2.10 Overnight Stay ............................................... 79

4.2.11 Service In Loco .............................................. 80

4.2.12 Transport ....................................................... 80

4.3 The Appeal Definition .......................................... 81

4.3.1 Comparison of two potential Tourism Products

5. Conclusions ....................................................... 101

Glossary ............................................................... 103

References ........................................................... 109

Books and Papers .................................................. 109

Web Sites .............................................................. 112

Appendix A ............................................................. 117
CRAWLER CODE ........................................................................ 117

APPENDIX B .............................................................................. 121

PAGERANK CODE ...................................................................... 121

APPENDIX C .............................................................................. 127

WORDRELATIONS CODE ...................................................... 127

APPENDIX D .............................................................................. 135

APPEAL AND COMPARE CODES ........................................... 135
Index of Figures

Figure 1 ~ Tourism................................................................. 9
Figure 2 ~ Data retrieval............................................................. 24
Figure 3 ~ UML schema Product Manager ..................................... 25
Figure 4 ~ Product Manager-Client relationship ............................ 47
Figure 5 ~ UML Product Manager ................................................ 48
Figure 6 ~ UML Client................................................................ 50
Figure 7 ~ Database Structure ..................................................... 68
Figure 8 ~ Appeal output............................................................ 86
Figure 9 ~ Compare result............................................................ 98
Figure 10 ~ Compare -Error message ............................................ 99

Index of Tables

Table 1 ~ Crawler example 1......................................................... 30
Table 2 ~ Crawler example 2......................................................... 30
Table 3 ~ Crawler example 3......................................................... 31
Table 4 ~ Crawler example 4......................................................... 31
Table 5 ~ PageRank example......................................................... 34
Table 6 ~ Wordrelations example 1............................................... 36
Index of Code

Code 1 ~ Appeal ........................................................................................................86
Code 2 ~ Compare ....................................................................................................96
Code 3 ~ Crawler ...................................................................................................119
Code 4 ~ PageRank ...............................................................................................126
Code 5 ~ Wordrelations .........................................................................................134
Code 6 ~ Database ..................................................................................................163
Introduction

Destination Monitor plays an important role in the field of the tourism. This paper will present an idea of development of a system, devoted, firstly, to tourism management, and secondarily to visitors’ and clients’ satisfaction. When we talk about tourism management, we are talking about pubs, restaurants, and hotels owner, or everything else that contributes to the improvement of tourism.

Day by day, we can see that tourism is a pillar on which a nation is based on.

The development of the theory we are going to suggest is based on an apprenticeship at CISET\(^1\), tourism department of Ca’ Foscari in Venice, Italy. In this period, we understood the significant idea of the creation of a system that can retrieve each information about the Product Manager’s activity, in both negative and positive ways, and compare the relative Tourism Products to similar others. It is important to underline that

\(^1\) CISET is International Centre of Studies on Tourism Economics
nowadays, there are a lot of systems that take the info and give an account of the total situation in the World Wide Web in similar ways.

The goal of this Master thesis is the definition of a final Appeal of a Tourism Product: nowadays, we can see that there are some services and systems that create lists with the best and the worst existent products; without differentiating them by typology and without considering some external conditions and variables, as well. So, Appeal, is our final result, and we can define a point of view, crossing ten different parameters.

The final Appeal is a value that is the indicator towards to improvement of the product in a determinate component, or it is a comparison between other products with similar in features. In this way, we can define a possible benchmark determining the progress of the Tourism Product.

The idea of this project is explained in three steps, in which we will show different sides of the final system.

- The first one is the data retrieval from the Web, by means of opportune indicators and appropriate algorithms\(^2\);
• The second step consists in the client analysis: what the customers want and how they want it, both online and offline\(^3\);

• The third one is the union of the previous outputs, defining new indicators and a code of the final algorithm, determining the *Appeal*.

The paper is divided into five chapters; each of them precisely describes the steps of the whole project.

1. Chapter one defines the Destination Monitor – what it is and how it is used in real life; it explains the actors co-existing in this dimension and their actions.

Eight real examples of the use of existent systems and their implementation will be presented: it is the starting point that allows a comparison to new our proposal\(^4\).

\(^3\) references [23], [24], [25], [26]

\(^4\) references [5], [6], [7], [8], [9], [27], [28], [29], [30], [31], [32]
2. Chapter two is the first step of the final *Appeal*.

   We will discover “when” and “how” the data are retrieved. We will analyze two famous algorithms, which are specific to this context – some execution examples are shown later in this chapter.

   In addition, we will study the indicators request by *Tourism Products*.

3. In Chapter three we will basically describe the second phase of the *Appeal* definition.

   We will see the Destination Monitor from the client side: in fact, without clients (namely tourists), we do not have any feedback and new potential customers of *Tourism Product*. So, we will analyze what features tourists look for and how they evaluate them.

4. After the previous analyses, in Chapter four we will have the possible solution for the *Appeal* definition. It is the heart of the whole hypothesis; in fact, we will define the indicators that are used in a weighted average, whose result is the *Appeal*. 
The indicators are the results of the study carried out during the training in CISET and what Product Managers want. We will have ten possible metrics of comparison.

5. In the final Chapter five we can theorize some possible optimizations, implementations and future works.

In the paper there are references to existent services and interviews to Product Managers, who helped us in understanding what they wish a system would analyze the final Appeal. The result is taken as a reference of the benchmark in the range of the improvement.
1. **Destination Monitor**

Before entering in the detail of our new system, it is important to define some basic concepts, such as what tourism is and what the requests by both customers and owners are.

When thinking about tourism, what immediately comes to our minds is the activity of touring, especially for pleasure, but there are many other elements that define tourism.

It is important to know and analyze tourism because it is connected to several fields – like economy, society and environment, which all impact on the territory and its development.

Another reason for studying tourism is provided by personal interconnections: we can see how the relationships are modified in a given territory – locally, nationally and internationally, as well the economy and other components.

Tourism is characterized by constant evolution, in fact destination trends rapidly change synchronically and asynchronally.
This thesis focuses on a branch of tourism, namely Destination Monitor\(^5\): we can see that the reputation and how a *Tourism Product* appears with respect to concurrent others are central.

With this respect, we have to define some concepts, like touristic actors in a Destination Monitor. A Destination Monitor is defined as a system that studies the progress of a destination, using indicators.

### 1.1 Actors in Tourism

It is common place that the actors which appear in a touristic environment are tourists and destinations, only.

A tourist is a person who (usually) travels for pleasure, sightseeing and thus staying in hotels; a destination is defined as the predetermined object of a journey.

But, these definitions are given by the fact that, for the majority of people, tourism only means a holiday resource; on

---

\(^5\) references [1], [2], [3], [4]
the contrary, the destination is very often related to one's work place, to one’s health, to family matter, or just to a step of a whole holiday. So, we can define tourism as a combination of *Tourism Producers, Distributors* and *Consumers*.

We will see these elements in detail: the following *Figure 1* shows the aforemenioned actors’ features and their distinctions.

![Figure 1 ~ Tourism](image-url)
We are paying a greater attention to the first actor: *Tourism Producers*. But it is important to get acquainted to the others, as well.

*Distributors* and *Consumer* are not part of the Destination Monitor and *Appeal* does not focus on them; but, in order to get a better view of the target in our case study, we should make an overview of all these components. So, the following subchapters show the categories in detail.

### 1.1.1 Tourist Producers

The first actors present in tourism are *Tourism Producers*: a tourism producer is the destination. Destination is not only the “city” that we visit, but it can also involve other kinds of *Tourism Producers*, which are divided into four typologies.

1. The first one is *Transport*.

   It is interesting to note that the *Transport* is divided into two different categories: the one which allows us to get to
the destination, and the one which is present in the destination.

So, *Transport* can include public transport (plane, train, bus, boat, etc.), or private means of transport, like cars. We can see that this information is important when we define the indicators about the *Appeal* definition.

2. The second typology is named *Holiday Makers*, which consists of pre-packaged travels. Two examples, that are cruise ship and Theme Parks, show what *Holiday Maker* means.

When tourists get involved in this destination typology, they have fun through the attractive which is present in it. Whether they chose a Theme Park, they know that they can get restaurants, shops and rides within the destination. If they chose a cruise ship, they know that their stay is decided and pre-packaged by others.

3. The third kind is featured by hotels or camping sites, that are the *Overnight Stay*. 
When *Overnight Stay* are meant as hotels, they provide tourists with overnight facilities and also kitchen equipments.

4. The fourth and last typology is the *Services In Loco*.

The *Services In Loco* are all the touristic services present in a determinate destination. They include restaurants, shops et similia. *Service In Loco* is sometimes particular and characteristic of the place, while others are chains present everywhere.

It is important to stress that these four typologies are not perfectly subdivided, overlaps are possible. In this thesis we try to divide them as clearly as possible, because typologies help us create the final *Appeal of the Tourism Product*, and we can compare *Tourism Products* basing on the same parameters.

We can see that there are mash ups among the products. An example is provided by *Holiday Makers*: there are some *Services In Loco* which are built especially for the structure; so, we have two different *Tourism Products* to compare in the same site.
In Chapter four we will see how typologies are used.

1.1.2 Distributors

A distributor is an entity that provides tourists with the tools to arrange their travel. Tourists can plan a journey in several ways – relying on different Distributors (agencies in the Web...).

There are three sub-categories.

1. Tour Operators.

They combine components to create a package holiday. They advertise and print brochures to promote their products, holidays and itineraries. The journey, the location and many other elements are decided by a third party.

2. Travel Agencies.

They are slightly different from the first distributors. In fact, we can define a Travel Agency as a retailer that provides with travel and tourism related services to the
public on behalf of suppliers. In this case, the holiday is defined in every small details, starting from the transport, to get to the daily program.

3. *Incoming Agencies.*

Different from the other distributors, in the fact they are located on the incoming destination: they prepare packages in which they define programs and what to visit in the incoming area.

All of these kinds of distributors can be in offline and online.

1.1.3 *Consumers*

Eventually, we make a brief overview of the last actor present in the tourism. A *Consumer* is defined as a person that uses *Tourism Producers* and *Distributors*.

Typically, the *Consumers* are the tourists; but we have to differentiate them into two categories:
1. A tourist is defined as someone leaving from a place to another, for at least twenty-four hours time but not over one year;

2. A hiker is someone visiting a target destination for no more than twenty-four hours, and without staying overnights.

It is important to underline that Tourist is not a worker: he/she spends his/her time for pleasure, during holiday time.
1.2 Pre-existent Destination Monitors

Several data retrieval tools are available, but our case study focused on the Appeal evaluation, given by eight services in eight Italian Regions: in detail, we study how they work and what they want.

The outputs were provided by regional agencies via e-mail exchange.

Those agencies found it very important to create a Destination Monitor service, as it will impact with economy. We make a brief overview of the system used by the aforementioned Italian Regions.

To simplify the concept, we can divide the Regions into two different groups, because some of them has similar attitude.

The first group is composed by Tuscany, Piedmont\textsuperscript{6}, Liguria, Emilia Romagna\textsuperscript{7} and Lombardy\textsuperscript{8}. In this group the Regions are not aware of how the data for defining Destination Monitor are

\textsuperscript{6} references [30]
\textsuperscript{7} references [9], [31]
\textsuperscript{8} reference [32]
obtained: they are supported by real services that analyze Open Data and then the outputs are saved in the Region’s database. The first group did not create or program any services, but used existent services that retrieve information and data from public database, Statistical National Institute and the Web (including social networks). Unfortunately, we do not have algorithms or info on how the data were retrieved. TheRegions all retrieve data from Open Data.

In this work we focus on the second group, composed by Puglia⁹, Trentino¹⁰ and Veneto – with particular attention to Venice¹¹.

Differently from the first group, those Regions all created a service that retrieves data from Open Data and specific indicators to adopt in the analysis, as well. They created a service basing on what the features of tourism are – e.g. tourist flows, customers inquiries, gemorphological diversity...

Now we will study these features, to understand their work and compare them.

---

⁹ references [5], [6], [7], [8], [29]
¹⁰ reference [28]
¹¹ reference [12], [27]
The first system that we explored was created by Puglia Region; it is divided into two different phases.

1. The first step retrieves data by a telematic system: basically, it allows the receiving structure to directly send data to the central server. Every two weeks there is a data storage, so that they are always updated.

2. The second step consists in sending data to ISTAT\textsuperscript{12} to check the statistics. Once ISTAT approves what is sent, the data are saved in the Region’s own database, creating and updating Puglia Open Data.

In Puglia, the Product Managers of various Tourism Products send information and data to the Region server and this information and data are sent to ISTAT to be checked in a second moment.

Now, we analyze the second system. Trentino Marketing is a branch of Trentino Region and works following two phases.

\textsuperscript{12} Istituto Nazionale di Statistica Italia. It is the Italian structure in which each commercial structure sends its statistics.
We do not focus on how the service checks the clients' (namely tourists) feedback. Their concern is mainly the *Overnight Stay*.

The retrieval is carried out by an external service named *Develon*: this project analyzes data from the hotel itself because *Product Managers* themselves insert their statistics. It is important to underline that data are not publicly revealed, as they are considered sensitive information, for this reason they are used only in an anonymous way.

As far as our work is concerned, the interesting part is the feedback retrieval from social networks: Trentino Marketing employs a program (named *TrustYou*) written by a Master's Degree student who proposed his program in his thesis. In this service, there is a crawler which retrieves the most frequent keywords related to tourism world from several social networks and inserts them in a list.

The last service of this group, is named *Venice Project Center*: this dashboard was created specifically for the city of Venice. The project is written by the students attending the Worcester Polytechnic Institute\(^\text{13}\) who also supplied with online

---

\(^{13}\) Private research university in Worcester focusing on the instruction and research of technical arts and applied sciences.
tutorial. *Venice Dashboard* is a web application developed in order to display information about Venice in real time, using individual modules or widgets. Each widget collects publically-available information from existing web sites, using mash up techniques or API’s.

After the data retrieval, the information is saved in the Venice Open Data, which becomes public and usable by other entities.
1.3 Purpose of this thesis

Basing on the aforementioned analyses, we came up with the following conclusion.

Examining the development of the *Appeal* appears to be crucial, but services mash up the result and the outputs, without understanding whether a component is more important or relevant than others. Thus it is unlikely to get homogenous data.

This situation leads to the misleading comparison of structures which have dissimilar features.

In this perspective, in the following chapters we will try to determine the *Appeal*, using indicators and outputs based on algorithms, which are specifically designed for our aim.

The result will have an important impact in the definition of the *Tourism Product* and we will be able to delineate a benchmark to be used by *Product Managers* to compare similar activities.

In the fourth chapter we will detail how the comparison is made.
2. **Service-manager**

**system**

In this chapter we will show the system we employed to retrieve the data in the Web and its relative algorithms.

This system works in backstage, analyzing data in the World Wide Web. The main idea is based on the retrieval, not only of the Open Data\textsuperscript{14}, but of sensitive data and then they are crossed in a weighted mean.

The chapter is divided into two parts: the first subchapter shows how this system works, illustrating the codes employed; the second part presents the indicators to be retrieved.

Retrieving the Open Data is quite easy as there are not copyrighted issues or anyone’s control, so they are free of use.

The final goal of this thesis consists in looking for an *Appeal* of the *Tourism Product*.

---

\textsuperscript{14} They are freely available to everyone to use and republish as they wish.
It is important to remember that the case study is concerned with the Destination Monitor, and we can find several interesting roles and consequences of it. In fact, we can have different results based on the choice of the Destination Monitor.

2.1 Data retrieval

This paragraph shows the data retrieval and the relative algorithms adopted. Figure 2 is a simplified illustration of how we can retrieve the data in the Web.

![Figure 2 ~ Data retrieval](image)
In this schema we can see that Service and Product Manager talk together: the double arrows show that Service writes and reads what the Product Manager does, and vice versa.

Service retrieves the data from social networks or tourism portals, in which customers can leave a feedback or comments: in this way, everyone can say his/her opinion about the Tourism Product.

Figure 3 shows the Product Manager actions.

---

Figure 3 ~ UML schema Product Manager
We analyze the different actions that Product Manager can do:

- **READ**

  *Product Manager* reads the service results;

- **ASK**

  *Product Manager* asks some information about his/her *Tourism Product*.

*System* is the hearth of the whole operation; in fact, it processes the data given by the algorithms and shows the final progress of the activity.

In the following paragraphs, we will see the *Crawler* algorithm and the *PageRank* algorithm\(^\text{15}\), with some examples.

We created the last algorithm, *Worldrelations*, which is a particular algorithm that creates links among the keywords.

\(^{15}\text{reference [10], [13]}\)
2.1.1 Appeal

The Appeal is defined as the power to attract interests, and it determines the choice of a destination. Appeal is the goal of every service or system addressed to Product Manager.

We can define the Appeal into two different categories\(^\text{16}\):

1. The first one consists in the Explicit Appeal, published on different web portals, with the goal of understanding how positive and negative feedback can influence the destination choice;

2. The second one is named Implicit Appeal, that allows Product Manager to evaluate Tourism Product, using data retrieval from statistic study.

Combining the previous parameters, and using a weighted average, we can define the final Appeal.

\(^{16}\) reference [1]
2.1.2 **Crawler**

We see the first algorithm used in the data retrieval. A *Crawler* is a computer program that is capable of performing recursive searches on the World Wide Web.

The *Crawler* systematically crawls pages and looks at the keywords and links within the page, then returns that information to the search engine’s server for indexing.

The code in Python\(^\text{17}\) language will be shown in Appendix A.

In the following *Table 1, Table 2, Table 3* and *Table 4*, we present four executions and relative outputs of the algorithm.

The execution is divided in two steps: the first step is the choice of a web site, the second step is search of the occurrences of some given keywords, in the web site.

The results are shown in the following tables; each of them has three attributes.

---

\(^{17}\) Python is chosen because is similar to pseudo code. It is particular because it is a dynamic programming language.
1. *Web Site*

   It is the first one, it is the web site from which we are able to retrieve the occurrences of the keywords;

2. *Level*

   The *Crawler* can search keywords in different levels of the web graph. In these specific examples we used one or two levels;

3. *Keywords*

   It contains all the keywords with the relative occurrences.
### Web Site

**http://www.legambiente.it/temi/turismo**

<table>
<thead>
<tr>
<th>Level</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keywords</strong></td>
<td></td>
</tr>
<tr>
<td>Roma</td>
<td>444</td>
</tr>
<tr>
<td>storico</td>
<td>51</td>
</tr>
</tbody>
</table>

*Table 1 ~ Crawler example 1*

### Web Site

**https://www.tripadvisor.it/Hotel_Review**-

**Web Site**

`g187849-d229090-Reviews-Hotel_Berna-

**Milan_Lombardy.html**

<table>
<thead>
<tr>
<th>Level</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keywords</strong></td>
<td></td>
</tr>
<tr>
<td>qualita</td>
<td>33</td>
</tr>
<tr>
<td>servizio</td>
<td>287</td>
</tr>
</tbody>
</table>

*Table 2 ~ Crawler example 2*
### Web Site

**https://www.tripadvisor.it**

<table>
<thead>
<tr>
<th>Level</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td></td>
</tr>
<tr>
<td>hotel</td>
<td>128345</td>
</tr>
<tr>
<td>Mestre</td>
<td>463</td>
</tr>
</tbody>
</table>

*Table 3 ~ Crawler example 3*

### Web Site

**http://www.volareweekend.com/it/offerte-voli/capodanno/capodanno-low-cost.html**

<table>
<thead>
<tr>
<th>Level</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td></td>
</tr>
<tr>
<td>volo</td>
<td>407</td>
</tr>
<tr>
<td>notte</td>
<td>210</td>
</tr>
<tr>
<td>citta</td>
<td>17</td>
</tr>
<tr>
<td>divertente</td>
<td>100</td>
</tr>
<tr>
<td>cultura</td>
<td>347</td>
</tr>
<tr>
<td>benessere</td>
<td>198</td>
</tr>
</tbody>
</table>

*Table 4 ~ Crawler example 4*
2.1.3 PageRank

In this paragraph we study the PageRank algorithm: it is used to rank web sites in their search engine results. PageRank works by counting the number and the quality of links to a page to determine a rough estimation of how important the web site is.

The underlying assumption is that more important web sites are likely to receive more links from other web sites. Actually, this algorithm was created by Google\textsuperscript{18}.

Its work is easy: it exploits incoming links from popular pages to raise the rank of the pages themselves. We get the position of the page, and we know how much that topic is quoted. This algorithm helps understand what favorite attributes are when clients are looking for a destination.

The PageRank algorithm sees the Web as a directed graph with the pages being nodes and hyperlinks being connections between those nodes. It can be used to rank the nodes of any kind of graphs (including undirected ones) by importance.

This description uses graph terminology and only shows how it is done for a directed graph such as the web graph.

\textsuperscript{18}The PageRank citation ranking: bringing order to the Web, 1998
While it is accurate to say that PageRank will tell us the importance of each page, a more accurate definition is that PageRank assigns a probability to each page. Specifically, the PageRank value of a page is the probability, between 0 and 1, that someone, surfing the page by clicking links randomly, will end up on that page.

With PageRank we have a measure of the rank prestige: it forms the basis of most web page link analysis algorithms.

In Appendix B we will present the PageRank code in C\textsuperscript{19} language.

The algorithm has a file in txt format as input in which there are two columns: each of the element of the first column is the starting point and the second column represents the arriving node.

The output is the a vector which presents the prestige of each link.

\textsuperscript{19}C is a structured and procedural programming language that has been widely used for both operating systems and applications.
The previous Table 5 shows an example of PageRank with a graph composed by 100 nodes as input.
2.1.4 Wordrelations

In this paragraph we present an algorithm that is a combination of the previous algorithms. From a given keyword provided by Google API\textsuperscript{20}, we search if there is a link to other keywords from the same set.

In Appendix C we will present the code written in Python language.

The results are shown in the following tables, through two parameters:

1. The first one is *Keyword*, that is the initial keyword we start searching;

2. The second one is *Wordrelations* and it is the results of the algorithm. We see the occurrences of the words starting from a keyword.

---

\textsuperscript{20} Google APIs are sets of application programming interfaces developed by Google which allow communication with Google Services and their integration to other services.
### Table 6 ~ Wordrelations example 1

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Capodanno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordrelations</td>
<td>Montagna</td>
</tr>
<tr>
<td></td>
<td>Spiaggia</td>
</tr>
</tbody>
</table>

### Table 7 ~ Wordrelations example 2

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Ferie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wordrelations</td>
<td>Montagna</td>
</tr>
<tr>
<td></td>
<td>Campeggio</td>
</tr>
<tr>
<td></td>
<td>Mare</td>
</tr>
</tbody>
</table>
### Table 8 ~ Wordrelations example 3

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Hotel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colazione</td>
<td>1</td>
</tr>
<tr>
<td>Spa</td>
<td>3397</td>
</tr>
<tr>
<td>Parcheggio</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 9 ~ Wordrelations example 4

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Vacanza</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roma</td>
<td>7</td>
</tr>
<tr>
<td>Venezia</td>
<td>1</td>
</tr>
<tr>
<td>Trieste</td>
<td>0</td>
</tr>
</tbody>
</table>
2.2 Indicators definition

According to CISET standard\textsuperscript{21}, we define now five indicators, that are used to determine the *Appeal of a Tourism Product*.

We analyze the definition of these indicators. They are presented through *Table 10, Table 11, Table 12, Table 13* and *Table 14* in which there are three cells.

I. NAME

The first cell shows the name of the indicator. The name is the identification ID for each indicator.

II. DEFINITION

The second attribute is the definition of the indicator. The definition helps us understand what it is and what its goal during the data retrieval is.

\textsuperscript{21} references [1], [2], [3], [4]
III. IMPACT

The third cell is particular because it explains what the impact in the Appeal search is: it takes the information from the client’s sides and the Product Manager’s side. We can see how the Appeal can change in front of determinate characteristics.

It is important to remember that these indicators are adopted to evaluate the Destination Monitor.
**NAME**

**Brand**

**DEFINITION**

Particular product or a characteristic that serves to identify a particular product. Using this term we can immediately think of the *Tourism Product*, and so we can distinguish it through the concurrency.

**IMPACT**

This indicator is important to guarantee a certain confidentiality with customers. The client knows the product and, to some extent, acts as he/she knows every part of the product. The role of this indicator consists in improving the brand, so every client knows it, as a friendly brand. If the client knows the brand, it is more likely that the client sponsors it to friends.

*Table 10 ~ Indicator – BRAND*
**NAME** | Products  
---|---  
**DEFINITION** | A set of tangible or intangible attributes of a service; it is usually obtained by a production process or a creation from initial resources with the goal of improving the final value.  
**IMPACT** | The impact of this indicator gives an account of the Tourism Product activity trend.  
From the Product Manager’s side it is important to understand the impact because it shows what clients ask, and what to change or to improve.  
From the client’s sides we can see that tourists choose a product with respect to another.

*Table 11 ~ Indicator ~ PRODUCTS*
**NAME**  
Performance compared to the same type destinations

| **DEFINITION** | Total progress of the activity, including the economic and, sometimes, political aspects. The results give an overview of the progress, with respect to other structures present in the same destination. The indicator consists in a comparison between similar structures in the same destination. |
| **IMPACT** | The main impact of this indicator is the concurrency control: in fact, a Product Manager can control the progress of the competitors at any time, so he/she can improve himself/herself. The clients have the possibility to choose the Tourism Product comparing it to others, in the same destination. |

Table 12 ~ Indicator - PERFORMANCE COMPARED TO THE SAME TYPE DESTINATION
### NAME

**Progress of investments and dynamic of the offer, expenses and use of attractors**

<table>
<thead>
<tr>
<th>DEFINITION</th>
<th>IMPACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progress of the activity using the clients’ feedback.</td>
<td>The indicator is related to the Product Manager, since it shows the progress of the Tourism Product using the feedback and the comments of customers. Using this indicator the Product Manager can immediately understand what clients do, and, can modify the features of the Tourism Product to attract clients.</td>
</tr>
<tr>
<td>What clients want and how they want it.</td>
<td></td>
</tr>
</tbody>
</table>

*Table 13 ~ Indicator - PROGRESS OF INVESTIMENTS AND DYNAMIC OF THE OFFER, EXPENSES AND USE OF ATTRACTORS*
**NAME**  Accessibility

**DEFINITION**  Possibility to be accessible to other people.

This indicator has two different impacts.

1. The first performance is on the accessibility of the web site, and it is addressed to the *Product Manager*: it makes the *Product Manager* himself/herself understand if the web site is accessible and user-friendly.

2. The second performance is on the physical accessibility of the *Tourism Product*. It is addressed to the clients, since it involves some parameters that identify whether the destination is easy to reach – e.g. considering public transport.

*Table 14 ~ Indicator – ACCESSIBILITY*
3. **Manager-client system**

In this chapter we will present the system which links product *Product Managers* and *Customers*. In this system we see that *Product Managers* and clients can speak together, through message exchanges.

In a social network everybody can leave a comment or an opinion about his/her life, or in this specific case, about a *Tourism Product*.

Basically, the idea consists in the creation of a social network in which the *Product Manager* can describe in detail his/her activity, and the client can read and, if he/she wants, leaves a comment.

The chapter is divided into three sections: the first one is the behaviour of the system, analyzing the clients’ and *Product Managers*’ actions; the second one is the clients’ requests. The last section is concerned about the keywords which are used in
the final part of the project: in fact, knowing the clients’ requests, we can get the weighted average and define the final Appeal.

3.1 **Product Manager-Client**

**relationship**

The technical job of the *Product Manager-Client* system involves the actors who exchange messages

*Figure 4* simplifies the behaviour of the *Product Manager-Client* system.

It is interesting to note that in this system each *Product Manager* can describe his/her activity product, so the *Product Manager* is also the *Product Manager* of his/her web site.

---

22 references [15], [16], [17], [18], [19], [20], [21], [22]
Between the Product Manager and the System there are double arrows because the Product Manager asks, answers and writes the page, while Clients can read and write (annotate, estimate, book) on the page, but the Client can not perform the ask action.

Figure 4 ~ Product Manager-Client relationship
Product Manager has three possible actions:

- **READ**

  The first action allows the Product Manager to read clients’ feedback, so he/she can have the idea of what clients require from his/her Tourism Product;
- **WRITE**

  *Product Manager* describes his/her activity, and upgrades the web site with news and advisement, he/she answers clients’ requests, as well he/she maintains a strict relationship with the client;

- **ASK**

  In this action, *Product Manager* interrogates the system, e.g. asking what is the percentage of a certain keyword.
The client has four actions:

- **READ**

  The *Client* reads the description and the *Product Manager*'s answers;
- **ANNOTATE**

  The *Client* writes his/her positive or negative opinions in the page, and asks the *Product Manager* some information;

- **ESTIMATE**

  This action is related to the evaluation of the activity, basing on several criteria. The activity is assigned one to five points;

- **BOOK**

  The last action allows the client to book a *Tourism Product*. 


3.2 User-friendly requests

During the training at CISET, we collected information from different Product Managers about what their expectations were, with respect to the system we are proposing.

As far the Clients’ requests are concerned, they are the results of the Wordrelations algorithm.

3.2.1 View from Product Manager’s side

We interviewed four Product Managers: they work in two different environments (Overnight Stay and Service In Loco), but each of them showed common requests for a monitoring system.

The first interview was to a Restaurant Manager: in this context we can see that the restaurant is in a small town, quite far from the city center; so the first problem is where it is located. This Tourism Product has, however, some clients,
because its main strength is the strict relationship between the
Product Manager and Customers: friendship allows a big word-
of-mouth with other possible clients.

The word-of-mouth is the biggest strength also because
this Tourism Product has its interest in tradition: within the last
thirty years, the restaurant has a little upgraded, maintaining
the focus on the food quality and the territory.

The second interview was to a bathhouse: the beach is a
strong attraction during the summer, so there are many
customers every year, especially during the weekends. It is
interesting that the Product Manager keeps a relationship with
the loyal customers even during winter time via emails, letters
and social networks.

The third interview was to a young adult who manages a
summerhouse near the beach, a few kilometers far from the city
center (Venice). In this interview, the territory was mentioned
several times: he thinks that tourists are attracted by the
touristic city which is a few kilometers far from the Tourism
Product. As well as the second Product Manager which was
interviewed, this Product Manager emphasizes the strict
friendship with clients, and he also wants to maintain the relation during the winter time, when his activity is closed.

In order to allow a bigger and new clientele, the Product Manager wishes that news and improvements would be advertised.

The last interview was to a man who worked for a renowned hotel chain in Italy. He has been working for a long time, he has discovered that Internet is a new source of clients as inside web sites it is possible to show the hotel’s improvements and news.

However, the word-of-mouth is still a strong way of advertising in a Tourism Product: if a customer is happy, then the reputation is good, hence other clients are attracted, otherwise fewer and fewer customers are likely to book. It is interesting to note that in this context, the friendship between the client and Tourism Product is a key point. The focus for this Product Manager is on the touristic city center.
3.2.1.1 Keywords

From the previous four interviews, it is interesting to note there are seven keywords that summarize what Product Managers think is the best way to maintain and increase the Appeal of their Tourism Product.

1. FRIENDSHIP

It is a little odd that friendship is the first and most popular keyword, as it is present in all the interviews. It is important to maintain a close bound with the clientele, so that customers will be likely to return in future.

2. INNOVATION

Although, innovation is not always a good idea, as changing the brand could increase the risks of losing customers, three of the interviewed Product Managers think that it is one of the strenghts of a Tourism Product: innovation shows the continuously wish of modernity.
3. **SOCIAL NETWORK**

Today, a vast part of the population owns a social network account, *Tourism Products* must be online, so that they can be almost instantly connected to their clients.

The *Product Manager* can illustrate his/her activity, describing it and answering any questions via social network.

4. **WORD-OF-MOUTH**

It is the oldest form of advertisement, born in the ancient times. If someone is satisfied, his/her friends or acquaintances are likely to be informed; and vice versa, they are likely to be advised not to book the structure.

Nowadays, it is done immediately through a *tweet* or a *like*.

5. **DESTINATION**

The importance of the destination was stressed by all the *Tourism Products*. Actually, tourists book their holidays basing on the popularity of the territory, choosing hotels,
restaurants, Theme Parks and so the choice is directly related to the destination.

6. **WEB SITE UPGRADES**

Customers choose a *Tourism Product* also relying on the quality of the web site, in terms of usability and accessibility.

The code must be upgraded to new technology, as nowadays the majority of customers own a smartphone and tend to surf the Internet using it, so every device must read the information in order to find the closing days, opening hours, prices, facilities...

7. **TRADITION**

More than one time, tradition was mentioned, above all when the territory as well as the *Tourism Product*, are able to offer typical or folk products – when a tourist goes to the beach, he/she wants sunglasses, umbrellas and deckchair; when a tourist visits Alps, he/she wants to eat mushroom.
3.2.2 View from the Client’s side

As we mentioned before, we have created an algorithm enabling to scan the web site in order to search for the occurrences of given keywords. We will present a potential example of clients’ requests using keywords.

In this experiment the keyword employed is “Christmas”.

The experiment is divided in two phases:

1. In the first phase, we ran Wordrelations inside Google APIs stopping at the first level of crawling;

2. In the second phase, we ran an adaptation of Crawler in order to find word occurrences in a given web site.

Since Crawler requires two parameters – a given keyword and a URL – we input the potential keyword related to tourism, “Christmas”, and the URL (the American web site https://www.timeanddate.com/holidays/us/christmas-day) related to “Christmas”, in order to find the occurrences of the given keyword.
The word “Christmas” occurred 128 times in Crawler.

This example shows the possible utilization of the algorithm: starting from given keywords we can create a list of the most frequent, save the results and create links among the given keywords.

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Wordrelations</th>
<th>Crawler</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2017</td>
<td>7</td>
<td>674</td>
</tr>
<tr>
<td>2 america</td>
<td>21</td>
<td>50</td>
</tr>
<tr>
<td>3 beach</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4 california</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5 car</td>
<td>311</td>
<td>606</td>
</tr>
<tr>
<td>6 cruise</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7 day</td>
<td>1314</td>
<td>12193</td>
</tr>
<tr>
<td>8 family</td>
<td>96</td>
<td>152</td>
</tr>
<tr>
<td>9 flight</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>10 food</td>
<td>51</td>
<td>18</td>
</tr>
<tr>
<td>11 friends</td>
<td>24</td>
<td>119</td>
</tr>
<tr>
<td>12 holiday</td>
<td>579</td>
<td>4424</td>
</tr>
<tr>
<td>13 italy</td>
<td>2</td>
<td>286</td>
</tr>
<tr>
<td>Keywords</td>
<td>Wordrelations</td>
<td>Crawler</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>journey</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>love</td>
<td>84</td>
<td>105</td>
</tr>
<tr>
<td>mountain</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>plane</td>
<td>2</td>
<td>75</td>
</tr>
<tr>
<td>santa</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>sharm</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ship</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>ski</td>
<td>54</td>
<td>50</td>
</tr>
<tr>
<td>snow</td>
<td>27</td>
<td>6</td>
</tr>
<tr>
<td>travel</td>
<td>36</td>
<td>204</td>
</tr>
<tr>
<td>tree</td>
<td>214</td>
<td>20</td>
</tr>
<tr>
<td>trento</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>usa</td>
<td>59</td>
<td>393</td>
</tr>
</tbody>
</table>

*Table 15 – Client side example*
4. **A new proposal**

In this chapter, we present the indicators and the parameters of the weighted average of the *Appeal*.

The chapter is divided into three parts:

1. In the first we study the indicators adopted to analyze the final *Appeal*;

2. In the second part we show the structure of the database which is used in the project;

3. The third part describes the algorithm that is used to create the weighted average of the *Appeal*.

The three parts describe our program to be used for data retrieval and *Appeal* definition.

We are also able to use the project as a benchmark structure: in fact, we can compare two *Tourism Products* with similar
features. The algorithm shows where a Tourism Product is better performing than the other one.

4.1 Indicators definition

We will analyze the indicators of the Appeal.

We found these ten indicators during the training in CISET department: they are the results obtained from the four interviews and the eight pre-existent services.

4.1.1 City

It is an important indicator because it considers the site in which the Tourism Product is located. The results are different whether we deal with a city or a town.

Famous museums of European capitals – British Museum in London, or Museo degli Uffizi in Florence – have a greater number of visitors every day than a small town in the suburbs.
In the algorithm that defines the *Appeal*, we use four distinctions.

<table>
<thead>
<tr>
<th>Dimension of City</th>
<th>City value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capital City</td>
<td>50</td>
</tr>
<tr>
<td>Chief</td>
<td>30</td>
</tr>
<tr>
<td>Town</td>
<td>15</td>
</tr>
<tr>
<td>Village</td>
<td>5</td>
</tr>
</tbody>
</table>

*Table 16 ~ City division*

### 4.1.2 Typology

In Chapter one we analyzed different kinds of typology: these distinctions are included in our database when we specify the *Tourism Product*.

When we compare two *Tourism Products*, the algorithm checks if the *Typology* of both products are the same. We can not compare two *Tourism Products* that are not of the same *Typology* – e.g. we do not compare a *Service in Loco* with a *Holiday Maker*.
When we try to compare two different Typologies of two Tourism Products, the algorithm breaks showing a message error.

4.1.3 Visitors number

To get an idea of the Tourism Product progress, we have to focus on the number of visitors.

This indicator is important when we have to evaluate some features of the tourism improvement. It is possible to retrieve it from the official statistics and Open Data.

4.1.4 Period

The period shows the time in which the Tourism Product is open.

In the algorithm, Period corresponds to the months in which the Tourism Product is open – e.g. if a Tourism Product is
open to tourists four months a year, the \textit{Period} value is equal to 4.

4.1.5 \textit{Evaluation number}

\textit{Evaluation number} is the number of the reviews of the \textit{Tourism Product} which we are analyzing.

In the algorithm, this indicator is not used alone, but in relation with the \textit{Evaluation quality}.

4.1.6 \textit{Evaluation quality}

\textit{Evaluation quality} is strictly linked to the previous indicator.

In the algorithm, this indicator is the arithmetic mean of all the evaluations.

In the \textit{Table 17} we show the five possible evaluation criteria.
### Significance

<table>
<thead>
<tr>
<th>Significance</th>
<th>Evaluation value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very bad</td>
<td>1</td>
</tr>
<tr>
<td>Low</td>
<td>2</td>
</tr>
<tr>
<td>Sufficient</td>
<td>3</td>
</tr>
<tr>
<td>Good</td>
<td>4</td>
</tr>
<tr>
<td>Excellent</td>
<td>5</td>
</tr>
</tbody>
</table>

*Table 17 ~ Evaluation value*

### 4.1.7 Area

*Area* is the surface area of the *Tourism Product*. The value is given by Open Data or statistic information.

In the algorithm, this indicator is used to calculate the *Capacity*.

---

23 *Capacity* is one of the eight parameters of the algorithm. The eight parameters will be described in paragraph 4.3.
4.1.8 Cost

The indicator shows the total costs of the Tourism Product – e.g. tickets and food prices.

Cost is the arithmetic average of the Tourism Product prices.

4.1.9 How to arrive

The indicator shows the transport available and used to reach the Tourism Product.

If the Tourism Product is Transport typology, the indicator is False. Otherwise, the indicator is True and shows the frequencies of the different public transports.

4.1.10 PageRank Results

Using the PageRank algorithm, we get the rank of the Tourism Product.
In the algorithm, *PageRank* Results is a number that corresponds to the percentage of frequency in the World Wide Web.

### 4.2 Database structure

The system we have created needs a database. In the database we save the *Tourism Products*, the *Product Managers* and the *Clients* that are logged.

*Figure 7* shows the relational model of the database.
The database is composed of twelve classes - we will study each class in detail.

4.2.1 Person

The class specifies the Person: it is a superclass, with determined attributes. In Person we find the Product Manager and the Client: they are saved when we they log in the system.

**ATTRIBUTES**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String type. It is univocal for each person logged in. There is a check if the name does not exist, if not, the program asks for a new name.</td>
</tr>
<tr>
<td>email</td>
<td>String type. The personal email is used to send and receive information.</td>
</tr>
</tbody>
</table>
**ATTRIBUTES**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>String type. When there is a login, the password is sent to the program with the hashed password (SHA256 with a salt): in this way we have an higher security control(^\text{24}).</td>
</tr>
<tr>
<td>birthday</td>
<td>Date type. It is built in the following schema: mm/dd/yyyy where mm means month, dd day and the last one is the year.</td>
</tr>
</tbody>
</table>

\(^\text{24}\) In order to avoid that a password could be sniffed while flowing an unprotected network, the client sends a hashed password, the server applies the salt and hashes the password again. Then the server checks whether the password is the same as the one saved in the database [14].

*Table 18 ~ Person class*
4.2.2 Client

It is a subclass that extends the Person class.

Clients who sign up into the service are saved in Client class.

4.2.3 Manager

This is the other subclass that extends Person class.

*Product Managers* are saved in Manager class.
### 4.2.4 Review

In *Review* there are all the reviews of the relative *Tourism Product*.

**Attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>idreview</em></td>
<td>Integer type. It defines the <em>Review</em>. Each <em>Review</em> has an univoque ID.</td>
</tr>
<tr>
<td><em>name</em></td>
<td>Client type. It is a reference to <em>Client</em> class, because every review has a client.</td>
</tr>
<tr>
<td><em>value</em></td>
<td>Integer type. It can change from one to five.</td>
</tr>
<tr>
<td><em>description</em></td>
<td>Text type. It is the text in which there is the review.</td>
</tr>
</tbody>
</table>

*Table 19 ~ Review class*
4.2.5 Tourism Product

This class is complex because it contains the higher number of indicators used in the weighted average. The class defines the Tourism Product that we are analyzing.

Product Manager fills an online form which comprehends the following attributes.

**ATTRIBUTES**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>id</strong></td>
<td>Integer value. It defines the Tourism Product. Each Tourism Product has an univoque ID.</td>
</tr>
<tr>
<td><strong>nametp</strong></td>
<td>String type. Each touristic activity has a name: to avoid that there could be possible conflicts with homonym, the server checks whether the name has not been already saved.</td>
</tr>
</tbody>
</table>
### ATTRIBUTES

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Manager type.</td>
</tr>
<tr>
<td></td>
<td>It is the Product Manager.</td>
</tr>
<tr>
<td>visitorum</td>
<td>Visitor presences. This value can change during the activity opening.</td>
</tr>
<tr>
<td>birthday</td>
<td>Date type.</td>
</tr>
<tr>
<td></td>
<td>When the Tourism Product has been started.</td>
</tr>
<tr>
<td>city</td>
<td>City type.</td>
</tr>
<tr>
<td></td>
<td>The name of the city in which the activity is located in.</td>
</tr>
<tr>
<td>typology</td>
<td>Typology type.</td>
</tr>
<tr>
<td></td>
<td>What kind of typology.</td>
</tr>
<tr>
<td>area</td>
<td>Integer type.</td>
</tr>
<tr>
<td></td>
<td>It defines the area of the Tourism Product (m²).</td>
</tr>
</tbody>
</table>
## Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>period</strong></td>
<td>Integer</td>
<td>It defines the opening period. The number indicates the months (range 1 and 12).</td>
</tr>
<tr>
<td><strong>path</strong></td>
<td>Boolean</td>
<td>The value verifies whether the <em>Tourism Product</em> can be reached via public transport. If it is TRUE the table named <em>How To Arrive</em> is created and contains the attribute <em>frequency</em> and <em>howtoarrive</em>. In contrast, if the Boolean is FALSE, the relative table is NULL.</td>
</tr>
<tr>
<td><strong>pagerank</strong></td>
<td>Integer</td>
<td>It is the result of the <em>PageRank</em> algorithm. The result belongs to a range 0-100.</td>
</tr>
<tr>
<td><strong>prices</strong></td>
<td>Integer</td>
<td>It is the arithmetic mean of all the prices of the <em>Tourism Product</em>.</td>
</tr>
</tbody>
</table>
**ATTRIBUTES**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>quality</code></td>
<td>Integer type. This attribute is the arithmetic average of all the Evaluation quality of the Tourism Product. The value belongs to the range 1-5.</td>
</tr>
<tr>
<td><code>howotoarrive</code></td>
<td>How To Arrive type. It refers to the class How To Arrive.</td>
</tr>
<tr>
<td><code>numreview</code></td>
<td>Integer type. It is the total number of the reviews.</td>
</tr>
</tbody>
</table>

*Table 20 ~ Tourism Product class*

### 4.2.6 How To Arrive

This class indicates whether it is possible to reach the Tourism Product via public transport.
**ATTRIBUTES**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
</table>
| Frequency | Integer type. The frequency is the result from Open Data and it shows the timetables of each public transport which allows to get to the *Tourism Product*. Three different ranges:  
1. First range $\rightarrow$ from 1 min to 1 hour time, $frequency = 3$;  
2. Second range $\rightarrow$ from 1 hour to 3 means per day, $frequency = 2$;  
3. Third range $\rightarrow$ over 3 public means per day, $frequency = 1$. |
| HowToArrive | Integer type. The attribute shows the transports available:  
1. PLANE, $howToArrive = 2$;  
2. TRAIN or BOAT, $howToArrive = 3$. |

*Table 21 ~ How To Arrive class*
4.2.7 City

The class indicates the city in which the Tourism Product is located. The Tourism Product is generally located within a City which has several other Tourism Product.

**ATTRIBUTES**

<table>
<thead>
<tr>
<th>attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimension</td>
<td>Integer type. It indicates the type of the City. It is classified into four values (see paragraph 4.2.2).</td>
</tr>
<tr>
<td>namect</td>
<td>String type. It is the name of the City.</td>
</tr>
<tr>
<td>cisnum</td>
<td>Integer type. It is the number of citizens.</td>
</tr>
<tr>
<td>visnum</td>
<td>Integer type. It is the number of the visitors.</td>
</tr>
</tbody>
</table>

*Table 22 ~ City class*
4.2.8 Typology

It is a superclass. It indicates the Typology of the Tourism Product through four subclasses: Holiday Maker, Overnight Stay, Service In Loco, Transport.

4.2.9 Holiday Maker

It is the first subclass of Typology.

In Holiday Maker we insert the Tourism Product that corresponds to this definition.

4.2.10 Overnight Stay

It is the second subclass of Typology.

In Overnight Stay we insert the Tourism Product that corresponds to this definition.
4.2.11 Service In Loco

It is the third subclass of Typology.

In Service In Loco we insert the Tourism Product that corresponds to this definition.

4.2.12 Transport

It is the fourth subclass of Typology.

In Transport we insert the Tourism Product that corresponds to this definition.
4.3 The Appeal definition

In order to obtain the Appeal weighted average, we insert eight variables (here indicated with capital letters).

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 TYPOLOGY</td>
<td>TYPOLOGY indicates the typology of the Tourism Product.</td>
</tr>
<tr>
<td></td>
<td>When comparing two Tourism Products of different typologies, the program breaks immediately.</td>
</tr>
<tr>
<td></td>
<td>It is used to check the comparison.</td>
</tr>
<tr>
<td>2 CITY</td>
<td>It indicates the city in which the Tourism Product is located in.</td>
</tr>
<tr>
<td>3 PERIOD</td>
<td>It indicates the opening months.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| PRICES | PRICES has different values, related to the *Typology*:

- If the *Tourism Product* is a *Service In Loco*,
  
  PRICES is equal to

  \[
  \text{prices of the Tourism Product} \div \text{PERIOD}
  \]

- If the *Tourism Product* is a *Holiday Maker*,
  
  PRICES is equal to

  \[
  \frac{\text{Area}}{\text{prices of the Tourism Product}} \div \text{PERIOD}
  \]

- If the *Tourism Product* is an *Overnight Stay*,
  
  PRICES is equal to

  \[
  \text{prices of the Tourism Product} \div \text{PERIOD}
  \]

- If the *Tourism Product* is a *Transport*,
  
  PRICES is equal to

  \[
  \frac{\text{prices of the Tourism Product}}{\text{visitors number + number of the citizen}}
  \]
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><strong>CAPACITY</strong>&lt;br&gt;The maximum amount of people that the <em>Tourism Product</em> can contain. CAPACITY is given by:&lt;br&gt;$\frac{\text{Area}}{\text{Visitor Number}}$</td>
</tr>
<tr>
<td>6</td>
<td><strong>EVALUATION</strong>&lt;br&gt;EVALUATION is given by:&lt;br&gt;$\frac{\text{Evaluation Number}}{\text{Arithmetic average of quality}}$</td>
</tr>
<tr>
<td>7</td>
<td><strong>WAY</strong>&lt;br&gt;WAY is the results of the sum of the frequency and the transport used to reach the <em>Tourism Product</em>. It is given by:&lt;br&gt;$\text{frequency} + \text{howtoarrive}$</td>
</tr>
<tr>
<td>8</td>
<td><strong>PAGERANK</strong>&lt;br&gt;PAGERANK is the result of the <em>PageRank</em> algorithm</td>
</tr>
</tbody>
</table>

*Table 23~ Appeal definition*

The weighted average is given by:

$$\frac{\text{CITY} + \text{PERIOD} + \text{PRICES} + \text{CAPACITY} + \text{EVALUATION} + \text{WAY} + \text{PAGERANK}}{7}$$

*Table 24~ Weighted Average of Appeal*
In *Code 1* we show the portion of the code that quantifies the Appeal of a *Tourism Product*.

```python
# Appeal returns an int
# Appeal is the value of the Tourism Product
def Appeal(tourism):

    # Parameters used in the weighted average
    # CITY
c = tourism.getcity().getbig()
    # CAPACITY
    s = tourism.getsurface()
    v = tourism.getvisitornum()
    capacity = s/v
    # EVALUATION
    e = tourism.getreview()
    q = tourism.getquality()
evaluation = e/q

    # PAGERANK
    p = tourism.getpagerank()
    # WAY
    w = tourism.gethowtoarrive()
    way = 0

    # Check if the Tourism Product has the parameter
    if w == False:
        way
    elif w == True:
        # WAY
        w1 = tourism.gethowtoarrive().getfrequency()
        w2 = tourism.gethowtoarrive().gethowtoarrive()
```
```python
way = w1 + w2

# PRICES
pr = tourism.getprices()

# PERIOD
pe = tourism.getperiod()

# We differentiate the different typologies
# because the prices change
if tourism.gettypology() == Serviceinloco:
    prices = pr / pe
elif tourism.gettypology() == Holidaymaker:
    prices = (su / pr) / pe
elif tourism.gettypology() == Overnightstay:
    prices = pr / pe
else:
    prices = pr / (tourism.getvisitornum() +
                  tourism.getcity().getcinum())

# We take seven parameters to have the final Appeal
# “ct” capital/chief/town/village
# “pe” is the opening months
# “prices” is the arithmetic average of the prices
# “capacity” is the relationship between
# “surface area” and “number of visitors”
# “evaluation” is the relationship between
# “number of evaluation” and
# “average of the quality”
# “way” is the result
# of “frequency” and “transport”
# “pa” is the result of PageRank
appeal = (ct + pe + prices +
          prices / pe)
```
The output is the following.

![Figure 8 - Appeal output](image)

In the Appendix D we present the whole code with the database and three potential examples of *Tourism Product*.
4.3.1 Comparison of two potential

Tourism Products

We use the program as a benchmark that compares two different Tourism Products of the same Typology.

The Product Manager can have the results of the progress of his/her Tourism Product and the concurrency's: in this way, he/she has the possibility to improve what the Tourism Product is lacking with respect the other Tourism Product.

The program compares each component of the Appeal weighted mean and shows where a Tourism Product is better than the other one.

The features are the same of the Appeal definition.

The Code 2 shows the portion of code that compares two Tourism Products.
We compare two Tourism Products with the same Typology

def Compare(t1, t2):
    Appeal(t1)
    Appeal(t2)

    # Parameters of the first TP
    # CITY t1
    ct1 = t1.getcity().getbig()
    # PERIOD t1
    pe1 = t1.getperiod()
    # CAPACITY t1
    su1 = t1.getsurface()
    vs1 = t1.getvisitornum()
    capacity1 = su1 / vs1

    # PRICES t1
    pr1 = t1.getprices()

    # We differentiate the different typologies,
    # first Tourism Product
    if t1.gettypology() == Serviceinloco:
        prices1 = pr1 / pe1
    elif t1.gettypology() == Holidaymaker:
        prices1 = (su1 / pr1) / pe1
    elif t1.gettypology() == Overnightstay:
        prices1 = pr1 / pe1
    else:
        prices1 = pr1 / (t1.getvisitornum() + t1.getcity().getcinum())

    # EVALUATION t1
    ev1 = t1.getreview()
qu1 = t1.getquality()
evaluation1 = evl/qu1

# WAY t1
wyl = t1.gethowtoarrive()

# Check if the first Tourism Product
# has the parameter
way1 = 0
if wyl == False:
    way1
elif wyl == True:
    wlt1 = t1.gethowtoarrive().getfrequency()
w2t1 = t1.gethowtoarrive().gethowtoarrive()
    way1 = wlt1 + w2t1

# PAGERANK t1
pal = t1.getpagerank()

# Parameters of the second TP
# CITY t2
ct2 = t2.getcity().getbig()

# PERIOD t2
pe2 = t2.getperiod()

# CAPACITY t1
su2 = t2.getsurface()
vs2 = t2.getvisitornum()
capacity2 = su2/vs2

# PRICES t2
pr2 = t2.getprices()

# We differentiate the different typologies,
# second Tourism Product
if t2.gettypology() == Serviceinloco:
    prices2 = pr2 / pe2

Giulia Schiavon 826788
elif t2.gettypology() == Holidaymaker:
    prices2 = (su2 / pr2) / pe2
elif t2.gettypology() == Overnightstay:
    prices2 = pr2 / pe2
else:
    prices2 = pr2 / (t2.getvisitornum() + t2.getcity().getcinum())

# EVALUATION t2
ev2 = t2.getreview()
qu2 = t2.getquality()
evaluation2 = ev2/qu2

# WAY t2
wy2 = t2.gethowtoarrive()

# Check if the second Tourism Product
# has the parameter
way2 = 0
if wy2 == False:
    way2
elif wy2 == True:
    w1t2 = t2.gethowtoarrive().getfrequency()
w2t2 = t2.gethowtoarrive().gethowtoarrive()
way2 = w1t2 + w2t2

# PAGERANK t2
pa2 = t2.getpagerank()

# We compare the parameters of both TPs
while True:
    # First control,
```python
# if the TPs are different,
# we can not compare them
if t1.gettypology() != t2.gettypology():
    print("We can't compare them!")
    print('
')
    break

# We compare the city
# where the TPs are located in
if ct1 > ct2:
    print("CITY")
    print("The City "+t1.getnametp() + " is bigger")

print('
')
elif ct2 > ct1:
    print("CITY")
    print("The City "+t2.getnametp() + " is bigger")

print('
')
elif ct1 == ct2:
    print("CITY")
    print("The cities are big both")

print('
')
```
# We compare the period of opening
if pe1 > pe2:
    print("PERIOD")
    print("The opening months of ")
    + t1.getnametp() +
    " are bigger then ")
    + t2.getnametp()
    )
    print('\n')
elif pe2 > pe1:
    print("PERIOD")
    print("The opening months of ")
    + t2.getnametp() +
    " are bigger then ")
    + t1.getnametp()
    )
    print('\n')
else:
    print("PERIOD")
    print("The opening months ")
    "of the TPs are equal"
    )
    print('\n')

# We compare the arithmetic
# averages of the prices
if prices1 > prices2:
print("PRICES")
print(t1.getnametp() +
    " is more expensive than "+ t2.getnametp())
print('
')
elif prices2 > prices1:
    print("PRICES")
    print(t2.getnametp() +
        " is more expensive than "+ t1.getnametp())
    print('
')
else:
    print("PRICES")
    print("The prices are equal")
    print('
')

# We compare the capacities
if capacity1 > capacity2:
    print("CAPACITY")
    print("The capacity of "+ t1.getnametp() +
        " is bigger than "+ t2.getnametp())
    print('
')
print('
')

elif capacity2 > capacity1:
    print("CAPACITY")
    print("The capacity of "
           + t2.getnametp() + " is bigger than " + t1.getnametp()"
    )

print('
')

else:
    print("CAPACITY")
    print("The capacities are equal"
    )

    print('
')

# We compare the evaluations
if evaluation1 > evaluation2:
    print("EVALUATION")
    print("The evaluation of "
           + t1.getnametp() + " is bigger than " + t2.getnametp()
    )

    print('
')

elif evaluation2 > evaluation1:
    print("EVALUATION")
    print("The evaluation of " +
if way1 > way2:
    print("WAY")
    print(t1.getnametp() +
          " is easier to reach")
    print('
')
elif way2 > way1:
    print("WAY")
    print(t2.getnametp() +
          " is easier to reach")
    print('
')
else:
    print("WAY")
    print("The TPs have the same way to reach")


```
249     }
250
251     print('
')
252
253     # We compare the PageRank results
254     if pa1 > pa2:
255         print("PAGERANK")
256         print(  
257             "The PageRank result is bigger in "+  
258             t1.getnametp()  
259         )
260         print('
')
261     elif pa2 > pa1:
262         print("PAGERANK")
263         print(  
264             "The PageRank result is bigger in "+  
265             t2.getnametp()  
266         )
267         print('
')
268     else:
269         print("PAGERANK")
270         print("The PageRank results are the same")
271         print('
')
272
273     break

Code 2 ~ Compare
```
In Appendix D we present the whole code with the database and three potential examples of *Tourism Products*.

The program checks if the comparison is possible, otherwise, it breaks with an error message.

*Figure 9 and Figure 10* show the relative outputs.
The final Appeal of JetMarket is: 102

The final Appeal of FUn! is: 24

CITY
The City FUn! is bigger

PERIOD
The opening months of JetMarket are bigger then FUn!

PRICES
JetMarket is more expensive than FUn!

CAPACITY
The capacities are equal

EVALUATION
The evaluation of JetMarket is bigger than FUn!

WAY
The TPs have the same way to reach

PAGERANK
The PageRank result is bigger in FUn!

Figure 9 ~ Compare result
The final Appeal of SunMall is: 15

The final Appeal of JetMarket is: 102

We can't compare them!

*Figure 10 ~ Compare - Error message*
5. Conclusions

In the thesis we presented the Appeal in the Destination Monitor.

We wanted to realize a prototype of the program that retrieves the data from the Web and defines the Appeal. We chose two simples languages, similar to pseudo code, that could be immediately understood. The database is quite small, because we wanted to emphasize the final results.

We are aware that the program has some limitations, however it is open to improvement and further research, both optimizing hardware and better performancing programs, e. g:

- using objective languages (like C# or JAVA) which are more performative when using a big storage;

- when using Big Data of information, the database can be stored in high performance disks (with a lower latency when accessing the memory);
• since Crawler, PageRank and Wordrelations work on little portion of the Web, they can be optimized by using different data structure or changing the hardware components.

Some future works will focus on optimizing the hardware, or the better performance of the program and the access disks in which the data are saved.

Using appropriate indicator changes we can apply the project to different environments. For example, we can define the Appeal of a Movie (Movie Appeal): it will be possible to analyze the feedbacks from social networks and other tools present in the Web; then it will be possible to analyze which are the indicators, hence define the components in the average weight to calculate the final Appeal.
Glossary

**API:** Application Programming Interface. It is a set of definitions, protocols and tools for building software and application. It specifies how software components should interact (Collins).

[Chap. 1, par. 1.2, page 20]

**Appeal:** The power of a Tourism Product to attract please, stimulate, or interest (Collins).

[Chap. 2, par. 2.1.1, page 27]

**Array:** A regular data structure in which individual elements may be located by references to one or more integer index variables. (Collins)

[App. A, page 117]

[App. B, page 121]

**Benchmark:** A criterion by which to measure something; standard; reference point. (Collins)

[Chap. 4, par. 4.3.1, page 87]
**Big Data:** It is a huge amount of information that can only be computed by special computers.

[Chap. 5, page 101]

**Complexity:** The time complexity of an algorithm quantifies the amount of time taken by the algorithm itself to run as a function.

[Chap. 5, page 101]

**Crawler:** A computer program that is capable of performing recursive searches on the Internet. (Collins)

[Chap. 2, par. 2.1.2 page 28]
[App. A, page 117]

**Dashboard:** User interface that organizes and presents information in a way that is easy to read.

[Chap. 2, par. 1.2, page 19]
**Destination Monitor:** System that monitors the destination using indicators.

[Chap. 1, page 7]

**Graph:** A draw that explains the relation between certain elements by means of series of dots and lines.

[Chap. 2, par. 2.1.3 page 32]

**Google API:** It is a set of application programming interfaces developed by Google which allows communication with Google Services and their integration to other services.

(Google)

[Chap. 2, par. 2.1.4 page 35]

**Indicator:** something that provides an indication.

(Collins)

[Chap. 2, par. 2.2, page 38]

[Chap. 4, par. 4.1, page 62]
**Open Data:** Data that are available to everyone to use and republish as they wish, without restrictions of copyright.

[Chap. 1, par. 1.2, page 17]

[Chap. 2, page 23]

**PageRank:** Algorithm used to rank web sites in search engine results.

[Chap. 2, par. 2.1.3, page 32]

**Prestige:** The power to influence.

[Chap. 2, par. 2.1.3, page 33]

[App. B, page 121]

**Product Manager:** Person who is the manager of the *Tourism Product.*

**Rank:** Importance of a determinate page.

[Chap. 2, par. 2.13, page 33]
**Relational Model:** It is the representation of the organization of data into collections of two-dimensional tables called relations.

[Chap. 4, par. 4.2, page 68]

**Salt:** It is a random number used as an additional input to a one-way-hash password.

[Chap. 4, par 4.2.1, page 70]

**SHA256:** Secure Hash Algorithm, set of cryptographic hash functions.

[Chap. 4, par. 4.2.1, page 70]

**Singleton:** It is a design pattern that restricts the instantiation of an object.

[App. C, page 128]

**Superclass:** It is a class from which other classes are derived.

[Chap. 4, par. 4.2, page 69]
Tourism Product: It is any product in the field of tourism.

UML: Unified Modeling Language, it shows the actions of the various actors present in a determinate project.

URL: Uniform Resource Locator, string that addresses to a determinate web site.

Web graph: It describes the directed links between pages in the World Wide Web.
References

BOOKS AND PAPERS


www.collinslanguage.com

WEB SITES

[13] Orlando S., Data and Web Mining,

[14] Focardi R. Security,
https://secgroup.dais.unive.it/teaching/security-course/identification/

[15] TripAdvisor,
https://www.tripadvisor.it/PressCenter-c6-About_Us.html

[16] PaesiOnLine,
http://www.paesionline.info/company.asp?domain=.it

[17] CastelliExperience,
http://www.castelliexperience.it/?page_id=3394

[18] Trivago,
http://company.trivago.it
[19] Expedia.it,
https://www.expedia.it/p/support/chi-siamo

[20] GroupOn,
https://www.groupon.it/about-us

[21] Booking.com,
http://www.booking.com/content/about.it.html?label=gen173
nr-
1FCAEogg|CAhYSDNiBW5vcmVmaHGIAQGYARS4AQfAQA3YA
QHARfAQH4AQoAgM;sid=181bbe04226965633091bf7d303b444b

[22] Venere.com,
http://it.venere.com/page/about-us/

[23] TravelAppeal,
https://www.travelappeal.com/it/

[24] FourTourism,
http://www.fourtourism.it/chi-siamo/
[25] Evols,
http://www.evols.it/it/#/chi-siamo/

[26] Reputation Manager,
http://www.reputazioneonline.it/chi-siamo

[27] Venice Project Center,
https://github.com/sunng87/heatcanvas
http://veniceprojectcenter.org

[28] Trentino Region,
http://www.trentinomarketing.org/it/cosa-facciamo/progetti-di-sistema/h-benchmark/
http://www.develongroup.com
http://www.trustyou.com
http://hotelexecutive.com/author/800

[29] Puglia Region,
http://www.agenziapugliapromozione.it/portal/spot
[30] Piedmont Region,
http://www.dati.piemonte.it

[31] Emilia Romagna Region,
http://digitale.regione.emilia-romagna.it/dati/temi/accesso-ai-dati
http://digitale.regione.emilia-romagna.it/piter

[32] Lombardy Region,
https://www.dati.lombardia.it
https://socrata.com/case-study/lombardia-open-data-designed-for-growth/
https://www.youtube.com/watch?v=Lyew8C6tcsk
Appendix A

Appendix A refers to paragraph 2.1.2.

Crawler code

*Crawler* needs two elements as input: the first one is the URL in which it searches for the keywords, the second is an array of keywords.

The code is written in Python language.

```python
1 import requests
2 import re
3 import urlparse
4
5 # In this example we are trying to collect
6 # the occurrences
7 # of touristic words
8 # HTML <a> regexp
9 # Matches href="" attribute
10 link_re = re.compile(r'href="(..*?)"')
11
12 # Words list
13 wlist = ['cruise']
14```
def crawl(url, maxlevel):
    # Limit the recursion, we are not downloading
    # the whole Internet, a crawler goes in depth
    if (maxlevel == 0):
        return [0, 0]

    # Get the web page
    req = requests.get(url)
    result = [0, 0]

    # Check if successful
    if (req.status_code != 200):
        return [0, 0]

    # Find and follow all the links
    links = link_re.findall(req.text)
    for link in links:
        # Get an absolute URL for a link
        link = urljoin(url, link)
        temp = crawl(link, maxlevel - 1)
        for i, v in enumerate(temp):
            result[i] += v

    # Find all occurrences on current page
    count = 0
    index = 0
    for word in wlist:
        while True:
            index = req.text.find(word, index + len(word))
            if index == -1:
                index = 0
```python
46    count += 1
47    break
48    else:
49        result[count] += 1
50
51
52    return result
53
54    # We insert the URL in the parameter
55    def main():
56        occurrences = \n57            crawl(
58                'https://www.timeanddate.com/
59                'holidays/us/christmas-day'
60                , 2)
61
62        print("Results:")
63        for i,word in enumerate(wlist):
64            print("Word: " + word + " --> " + str(occurences[i]) + " occurrences")
65
66    if __name__ == "__main__":
67        main()

Code 3 ~ Crawler
```
Appendix B

Appendix B refers to paragraph 2.1.3.

PageRank code

The input is a txt file in which there are two columns: the first represents the outgoing link from one node of the web graph, the second column is the ingoing link to another node. This algorithm summarizes the graph of connections between nodes. It is written in C language.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main() {
    // number of nodes
    int n = 100;
    // indexes
    int i = 0;
    int j = 0;
    // input file
    FILE *fp;
```


```c
// matrix
float matrix[n][n];

// vector of outgoing links
int out_link[n];

// transposed matrix
float t_matrix[n][n];

// p vector
float p[n];

// new p vector
float p_new[n];

// iteration counter
int k = 0;
float error = 0;
int looping = 1;

// damping factor
float d = 0.85;

// initialization of the matrix
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
        matrix[i][j]=0;
    }
}

// initialization of the transposed matrix
for (i=0; i<n; i++) {
    for (j=0; j<n; j++){
        t_matrix[i][j]=0;
    }
}
```
// initializing the outlinks vector
for (i=0; i<n; i++){
    out_link[i]=0;
}

// initializing the p vector
for (i=0; i<n; i++){
    p[i]=1.0/n;
}

printf("File: \\
");

// opening file
fp = fopen("file100.txt", "r");
while (!feof(fp)) {
    // reading numbers from file
    fscanf(fp, "%d %d \\
", &i, &j);
    i = i;
    j = j;
    // putting the edges in the matrix
    matrix[i][j]= 1;
}

// counting the outlinks
for (i=0; i<n; i++){
    for (j=0; j<n; j++){
        if (matrix[i][j] != 0){
            out_link[i] =
            out_link[i] + 1;
        }
    }
}
// normalizing the dangling nodes' rows
for (i=0; i<n; i++){
    if (out_link[i] == 0){
        for (j=0; j<n; j++){
            matrix[i][j]=1.0/n;
        }
    }
    else{
        // normalizing the values of the rows
        for (j=0; j<n; j++){
            if (matrix[i][j] != 0.0){
                matrix[i][j] = matrix[i][j]/out_link[i];
            }
        }
    }
}

// transpose
for (i=0; i<n; i++){
    for(j=0; j<n; j++){
        t_matrix[j][i] = matrix[i][j];
    }
}

// probability vector
while (looping){
    // initialising the new p vector
    for (i=0; i<n; i++){
        p_new[i]=0;
    }
for (i=0; i<n; i++){
    for (j=0; j<n; j++){
        p_new[i] =
        p_new[i] + t_matrix[i][j] * p[j];
    }
}

for (i=0; i<n; i++){
    p_new[i] =
    (p_new[i]*d) + ((1.0 - d)/n);
}

error=0;

// check if we have to stop
for (i=0; i<n; i++){
    error =
    error + (fabsf(p_new[i] - p[i]));
}

if (error < 0.000001){
    looping = 0;
}

// updating p
for (i=0; i<n; i++){
    p[i] = p_new[i];
}
k = k + 1;
}

printf("Final P Vector:\n");
// printing the vector
for (i=0; i<n; i++) {
    printf("%3.3f ", p[i]);
}

printf("\n");

return 0;

Code 4 ~ PageRank
Appendix C

Appendix C refers to paragraph 2.1.4.

Wordrelations code

The inputs are two arrays. In the first we insert the keywords to be searched on Google, the second is the list of words to be searched within a page.

It is written in Python language.

```python
import sys
import requests
from requests.packages.urllib3.exceptions\n import InsecureRequestWarning
import re
import urlparse
from googleapiclient.discovery import build

# This is the list of words to be searched on Google
primary_keys = []

# This is the list of words to be searched within a page
related_keys = []
```
# How deep do we want to go?
level = 1

class WordRelations:
    # This class is managed as a Singleton
    # This static variable represents the
    # current active instance
    Instance = None

    # Google API keys
    GOOGLE_SEARCH_ID = "011139836630747192430:vubffkynfnO"
    GOOGLE_API_KEY = "AIzaSyB7NTxXFj4x1zTkahqWojfd-HjXXaoumZAQ"

    # To search deeper we need to find
    # the links within a page
    # We match here the href attribute
    PAGE_LINK = re.compile(r'href="(.*?)"')

    # Singleton instantiation
    @staticmethod
    def GetInstance():
        if WordRelations.Instance == None:
            Instance = WordRelations()

        return Instance

    # We google a certain word
    def GoogleSearch(self, word):
# We return a list of URLs
urls = []

# Google API engine must be initialized
google = build(
    "customsearch",
    "v1",
    developerKey = WordRelations.GOOGLE_API_KEY)

# GOOGLE IT!
results = google.cse().list(
    q = word,
    cx = WordRelations.GOOGLE_SEARCH_ID
).execute()

# For each result found by Google
# we extract the URL
for item in results['items']:
    # If a result is on a https site,
    # the URL is already ok
    # otherwise, add a http://
    if item['formattedUrl'][0:5] != "https":
        urls.append("http://" +
                     item['formattedUrl'])
    else:
        urls.append(item['formattedUrl'])

# Return the results
return urls

# Crawl a web page and find
# the related words occurrences
def Crawl(self, url, deep):
# Limit the recursion,

# we are not downloading the whole Internet
if (deep == 0):
    return [0] * len(related_keys)

# Download the webpage...
# Never verify the certificate
# (https can be a pain in the neck...)
try:
    req = requests.get(url, verify = False)
except:
    # If something goes wrong with the connection,
    # just ignore the web site
    print("Unrecoverable error: connection "
          "aborted for website: "
          + url)
    return [0] * len(related_keys)

result = [0] * len(related_keys)

# Check if successful
if (req.status_code != 200):
    return [0] * len(related_keys)

# Once the page is downloaded,
# look for all the links
links = WordRelations.PAGE_LINK.
findall(req.text)

# For each link...
for link in links:
Get an absolute URL

```python
link = urlparse.urljoin(url, link)
```

If the link does not start with 'http:',

```python
if link[0:4] != 'http':
    continue
```

Get the frequencies in the deeper link

```python
temp = self.Crawl(link, deep - 1)
```

Update the current to get the total

```python
for i, v in enumerate(temp):
    result[i] += v
```

Search each related keys in the page and count how many times they appear

```python
count = 0
index = 0
for word in related_keys:
    print("Checking related word: " + word)
    while True:
        index = req.text.find(
            word, index + len(word))
        if index == -1:
            index = 0
            count += 1
            break
        else:
            result[count] += 1

return result
```
# Search each primary word on Google
# For each website crawl it and find
# the related key occurrences

def CreateWordsRelationship(self):
    # The graph we're returning
    words_graph = []

    # For each word...
    for word in primary_keys:
        print("Checking key: " + word)
        # We google it
        google_results = self.GoogleSearch(word)
        # Init the results for the current word
        temp_results = [0] * len(related_keys)
        # For each website found by Google
        for url in google_results:
            print("Checking website: " + url)
            # Find all occurrences
            current_results =
                self.Crawl(url, level)
            # When found,
            # add them to the previous ones
            for i, result in enumerate(current_results):
                temp_results[i] += result

        # Ok, the current word has been checked,
        # let's Google the next
        words_graph.append(temp_results)

    return words_graph
# Search&Crawl and then save the result
# on a file called words_graph.txt

def main(self):
    engine = WordRelations.GetInstance()
    results = engine.CreateWordsRelationship()

    try:
        out_file = open("./words_graph.txt", "w")
        out_file.write("# Columns header: | ")
        for key in related_keys:
            out_file.write(key + " | ")
        out_file.write("\n# Rows header: | ")
        for key in primary_keys:
            out_file.write(key + " | ")
        out_file.write("\n\n")

        for row in results:
            for col in row:
                out_file.write(str(col) + "\t")
            out_file.write("\n")

        out_file.close()
    except IOError as e:
        print("FATAL: I/O error: " + e.strerror)

    print("\n\nExecution ended. Check file "."./words_graph.txt for results.")
# If we are in the main,
# we create perform some activities and the let's
# find the results!

if __name__ == "__main__":
    requests.packages.urllib3.disable_warnings(
        InsecureRequestWarning)

    if len(sys.argv) < 3:
        print("USAGE: ./wordrelations.py "
              "<primary_words_list>
              <related_words_list>\n")

        print("Lists are in the form"
              " item1,item2,...,itemN\n")

        print("primary_words_list "
              "is the list of words to be googled")

        print("related_words_list is "
              "the list of words "
              "to be found for each web site "
              "found by google\n")

        exit(0)

    primary_keys = sys.argv[1].split(',')

    related_keys = sys.argv[2].split(',')

    WordRelations.GetInstance().main()
Appendix D

Appendix D refers to paragraphs 4.3.

Appeal and Compare codes

In the following Code 6, we show the creation of the twelve classes and the Appeal and Compare codes.

The program is written in Python language.

```python
1 def main():
2
3     # Person class
4     class Person:
5
6         def __init__(
7             self, name, email, password, birthday
8         )::
9             # "name" is a string, it is the person's name
10             self.name = name
11             # "email" is a string,
12             # it is the username used to log in
13             self.email = email
14             # "password" is a string,
15             # it is the password to used to log in
16             self.password = password
```
# "birthday" is a string
self.birthday = birthday

# getname() retrieves the person's name, 
# and setname() changes the person's name
def getname(self):
    return self.name

def setname(self, a):
    self.name = a
    return self.name

# getemail() retrieves the person's email, 
# and setemail() changes the person's email
def getemail(self):
    return self.email

def setemail(self, a):
    self.email = a
    return self.email

# getpwd() retrieves the person's password, 
# and setpwd() changes the person's password
def getpwd(self):
    return self.password

def setpwd(self, a):
    self.password = a
    return self.password

# getbday() retrieves the person's birthday,
# stebday() changes the person's birthday

def getbday(self):
    return self.birthday

def setbday(self, a):
    self.birthday = a
    return self.birthday

# Manager class, it is a subclass of Person

class Manager(Person):
    
def __init__ (
        self, name, email, password, birthday
    ):  
        Person.__init__(
            self, name, email, password, birthday
        )

# Client class, is a subclass of Person

class Client(Person):
    
def __init__ (
        self, name, email, password, birthday
    ):  
        Person.__init__(
            self, name, email, password, birthday
        )

# Review class

class Review:

Giulia Schiavon 826788
def __init__(self, idreview, name, value, description):
    # “idreview” is an int
    self.idreview = idreview
    # “name” is the client's name,
    # it is a Client type
    self.name = name
    # “value” is an int
    # 1, 2, 3, 4, 5
    self.value = value
    # “description” is a string
    # in which there is the feedback
    self.description = description

    # getidreview() retrieves the id of the review,
    # setidreview() changes the id of the review
    def getidreview(self):
        return self.idreview

    def setidreview(self, a):
        self.idreview = a
        return self.idreview

    # getclient() retrieves the client,
    # setclient() changes the client
    def getclient(self):
        return self.name

    def setclient(self, a):
        self.name = a
return self.name

def getvalue(self):
    return self.value

def setvalue(self, a):
    self.value = a
    return self.value

def getdescription(self):
    return self.description

def setdescription(self, a):
    self.description = a
    return self.description

class Typology:
    def __init__(self):
        self = self

class Holidaymaker(Typology):
    def __init__(self):
class Overnightstay(Typology):
    def __init__(self):
        Typology.__init__(self)

class Serviceinloco(Typology):
    def __init__(self):
        Typology.__init__(self)

class Transport(Typology):
    def __init__(self):
        Typology.__init__(self)

class TourismProduct:
    def __init__(
        self, id, nametp, name, visitornum,
        birthday, city, typology, surface,
        period, arrive, pagerank, prices,
        quality, howtoarrive, numreview
[172] }

[173] # "id" is an int
[174] self.id = id
[175] # "nametp" is a string
[176] self.nametp = nametp
[177] # "name" is a Client type
[178] self.name = name
[179] # "visitorum" is an int
[180] self.visitorum = visitornum
[181] # "birthday" is a string
[182] self.birthday = birthday
[183] # "city" a City type
[184] self.city = city
[185] # "typology" is a Typology type
[186] self.typology = typology
[187] # "area" is an int
[188] self.surface = surface
[189] # "period" is an int
[190] self.period = period
[191] # "path" is a boolean
[192] self.arrive = arrive
[193] # "pagerank" is an int
[194] self.pagerank = pagerank
[195] # "prices" is an int
[196] self.prices = prices
[197] # "quality" is an int
[198] self.quality = quality
[199] # "howtoarrive" is Howtoarrive type
[200] self.howtoarrive = howtoarrive
[201] # "numreview" is an int
[202] self.numreview = numreview
```python
# getid() retrieves the id,
# setid() changes the id

def getid(self):
    return self.id


def setid(self, a):
    self.id = a
    return self.id

# getnametp() retrieves the name of the TP,
# setnametp() changes the name of the TP

def getnametp(self):
    return self.nametp


def setnametp(self, a):
    self.nametp = a
    return self.nametp

# getname() retrieves the manager's name,
# setname() changes the manager's name

def getname(self):
    return self.name


def setname(self, a):
    self.name = a
    return self.name

# getvisitornum() retrieves
# the visitors number
# setvisitornum() changes the visitors number
```
def getvisitornum(self):
    return self.visitornum

def setvisitor(self, a):
    self.visitornum = a
    return self.visitornum

# getbirthday() retrieves the date
# of the TP has been started,
# setbirthday() changes the date

def getbirthday(self):
    return self.birthday


def setbirthday(self, a):
    self.birthday = a
    return self.birthday

# getcity() retrieves the city,
# setcity() changes the city

def getcity(self):
    return self.city


def setcity(self, a):
    self.city = a
    return self.city

# gettypology() retrieves the typology,
# settypology() changes the typology

def gettypology(self):
    return self.typology
def settypology(self, a):
    self.typology = a
    return self.typology

def getsurface(self):
    return self.surface

def setsurface(self, a):
    self.surface = a
    return self.surface

def getperiod(self):
    return self.period

def setperiod(self, a):
    self.period = a
    return self.period

def getarrive(self):
    return self.arrive

def setarrive(self, a):
    self.arrive = a
    return self.arrive
getpagerank() retrieves the pagerank value,
setpagerank() changes the pagerank value

def getpagerank(self):
    return self.pagerank

def setpagerank(self, a):
    self.pagerank = a
    return self.pagerank

getprices() retrieves prices,
setprices() changes prices

def getprices(self):
    return self.prices

def setprices(self, a):
    self.prices = a
    return self.prices

getquality() is the arithmetic average
of the total value of the review
setquality() changes the value
def getquality(self):
    return self.quality

def setquality(self, a):
    self.quality = a
    return self.quality

gethowtoarrive() retrieves the value,
sethowtoarrive() changes the value
def gethowtoarrive(self):
return self.howtoarrive

def sethowtoarrive(self, a):
    self.howtoarrive = a
    return self.howtoarrive

# getreview() retrieves the int
# of the number of the reviews,
# setreview() changes the int
def getreview(self):
    return self.numreview

def setreview(self, a):
    self.numreview = a
    return self.numreview

# City class
class City:
    def __init__(self, 
        self, big, namect, cinum, visum):
        # “dimension” is an int,
        # we have three value:
        # 50 capital
        # 30 chief
        # 15 town
        # 5 village
        self.big = big
        # “namect” is a string
self.namect = namect

# "cinum" is an int
self.cinum = cinum

# "visnum" is a int
self.visnum = visnum

# getnamect() retrieves the name city,
# setnamect() changes the name city
def getnamect(self):
    return self.namect
def setnamect(self, a):
    self.namect = a
    return self.namect

# getbig() retrieves the value,
# setbig() changes the value
def getbig(self):
    return self.big
def setbig(self, a):
    self.big = a
    return self.big

# getcinum() retrieves the number of citizens,
# setcinum() changes the number of citizens
def getcinum(self):
    return self.cinum
def setcinum(self, a):
    self.cinum = a
return self.cinum

# getvisnum() retrieves the number
# of the visitors,
# setvisnum() changes the number of visitors
def getvisnum(self):
    return self.visnum

def setvisnum(self, a):
    self.visnum = a
    return self.visnum

# Howtoarrive class
class Howtoarrive:

def __init__(self, frequency, howtoarrive):
    # “frequency” is an int,
    # 3 -> 1 min to 1 hour
    # 2 -> 1 hour to 3 per day
    # 1 -> over 3 per day
    self.frequency = frequency
    # howtoarrive is an int,
    # Plane = 2
    # Train or Boat = 3
    self.howtoarrive = howtoarrive

def getfrequency(self):
    return self.frequency
def setfrequency(self, a):
    self.frequency = a
    return self.frequency

# gethowtoarrive() retrieves the int,
# sethowtoarrive() changes the value
def gethowtoarrive(self):
    return self.howtoarrive

def sethowtoarrive(self, a):
    self.howtoarrive = a
    return self.howtoarrive

# Appeal returns an int
# Appeal is the value of the Tourism Product
def Appeal(tourism):
    # Parameters used in the weighted average
    # CITY
    ct = tourism.getcity().getbig()
    # CAPACITY
    su = tourism.getsurface()
    vs = tourism.getvisitornum()
    capacity = su/vs
    # EVALUATION
    ev = tourism.getreview()
    qu = tourism.getquality()
    evaluation = ev/qu
`# PAGERANK
pa = tourism.getpagerank()
`# WAY
`wy = tourism.gethowtoarrive()
way = 0
`# Check if the Tourism Product
# has the parameter
`if wy == False:
    way
`elif wy == True:
    # WAY
    w1 =
        tourism.gethowtoarrive()
        .getfrequency()
    w2 =
        tourism.gethowtoarrive()
        .gethowtoarrive()
    way = w1 + w2
`# PRICES
`pr = tourism.getprices()
`# PERIOD
`pe = tourism.getperiod()
`# We differentiate the different typologies
# because the prices change
`if tourism.gettypology() == Serviceinloco:
    prices = pr / pe
`elif tourism.gettypology() == Holidaymaker:
    prices = (su / pr) / pe
`elif tourism.gettypology() == Overnightstay:
    prices = pr / pe
`else:
prices = pr / (tourism.getvisitornum() + tourism.getcity().getcinum())

# We take seven parameters
# to have the final Appeal
# "ct" capital/chief/town/village
# "pe" is the opening months
# "prices" is the arithmetic average
# of the prices
# "capacity" is the relationship between
# "surface area" and "number of visitors"
# "evaluation" is the relationship between
# "number of evaluation" and
# "average of the quality"
# "way" is the result
# of "frequency" and "transport"
# "pa" is the result of PageRank
appeal =
    (ct + pe + prices +
     capacity + evaluation +
     way + pa) / 7

print("The final Appeal of " +
    tourism.getnametp() +" is:")
print(appeal)
print(\n)

# We compare two Tourism Products
# with the same Typology
def Compare(t1, t2):

Appeal(t1)

Appeal(t2)

# Parameters of the first TP

# CITY t1
ctl = t1.getcity().getbig()

# PERIOD t1
pel = t1.getperiod()

# CAPACITY t1
sul = t1.getsurface()
vs1 = t1.getvisitornum()
capacity1 = sul / vs1

# PRICES t1
pr1 = t1.getprices()

# We differentiate the different typologies,
# first Tourism Product

if t1.gettypology() == Serviceinloco:
    prices1 = pr1 / pel

elif t1.gettypology() == Holidaymaker:
    prices1 = (sul / pr1) / pel

elif t1.gettypology() == Overnightstay:
    prices1 = pr1 / pel
else:
    prices1 = pr1 / (t1.getvisitornum() + t1.getcity().getcinum())

# EVALUATION t1
evl = t1.getreview()
qu1 = t1.getquality()
evaluation1 = evl / qu1
544  # WAY t1
545  wy1 = t1.gethowtoarrive()
546  # Check if the first Tourism Product
547  # has the parameter
548  way1 = 0
549  if wy1 == False:
550      way1
551  elif wy1 == True:
552      w1t1 = t1.gethowtoarrive()
553      .getfrequency()
554      w2t1 = t1.gethowtoarrive()
555      .gethowtoarrive()
556      way1 = w1t1 + w2t1
557  # PAGERANK t1
558  pal = t1.getpagerank()
559
560  # Parameters of the second TP
561  # CITY t2
562  ct2 = t2.getcity().getbig()
563  # PERIOD t2
564  pe2 = t2.getperiod()
565  # CAPACITY t1
566  su2 = t2.getsurface()
567  vs2 = t2.getvisitornum()
568  capacity2 = su2/vs2
569  # PRICES t2
570  pr2 = t2.getprices()
571  # We differentiate the different typologies,
572  # second Tourism Product
573  if t2.gettypology() == Serviceinloco:
574      prices2 = pr2 / pe2
elif t2.gettypology() == Holidaymaker:
    prices2 = (su2 / pr2) / pe2
elif t2.gettypology() == Overnightstay:
    prices2 = pr2 / pe2
else:
    prices2 = pr2 / (
        t2.getvisitornum() +
        t2.getcity().getcinum()
    )

# EVALUATION t2
ev2 = t2.getreview()
qu2 = t2.getquality()
evaluation2 = ev2/qu2

# WAY t2
wy2 = t2.gethowtoarrive()

# Check if the second Tourism Product has the parameter
way2 = 0
if wy2 == False:
    way2
elif wy2 == True:
    w1t2 = t2.gethowtoarrive()
    .getfrequency()
    w2t2 = t2.gethowtoarrive()
    .gethowtoarrive()
    way2 = w1t2 + w2t2

# PAGERANK t2
pa2 = t2.getpagerank()

# We compare the parameters of both TPs
while True:
# First control,
# if the TPs are different,
# we can not compare them
if t1.gettypology() != t2.gettypology():
    print("We can't compare them!")
    print('\n')
    break

# We compare the city
# where the TPs are located in
if ct1 > ct2:
    print("CITY")
    print("The City "
    + t1.getnametp()
    + " is bigger"
    )
    print('\n')
elif ct2 > ct1:
    print("CITY")
    print("The City "
    + t2.getnametp() +
    " is bigger"
    )
    print('\n')
elif ct1 == ct2:
    print("CITY")
    print("The cities are big both")
# We compare the period of opening
if pe1 > pe2:
    print("PERIOD")
    print("The opening months of ",
          + t1.getnametp() +
          " are bigger then ",
          + t2.getnametp()
          )
    print('\n')
elif pe2 > pe1:
    print("PERIOD")
    print("The opening months of ",
          + t2.getnametp() +
          " are bigger then ",
          + t1.getnametp()
          )
    print('\n')
else:
    print("PERIOD")
    print("The opening months "
          " of the TPs are equal"
          )
    print('\n')

# We compare the arithmetic averages
668 # of the prices
669 if prices1 > prices2:
670     print("PRICES")
671     print(
672         t1.getnametp() +
673         " is more expensive than "
674         + t2.getnametp()
675     )
676     print('\n')
677 elif prices2 > prices1:
678     print("PRICES")
679     print(
680         t2.getnametp() +
681         " is more expensive than "
682         + t1.getnametp()
683     )
684     print('\n')
685 else:
686     print("PRICES")
687     print(
688         "The prices are equal"
689     )
690     print('\n')
691
692 # We compare the capacities
693 if capacity1 > capacity2:
694     print("CAPACITY")
695     print(
696         "The capacity of "
697         + t1.getnametp() +
698         " is bigger than "
699     )
700     print('\n')
```python
+ t2.getnametp()

)  

print('
')  

elif capacity2 > capacity1:
    print("CAPACITY")
    print("The capacity of "+ t2.getnametp() + " is bigger than "+ t1.getnametp())
    print('
')
else:
    print("CAPACITY")
    print("The capacities are equal")
    print('
')

# We compare the evaluations
if evaluation1 > evaluation2:
    print("EVALUATION")
    print("The evaluation of "+ t1.getnametp() + " is bigger than "+ t2.getnametp())
    print('
')
elif evaluation2 > evaluation1:
    print("EVALUATION")
```

    print("The evaluation of " +
    t2.getnametp() +
    " is bigger than " +
    t1.getnametp()
    )

    print('\n')

else:
    print("EVALUATION")
    print(
        "The number evaluations are equal"
    )
    print('\n')

# We compare way parameter
if way1 > way2:
    print("WAY")
    print(
        t1.getnametp() +
        " is easier to reach"
    )
    print('\n')

elif way2 > way1:
    print("WAY")
    print(
        t2.getnametp() +
        " is easier to reach"
    )
    print('\n')

else:
    print("WAY")
print("The TPs have"
    "the same way to reach"
)
print('n')

# We compare the PageRank results
if pa1 > pa2:
    print("PAGERANK")
    print("The PageRank result is"
        "bigger in " +
        t1.getnametp()
    )
    print('n')
elif pa2 > pa1:
    print("PAGERANK")
    print("The PageRank result is"
        "bigger in " +
        t2.getnametp()
    )
    print('n')
else:
    print("PAGERANK")
    print("The PageRank results are"
        "the same")
    print('n')
break
# Some examples

# Managers
ben = Manager("Ben", "ben@ulli.com", "benebnene", "7/8/1978")

# Clients
trilli = Client("Trilli", "trilli.miao@trilli.com", "miaomiao", "17/5/2005")

# Reviews
a = Review(23, trilli, 5, "this is fantastic")

b = Review(45, giu, 1, "not good")
# Cities
london = City(50, "London", 789564215458, 7878788154131)
venice = City(50, "Venice", 7884154478, 5887818989898)
smallville = City(30, "Smallville", 89890, 909980)
cicut = City(15, "Cicut", 8989, 76767)

# How To Arrive
ha1 = Howtoarrive(3, 2)
ha2 = Howtoarrive(2, 4)
ha3 = None

# Tourism Products
854  jetmarket = \n855      TourismProduct(
856          78, "JetMarket", ben, 78778,
857          "8/8/1988", smallville, Serviceinloco,
858          56547, 12, True, 5, 7878, 5, ha1, 89
859      )
860  sunmall = \n861      TourismProduct(
862          5, "SunMall", cam, 7455521,
863          "8/5/2012", venice, Holidaymaker,
864          78878, 12, True, 26, 8568774, 5, ha1, 85
865      )
866  fun = \n867      TourismProduct(8, "FUN!", ben, 855151,
868          "5/5/2005", london,
869          Serviceinloco,
870          555, 4, False,
871          89, 74, 3, ha3, 25
872      )
873
874  #Appeal(fun)
875  #Compare(sunmall, jetmarket)
876  Compare(jetmarket, fun)
877
878  if __name__ == "__main__":
879      main()