UNIVERSITY OF OVIEDO

PhD program in Computer Science

INFLUENCE OF THE INCLUSION OF UNSTRUCTURED DATA IN RECOMMENDER SYSTEMS

THESIS PRESENTED BY PABLO PÉREZ NÚÑEZ
DOCTORATE IN COMPUTER SCIENCE

Thesis supervisors
Dr. Jorge Díez Peláez
Dr. Oscar Luaces Rodríguez

Gijón, 2022

Universidad de Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: *Influencia de la incorporación de información no estructurada en sistemas de recomendación*	Inglés: *Influence of the inclusion of unstructured data in recommender systems*

2.- Autor	
Nombre: **Pablo Pérez Núñez**	DNI/Pasaporte/NIE:
Programa de Doctorado: **Doctorado en Informática**	
Órgano responsable: **Centro Internacional de Postgrado, Universidad de Oviedo**	

RESUMEN (en español)

La gran cantidad de datos disponibles a través de la red hacen más que patente la necesidad de sistemas que nos ayuden a separar el contenido relevante del que carece de importancia. Esta tarea es realizada por los llamados Sistemas de Recomendación, y en la actualidad podemos encontrarlos en la mayoría de las webs que utilizamos diariamente.

Estos sistemas se entrenan, tradicionalmente, haciendo uso de la interacciones históricas de los usuarios (compras realizadas, canciones o videos reproducidos, noticias leídas, ...) puesto que no suele existir más información disponible para hacerlo. En la actualidad, muchas plataformas permiten también que los usuarios extiendan su opinión sobre la interacción mediante texto o imágenes (información no estructurada), los cuales, creemos que pueden utilizarse a la hora de crear o mejorar un Sistema de Recomendación.

En los diversos trabajos de esta tesis pretendemos sacar provecho de este tipo de información (texto e imágenes particularmente) de diferentes formas con el objetivo de, no solo mejorar el rendimiento de los Sistemas de Recomendación, también su experiencia de uso, las formas en las que obtener una recomendación o la presentación de las recomendaciones finales entre otros.

La tesis se divide, principalmente, en cuatro partes, en la primera de ellas detallamos como se ha creado y analizamos el conjunto de datos que se utilizará a lo largo de todo el documento. La parte número dos contiene todos los trabajos relacionados con el uso del texto, en este caso un Sistema de Recomendación de restaurantes basado en texto con capacidad de generación de explicaciones así como transparencia.

En la tercera parte del documento, la más extensa, se encuentran los trabajos relacionados con el campo de las imágenes. El primer trabajo presenta un sistema capaz de predecir, a partir de una foto, el usuario que la ha realizado. El objetivo que se persigue es poder utilizar este sistema para, cuando un usuario entra en la web de un restaurante, mostrarle solo aquellas fotos que, aunque no las ha realizado, podría haberlo hecho (poseen un estilo o contenido similar al del usuario).

La siguiente propuesta relacionada con imágenes presenta una nueva forma de crear embeddings para imágenes a partir de los cuales se pueden realizar recomendaciones. Esta nueva codificación se diferencia de las tradicionales en que no está basada en el contenido de la imagen, si no que en los gustos de los usuarios.

Partiendo de la codificación anterior, en el siguiente trabajo se presenta un sistema capaz de resumir, en pocas imágenes los gustos gastronómicos de una ciudad. El sistema es capaz de crear grupos de usuarios según sus gustos y de seleccionar, para cada uno, las imágenes que mejor los representen.

Finalmente, gracias a lo aprendido en las propuestas anteriores, en el último trabajo de esta parte y de la tesis, se presenta una aplicación móvil capaz de generar recomendaciones de restaurantes partiendo únicamente de una imagen de comida.

La última parte del documento lista las publicaciones relacionadas con los trabajos presentados y detalla su lugar de publicación, impacto y estado. Para finalizar se cierra el documento con las conclusiones globales de la tesis.

RESUMEN (en inglés)

The vast amount of data available on the web makes the need for systems that help us
to distinguish relevant content from irrelevant content more than evident. This task is carried out by the so-called Recommendation Systems, and we can currently find them on most of the websites we use on a daily basis.

These systems are traditionally trained by making use of users' historical interactions (purchases made, songs or videos played, news read, ...) since there is usually no other information available to do so. Nowadays, many platforms also allow users to provide feedback on the interaction through text or images (unstructured information), which, we believe, can be used when creating or improving a Recommender System.

In the multiple works of this thesis, we intend to take advantage of this type of information in different ways with the aim of, not only improving the performance of Recommender Systems, but also their user experience, the ways in which to obtain a recommendation or the presentation of the final recommendations among others.

The thesis is mainly divided into four parts, in the first part we detail how the dataset that will be used throughout the paper has been created and analysed. Part two contains all the work related to the use of text, in this case a text-based restaurant Recommendation System with the ability to generate explanations as well as transparency.

In the third part of the document, the most extensive, are the works related to the field of images. The first work presents a system capable of predicting, from a photo, the user who has taken it. The aim is to be able to use this system, when a user enters a restaurant's website, to show only those photos that, although not taken by the user, could have been taken by the user (they have a similar style or content to the user).

The following proposal related to images presents a new way of creating embeddings for images from which recommendations can be made. This new coding differs from traditional coding in that it is not based on the content of the image, but on the tastes of the users.

Based on the previous coding, the following work presents a system capable of summarising, in a few images, the gastronomic tastes of a city. The system is able to create groups of users according to their tastes and to select, for each one, the images that best represent them.

Finally, thanks to what has been learned in the previous proposals, the last work of this part and of the thesis presents a mobile application capable of generating restaurant recommendations based only on food images.

The last part of the document lists the publications related to the works presented and details their form of publication, impact and status. Finally, the document closes with the overall conclusions of the thesis.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN INFORMÁTICA

University of Oviedo

PhD program in Computer Science

Influence of the inclusion of unstructured data in recommender systems

Thesis presented by Pablo Pérez Núñez
Doctorate in computer science

Thesis supervisors
Dr. Jorge Díez Peláez
Dr. Oscar Luaces Rodríguez

Gijón, 2022

*A mis directores de tesis y
resto de miembros del AIC.*

Abstract

The vast amount of data available on the web makes the need for systems that help us to distinguish relevant content from irrelevant content more than evident. This task is carried out by the so-called *Recommender Systems*, and we can currently find them on most of the websites we use on a daily basis.

These systems typically learn from the user's consumption history (purchases, listens, clicks, ...) but rarely make use of additional information provided by users in the form of natural language texts or images (unstructured data).

In the multiple works of this thesis we intend to take advantage of this type of information in different ways with the aim of improving the performance of Recommender Systems as well as users experience, the ways in which to obtain a recommendation or the presentation of the final recommendations among others.

Resumen

La gran cantidad de datos disponibles a través de la red hacen más que patente la necesidad de sistemas que nos ayuden a separar el contenido relevante del que carece de importancia. Esta tarea es realizada por los llamados *Sistemas de Recomendación*, y en la actualidad podemos encontrarlos en la mayoría de las webs que utilizamos diariamente.

Para aprender, típicamente, estos sistemas hacen uso de el historial de consumo del usuario (compras, reproducciones, clicks, ...) pero rara vez utilizan la información adicional aportada por los usuarios en forma de reseñas textuales o imágenes (datos no estructurados).

En los diversos trabajos de esta tesis pretendemos sacar provecho de este tipo de información de diferentes formas con el objetivo de, no solo mejorar el rendimiento de los Sistemas de Recomendación, si no también su experiencia de uso, las formas en las que obtener una recomendación o la presentación de las recomendaciones finales entre otros.

Contents

III Using images   50

4 *ELVis*: *Explaining Likings Visually*   51

5 *Sem*: *Semantics of Images*   74

IV Publications and conclusions 124

List of Figures

List of Tables

Chapter 1

Introduction

In today's globalized and connected world, the amount of digital information generated and consumed every day is unimaginable. It is estimated that, on average, each person on the planet generates about 1.7MB per second, resulting in approximately 40 trillion gigabytes or 40 zettabytes generated in 2020 alone.

All this information is often stored in large data centers that allow users to make queries in the future. Given the magnitude of the dataset, when a query is made, a user may be confronted with hundreds of millions of results, which makes evident the need for a system that helps to separate the relevant elements from the less important ones. Such systems are called **Recommender Systems (RS)** (Resnick and Varian, 1997).

1.1 Recommender Systems

Most of the services we use today through digital media make recommendations to their users by means of these Recommender Systems. This is the case, for example, when we enter the website of a digital newspaper, where the order in which the news are placed is, in a way, a reading recommendation. The ordered lists of news that appear in the margins (*the most read* or *the most commented*) are also a recommendation, based in this case on the interaction of other users with the digital publication. Another example of a recommendation can be found within each of the news items in the form of links to other news; these are the *related news* type of recommendations.

Service providers have many reasons to use and implement such systems in their shops or applications (Ricci et al., 2011):

- **Increase sales:** Encourage users to consume more products than would otherwise be the case without a recommendation system. In the case of companies that do not sell physical products (such as free video platforms, e.g YouTube), the goal is to keep the user connected to the service in order to show them more adverts.

- **Sell more diverse items:** RS allow users to find products that would otherwise be difficult to find. A service provider wants to sell all types of content, not just the most popular one.

- **Increase user satisfaction:** When receiving good recommendations they are actually looking for, users feel well advised and therefore satisfied.

- **Enhance user loyalty:** If the RS makes use of the user's history to provide recommendations, the user will feel that not only their recent behavior, but also their past behavior is taken into account, encouraging her/him to use it again.

- **Understand user behavior:** An RS allows to know the behavioral patterns of users: what kind of products they consume, how often and in what order. This information is very useful for service providers, allowing them to predict demand or generate specific offers.

The use of RS is very common in online shops. In this case, recommendations are of the form *customers who saw this product also saw . . .*, *bought together regularly* or even product rankings based on the number of sales or buyer reviews. We can also find recommendations on multimedia content streaming platforms, for example Netflix or Spotify, or on websites dedicated to managing hotel and restaurant reviews, such as TripAdvisor, Yelp, etc.

Some of the examples mentioned above are elementary **non-personalized** RS, which base their recommendations simply on majority behavior or tastes: if many people watch a film and, moreover, the ratings are very good, then it is quite likely to be recommended to users who have not yet watched it.

Other systems, the **personalized ones**, create specific recommendations for each available user. They are more common on platforms where a user registration is required to store all consumption history.

To generate personalized recommendations, the most usual approach is to create user profiles that condense, in some way, the tastes or preferences of the users (Díez et al., 2016). These consumption profiles will encode a user's history of interactions with products, where interaction is understood as any form of consumption of a product: the purchase of an item, listening to a song, viewing a product in an online shop, reading a news item in a digital publication, and so on.

This can be achieved by, for example, constructing embeddings using matrix factorization (Koren et al., 2009). Similarly, products can also be represented by vectors that summaries the interaction that has taken place with them.

1.1.1 Types of Recommender Systems

In order to be able to carry out any kind of recommendation, it is necessary to have data that either represent the interaction between user and items (click, like, rating, ...) or data that describe the item (video, audio, text, ...) or user (age, gender, ...). Depending on the availability of this data, different recommendation strategies can be applied:

- **Content-based:** To create this kind of systems we need information about the content of each of the items to recommend. For example, if we are working with movies, we might know, for each one, the genre, the duration, the actors, ... and also details of the users of the system (age, gender, location, ...). With this information, a system can learn to recommend similar items to the ones the user liked in the past. The similarity is calculated based on the known information of the items. In the case of movies, if a user rated a comedy movie positively, the system can learn to recommend more items of this genre. Knowing so much detail about the items is not usual, so this kind of systems cannot always be implemented.

- **Collaborative Filters (CF):** These systems are implemented when we do not have detailed information about users and items, but there is a rating of the items by the users (which is more common). Early versions were based on a *user-to-user* methodology, i.e., recommending to the user items that other similar users (same tastes) have liked in the past. In this context, two users are more similar the greater the number of matches in their consumption history. Later on, *item-to-item* methodology emerged to remedy some of the weaknesses of the previous system, in this case, the aim is to predict the rating a user would give to a product based on the ratings given to similar items.

- **Hybrid:** This category combines the previous categories using information from the content of the item as well as ratings made by other users. Systems of this type were less common in the past, as they require a lot of information (not always available), but nowadays are increasingly becoming more popular.

- **Demographic:** The popularity of this type of systems comes from the marketing world. They start from the premise that different demographic groups will require different recommendations. These groups can be made according to various criteria such as language, country or age of the users.

- **Community-based:** Also known as *Social Recommender Systems*, this kind of systems generates recommendations based on the preferences of the user's friends. It is based on the idea: *Tell me who your friends are, and I will tell you who you are*, since people tend to listen more to recommendations from friends than from anonymous people or opaque systems.

All the recommender systems we have discussed store the interactions between users $\vec{u} \in \mathcal{U}$ and products $\vec{it} \in \mathcal{I}$ in the so-called *user-item* matrix. In each position of this matrix a void represents that there has been no interaction; otherwise the values are usually in the set 0,1 to represent whether it was liked or disliked.

$$(\vec{u}, \vec{it}) \rightsquigarrow [0|1]. \tag{1.1}$$

It can also include some kind of information or evaluation by the user in the form of a numerical score (rating, stars...).

$$(\vec{u}, \vec{it}) \rightsquigarrow [1 \dots 5]. \tag{1.2}$$

However, on some platforms, e.g., Amazon and TripAdvisor, users do not only give stars to the product, but can also write a review and take relevant pictures in their interaction. This type of additional information (images, video, audio or text) is known as **unstructured information** and is not usually exploited in RS beyond the improvement of recommendations. In this thesis we intend to exploit this information much more through different approaches such as adding explanations to the given recommendations or creating a custom way of encoding images.

1.2 Unstructured information

Nowadays, multiple types of data are available, from the most basic, numbers, to one of the most complex, video, with a wide range of intermediate elements such as text, audio or images. These elements can be classified in two categories according to its internal organization: *structured* and *unstructured* information.

Structured data, as the name suggests, represents all the information that follows an structure or data model and can therefore be easily organized. Examples of this category are dates, dimensions or coordinates.

Unstructured data, on the other hand, is data that is not organized according to a predetermined data model or schema and cannot be stored in a conventional relational database system. Images, texts, videos or audio files are all examples of unstructured information.

As mentioned above, in this thesis we will focus our attention in unstructured information, and particularly on **texts** and **images**. Next, we will detail the most popular encoding techniques for each of these types of information.

1.2.1 Text

Text is the most abundant type of unstructured data available, we can find it in each of the websites we visit and, as users, we generate and consume large amounts of it in the average course of a day (mails, messages, posts, ...).

Text has to be encoded in order to be processed by an algorithm. The encoding of textual information usually requires an approach to encode individual words and then encode the entire text/document.

Word encoding

The most straightforward word encoding method is ***one-hot***. One-hot encoding represents each word with a vector of length n, where n is the number of words in the vocabulary. Each word has an unique index so that its one-hot vector will have a 1 in the corresponding position, and the rest will be zeros.

Although simple, this method, has several disadvantages. The first is the length of the vocabulary; in most cases our set of possible words is very large and so will be each of the vectors we create to encode each word. The second disadvantage is less obvious; given the nature of this encoding, the semantics of the words are not taken into account. Thus, the difference between vectors representing two very similar words (synonyms, for instance) is the same as the difference between completely unrelated words.

In order to fix these problems, more complex and elaborate encodings emerged such as ***Word2vec*** (Mikolov et al., 2013) or ***GloVe*** (Pennington et al., 2014). In these, by means of a learning task based on neural networks and matrix factorization respectively, an embedding (of user-defined size) containing information about the meaning and context of the word is learned. This type of encodings is currently widely used because, once trained with a very large corpus of words, the resulting embedding can be reused in many other tasks without the need to retrain the net from scratch.

Text encoding

When encoding a text, the simplest method is the so-called ***Bag of Words (BoW)*** method. It simply adds up all the one-hot encodings of the words in the text, obtaining a single vector of the vocabulary size, where we will have values greater than zero in those positions corresponding to the words in the text and zeros in the rest. Another option is to simply put a one in the words that appear in the text regardless of their frequency, but this method is less common as it loses crucial information in many cases.

As with the one-hot, the BoW has several disadvantages, the most notable of which is the loss of order. This can affect, for example, in texts where we have several words and one of them appears negated, this encoding does not know which of them the negation refers to. The encoding of the phrase *I like cars but no motorbikes* will be the same as *I like motorbikes but no cars*.

One way to partially solve this issue is through the creation of **n-grams**. Instead of having only one word in each position of the encoding vector, we can append new terms to the vocabulary formed by the conjunction of two or more words. Returning to the same example as before, with 2-grams, we will also have the terms *i-like*, *like-cars*, *cars-but*, *but-no*, *no-motorbikes*, *like-motorbikes*, *motorbikes-but* and *no-cars* in our vocabulary, which solves the aforementioned problem.

N-grams are a possible solution, but as can be seen, it greatly increases the size of the vocabulary and therefore the size of the encodings. Nowadays, as with words, there are numerous encoding methods based on neural networks, such as **Doc2Vec** (Le and Mikolov, 2014), **LSTM** (Hochreiter and Schmidhuber, 1997), **BERT**(Devlin et al., 2018), **T5** (Raffel et al., 2020). These methods return an embedding with a reduced size for each of the documents, taking into account, among other factors, the order of the words, which is why they are currently the most effective and widely used techniques.

1.2.2 Images

Images are also one of the most abundant types of data today. Their encoding, as with text, has evolved greatly over time. Initially, before the proliferation of deep neural networks, an expert manually decided which features of the image were the most relevant and after extracting them, use those to represent each image. Some examples of this features can be the Fourier descriptors (Kuhl and Giardina, 1982), the Hu moments (Hu, 1962) or the Haralick texture features (Haralick et al., 1973)

With the advancement of technology and the ability to create deeper and deeper neural networks, **Convolutional Neural Networks (CNN)** arise. This type of network is specifically designed to process images and automatically search for the features considered most relevant to the problem to be solved (also known as deep features). One of the best-known benchmarks in this field is the so-called **ImageNet** (Deng et al., 2009) where the aim is to classify an image among 1000 possible classes.

The most common way to encode images is to use the features extracted by an existing pre-trained CNN (typically on the ImageNet dataset). There are multiple pre-trained CNN architectures such as VGG (Simonyan and Zisserman, 2014), ResNet (He et al., 2016), DenseNet (Huang et al., 2017) or MobileNet (Howard et al., 2019).

1.3   Research questions and objectives

As stated in the previous sections, the main goal of this thesis is to take advantage, in multiple ways, of the unstructured information (mainly text and images) that accompanies many of the datasets used in the creation of Recommender Systems. The use of text or images to improve the results of a Recommender System is nothing new, but we intend to take it further, improving not only the recommendations, but also other elements surrounding these systems.

Given this scenario, the first research question we pose is the following:

RQ1. In which ways can we combine unstructured data and RS?
Are there numerous ways in which unstructured information can be used within a Recommender System? We will answer this question throughout the paper by selecting those options that make the most sense for the particular problem we are trying to solve at any given time.

Currently, two issues are widely explored in the field of Recommender Systems: explainability and transparency. What is being pursued in this case is the creation of systems able to explain to the final user where does a particular recommendation come from and how did the system come up with it. This raises the following research question:

RQ2. How can we explain RS recommendations using unstructured data?
Unstructured information can help us in some way to explain the functioning of our Recommender Systems. Therefore throughout the thesis we will try to use this type of information to answer this question.

Finally, the last research question we would like to answer is the following:

RQ3. How can we recommend items directly from unstructured data?
We already know that it is possible to generate recommendations from unstructured data, but in this case we are interested in exploring new ways of doing so, if possible also complying with the explanability mentioned in the previous research question.

To answer these questions, we set out the following three more specific objectives:

1. **Dataset creation:** Throughout the document we will use, for all experiments, the same dataset. We will create this dataset from scratch starting from a data source that fulfills all the requirements of our main objective. In Part I of the document we will detail everything related to this dataset from its creation to its basic statistics.

2. **Using text:** Once we have created the set, we will start exploring the text as a first approach to unstructured information. In this case, the goal will be to create methods that take advantage of this information to either improve the experience surrounding the recommendation (explainability, transparency...), extract underlying information or simply improve the results of the recommendation.

3. **Using images:** Once the work with texts has been completed, we intend to do similar work with images. Using them to improve systems, extract implicit information and improve transparency and trust towards RS.

1.4 Document structure

The rest of the thesis is divided in four parts: Part I includes all the relevant information about the datasets used in the rest of the document (creation, statistics, availability...), then Parts II and III include all the work done using texts and images respectively. Finally, Part IV include information about the publications related to the works included in the previous parts and the final conclusions of the thesis.

Part I

Dataset

Chapter 2

TripAdvisor

To achieve our main goal, we require a dataset with the necessary information to train a Recommender System. As we mentioned at the end of the Section 1.1.1, traditionally, this type of dataset is formed by a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$ where each $\vec{u} \in \mathcal{U}$ has interacted with some $\vec{it} \in \mathcal{I}$. The interaction can be represented by simple Boolean predicates (has bought, like, ...), or it can also include some kind of information or user evaluation in the form of a numerical score (rating, stars, ...).

We are interested in more complex and information-rich interactions, in particular those with textual reviews and associated images , that is, **unstructured information**, which is the type of data that we intend to take advantage of in this thesis.

We will therefore need a dataset containing this type of information. We have chosen to create it from publicly available data on ***TripAdvisor***[1] travel portal. Within this website we have information available in several categories, mainly hotels, restaurants and points of interest (POIs).

All the categories meet our requirements, but we will focus mainly on **restaurants**, since they have a greater number and variability of cases, and also, but to a lesser extent, on **POIs**. For example, in the restaurant category we can find, for the same item, multiple user photographs of different food dishes, and it is very common to find almost the same photograph in different restaurants. This is because many restaurants serve the same type of food. It is also common to find restaurants offering various gastronomic styles.

---

[1]https://www.tripadvisor.com

In the case of POIs the data behave differently, as all images of an item will be very similar (same object but different viewpoints) and it is not as usual to find similar photos between different points of interest.

Name	Population (in millions)	Continent	Language	Restaurants	POIs
Gijón	0.3	Europe	Spanish	✓	
Barcelona	1.6	Europe	Spanish	✓	✓
Paris	2.1	Europe	French	✓	✓
Madrid	3.2	Europe	Spanish	✓	✓
New York	8.3	America	English	✓	✓
London	8.9	Europe	English	✓	✓

Table 2.1: Selected cities sorted by population.

Based on the criteria of size, population, culture and language, we have selected the cities listed in Table 2.1 to download the data of their restaurants and POIs. Note that we decided not to download the POIs for the city of Gijón due to its reduced number of items.

2.1 Creation procedure: Web scrapping

Once we have decided which categories we want to download, we need some kind of automatism to make this task easier. The easiest way to create this type of set is to use the API provided by the specific company, but in this case, the API[1] is not free for public use. The only option available was the development of a software program capable of entering the web and extracting all the relevant information for us.

This automatic information extraction technique is known as *web scrapping* and can be done in two main ways, the first is by simulating the behavior of a user using a web browser and the second is by directly making the necessary HTTP requests to obtain the required information.

---

[1]`https://tripadvisor-content-api.readme.io/reference/overview`

The first is the easiest to implement as it is more visual and we only have to indicate the URL, which buttons to click and which fields we want to extract. This is usually done using applications such as *Microsoft Power Automate*[1] or *Selenium*[2], which has a version in the form of a library for use in a programming language such as *Python*. This method, despite its simplicity, is much slower, which slows down the creation of large datasets with complex data structures.

The second option is the most difficult to implement, but on the other hand it is the most efficient and will be the one we will use in this case. To carry it out, we have implemented a software program[3] in **Python** language that makes use, among others, of the **PyQuery**[4] library, which will allow us to carry out HTTP requests and web scraping at the same time.

For efficiency, this program splits the data download into four different stages, each of which can be executed in parallel using threads:

1. **Item download:** The first step is to obtain a list with all available items (Restaurants or POIs) and their basic details. In this phase we will obtain, therefore, the **name** of the item, its **identifier** within the website, its **rating** (calculated by TripAdvisor based on user reviews) and most importantly, the **URL** of the page where the details of the specific item can be found. This last element will allow us to run the next step.

2. **Obtain reviews:** Starting from the list of items in the previous step, we are going to enter the *URL* of each one in order to extract the basic details of their reviews. The way the website is implemented, it is not possible to see all the content of each review directly on the item's details page, so this will have to be done in a later step. From this phase we will extract the **identifier**, **title**, **stars** and **URL** of each review of each item. It should be noted that the reviews can be written in different languages, but we have chosen to download those in the native language of the city.

---

[1]`https://powerautomate.microsoft.com/es-es`
[2]`https://www.selenium.dev`
[3]`https://github.com/pablo-pnunez/TAVdownload`
[4]`https://github.com/gawel/pyquery/`

3. **Extend reviews:** With the list and basic details of each review, we can now expand each of them by obtaining the full ***text*** of the review and a list of the ***images*** uploaded by the user. Regarding the user, we will take advantage of this phase to store the ***name*** and ***identifier*** of the author of each review. For the images, we will simply store the ***URL*** to download, in the last phase, the associated file. It should be noted that, due to the limitations of the website, only a maximum of four photographs can be viewed per review (although there are more in some cases), which limits the maximum number of photographs to this number.

4. **Image download:** Finally, the last step is the downloading and storage of images. This phase can be omitted if these files are not going to be used, but in our case we will use them in the future, together with the texts, as sources of unstructured information.

It is worth noting the strong dependence of web scraping with the current version of the website. Any slight change in the website, where one of the elements we use is modified, can cause the code to malfunction or stop working. This makes it necessary to review and update the code every time you want to use it again.

After performing this procedure for the two categories and six cities, we will end up with 11 datasets (omitting the one from Gijón for the POIs category) to work with throughout the thesis. With these data we will be able to create RS based on Collaborative Filtering, due to the knowledge of the interaction between users and items, and also Content Based RS if we use the text and images as additional information.

2.2 Exploratory data analysis

In this section we will perform an exploratory data analysis of the sets in order to obtain relevant statistics to be taken into account throughout our research. We will start by showing the most basic statistics of each set, and then we will extract more specific details related to the field of Recommender Systems.

The simplest statistics for the sets can be seen in the Table 2.2: the number of reviews, users, items and images for each combination of city and set. Notice that, the greater the number of inhabitants (Table 2.1) the greater the number of items. This statement is true in all the cities except Paris, as it is a very touristy place where the number of items is higher than expected in relation to its population.

Dataset	City	Reviews	Users	Items	Images
Restaurants	Gijón	54787	26450	716	19362
	Barcelona	466964	184307	7602	153707
	Paris	1135192	463098	15410	257447
	Madrid	641561	246618	8706	208430
	New York	1008761	430082	10271	234892
	London	2271164	1037845	17976	489064
POIs	Barcelona	323190	140628	767	111235
	Paris	383686	147082	1304	130988
	Madrid	252462	105812	918	66290
	New York	598128	233797	997	183200
	London	632068	281454	1979	193840

Table 2.2: Basic stats for the datasets of both categories. Cities are sorted by population.

It is worth remembering that, when working with datasets that are going to be used to train Recommender Systems, one of the most relevant indicators is the number of reviews per item. Items with few reviews will be difficult to learn and therefore to recommend by an RS given the lack of information available about them.

A similar indicator is the number of reviews per user, which tells us how much information we have about the user. The higher this number, the better for learning personalized recommenders. The values of this indicator are usually low, which presents the problem of the so-called **cold-start** (Schein et al., 2002), where we barely have any historical information about a user to be able to make a personalized recommendation. The value of these indicators (in average) for our sets are reflected in the Table 2.3 together with other relevant information such as the number of images per user and item.

This table mainly highlights the difference between the sets of both categories (Restaurants and POIs). The average number of reviews per item and the average number of images per item is substantially higher in the set of POIs compared to the restaurant set. This would be expected after analyzing Table 2.2, where it can be seen that the number of POIs is much lower than the number of restaurants, which results in the items receiving many more reviews and therefore images.

Figures 2.1 and 2.2 show the percentage of reviews (with respect to the total for the city) of the most popular item (restaurant or POI) of the set.

| Dataset | City | AVG reviews | | AVG images | | AVG items |
		per user	per item	per user	per item	per user
Restaurants	Gijón	2.07	76.52	0.73	27.04	2.00
	Barcelona	2.53	61.43	0.83	20.22	2.45
	Paris	2.43	73.67	0.56	16.71	2.36
	Madrid	2.60	73.69	0.85	23.94	2.52
	New York	2.35	98.21	0.55	22.87	2.30
	London	2.19	126.34	0.47	27.21	2.12
POIs	Barcelona	2.30	421.37	0.79	145.03	2.28
	Paris	2.61	294.24	0.89	100.45	2.59
	Madrid	2.39	275.01	0.63	72.21	2.33
	New York	2.56	599.93	0.78	183.75	2.53
	London	2.25	319.39	0.69	97.95	2.22

Table 2.3: Basic stats of datasets.

Once again the difference in the number of reviews per item stands out in favor of the set of POIs. The most popular POI in London (the smallest in Figure 2.2) outperforms the most popular restaurant in Gijón (the largest in Figure 2.1) by some margin.

Also noteworthy is the following: while in the POIs we find that the larger the population, the less likely there is to be a popular point of interest, in the restaurants this is more unpredictable and irregular. While Gijón and London behave in a similar way to the POIs, the other cities behave in a completely opposite way.

As for the rest of the indicators in the Table 2.3 (those that represent the behaviour per user), we can see a very similar behaviour in all the cities and categories. We have, therefore, two categories where, on average, each user has made about two reviews and barely has one photo, which represents, as previously mentioned, a context of *cold-start*. Note that the number of items visited by the user and the number of reviews is similar but does not coincide, reflecting the fact that some users visit and evaluate the same item more than once.

Finally we want to emphasise that all these datasets were created in 2019 and are publicly available for download in (Pérez-Núñez et al., 2021) and (Alonso et al., 2021) (Restaurants and POIs respectively) since 2021 in order to allow other researchers to reuse and explore them.
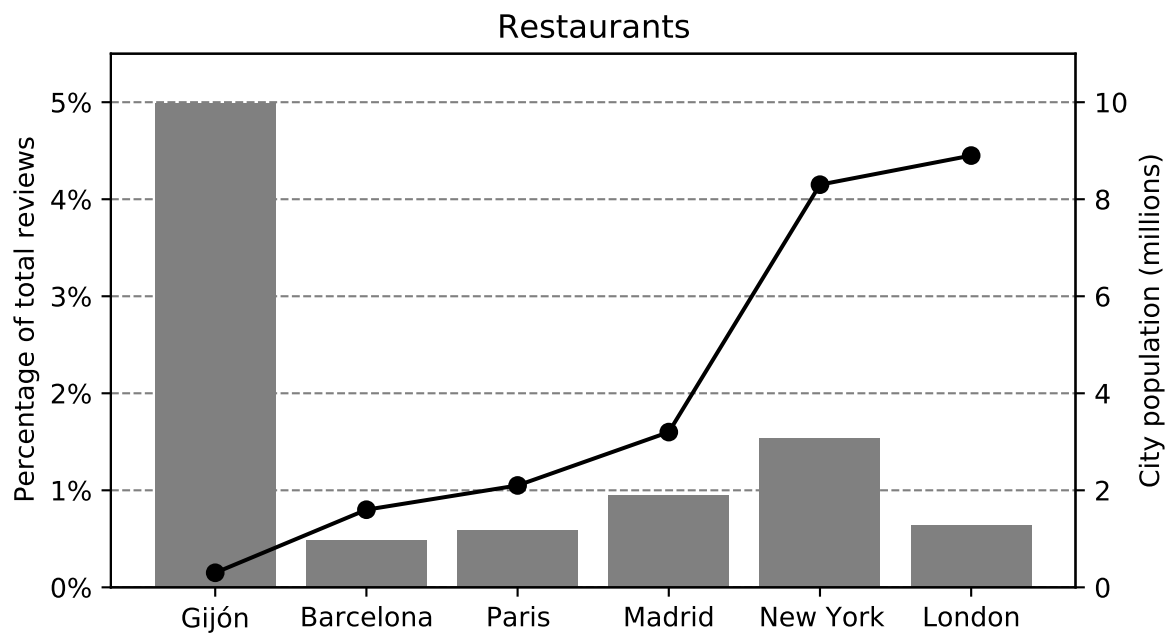
Figure 2.1: Percentage of total reviews of the most popular restaurant in each city (bars) with respect to its population (line).
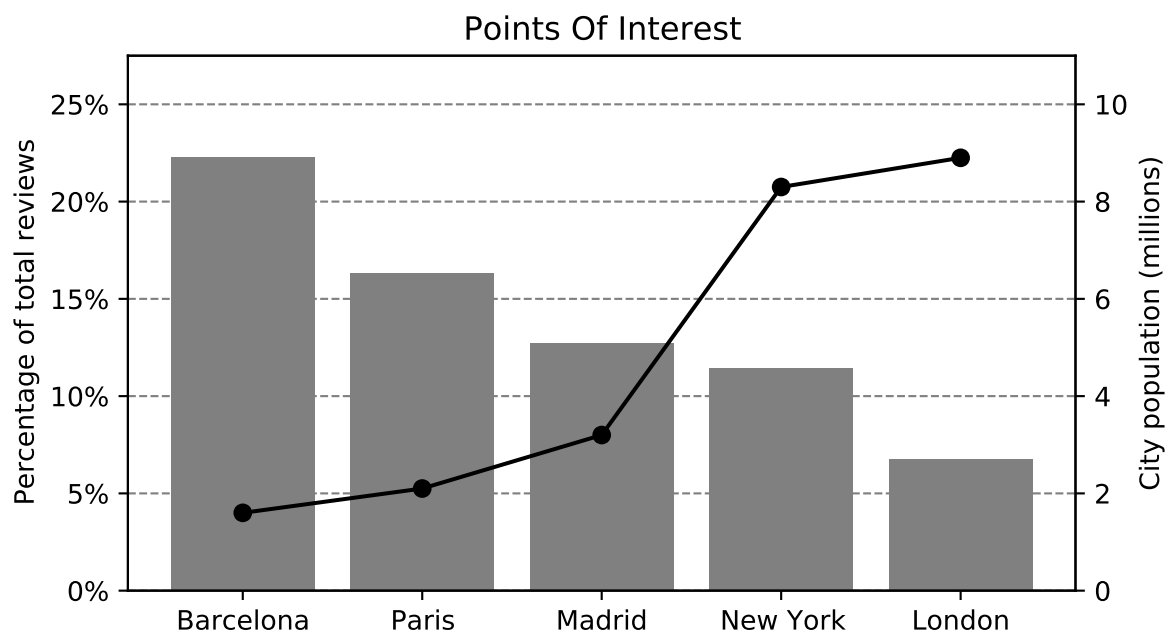
Figure 2.2: Percentage of total reviews of the most popular point of interest in each city (bars) with respect to its population (line). Note the difference in scale of the axes compared to the previous graph (Figure 2.1).

Part II

Using text

As detailed in the previous chapter, our dataset has basically two types of unstructured data: text and images. Given the high complexity associated with each of them and taking into account that each one has a research field of its own (Natural Language Processing and Computer Vision), it has been decided that all the work to be carried out in this thesis will not mix both types of data. The main reason is to understand each of them in detail separately in order to, in the future, be able to combine them when we have more knowledge of their particularities and be able to make decisions in a more justified way.

In the following two parts of the document (Parts II and III) we will present five different works each included in a different chapter. We will start, as the title of this part reflects, by combining the use of Recommender Systems with text. In Chapter 3 we pose a system capable of recommend restaurants from a textual review while keeping transparency and explainability at the same time.

Chapter 3

TReX: Text-based Recommender with eXplanations

Recommender systems have proven their usefulness both for companies and customers. The former increase their sales and the latter get a more satisfying shopping experience. These systems can benefit from the advent of **Explainable Artificial Intelligence (XAI)**, since a well-explained recommendation will be more convincing and may broaden the customer's purchasing options. Many approaches offer justifications for their recommendations based on the similarity (in some sense) between users, past purchases, etc., which require some knowledge of the users.

In this chapter we present a recommender system with explanatory capabilities which is able to deal with the aforementioned *cold-start* problem (Section 2.2), since it does not require any previous knowledge of the user. Our method learns the relationship between the products and some relevant words appearing in the textual reviews written by previous customers for those products. Then, starting from the textual query of a user's request for recommendation, our approach elaborates a list of products and explains each recommendation on the basis of the compatibility between the query's words and the relevant terms for each product.

3.1   Introduction

The layout of the products offered in stores (physical or on-line) has the main purpose of encouraging our consumption, thus increasing the sales. Usually, some items are highlighted because the sellers want to promote them, or because the users (consumers) most likely buy them. But detecting the personal tastes of consumers is not straightforward, so special algorithms called recommender systems (RS) have emerged (Resnick et al., 1994; Ricci et al., 2011). Ideally, recommender systems take into account the interactions between users and items, inducing a predictive model able to make personalized recommendations.

It has been proven that the use of recommender systems increases the consumption of products and services offered by online platforms (Pathak et al., 2010). Clearly, consumers are pleased with the recommendations provided by the RS, thus consuming more. On the other hand, they are also beneficial for the sellers, for obvious reasons: increasing their sales increase their benefits.

However, a lack of confidence in the decisions taken by algorithms has raised lately. This mistrust is due to the bias that some algorithms may present in their behavior (Banker and Khetani, 2019), which can induce the users to make wrong choices.

Explainable Artificial Intelligence has been devised to tackle this issue. It is a recent branch of AI aimed at explaining the output of intelligent algorithms in a human-friendly manner, such that users can understand the reasons behind the algorithms' decisions (Monroe, 2018).

It is becoming an important area of interest since *explainability* is increasingly necessary to meet stakeholder demands. In particular, the General Data Protection Regulation (GDPR) (Voigt and Bussche, 2017) of the European Union demands transparency in systems that take decisions affecting people, making explanations more needed than ever. Additionally, explanations may help increase the trust of users in AI algorithms, since people rely not only on their efficacy but also on the degree of understanding of the process they follow.

More specifically, XAI is being applied in the field of recommender systems (Burke et al., 2021; Zhang and Chen, 2018) to provide convincing recommendations. The benefits of these explanations are, mainly, the following (Tintarev and Masthoff, 2007):

- *Transparency*: users can understand how the recommender works.

- *Scrutability*: wrong recommendations can be analyzed, and the model results can be tuned by human conscious intervention.

- *Trust*: the reasons for a recommendation are explained to the users.

- *Persuasiveness*: explanations can emphasize those aspects of the product in which the user is really interested.

- *Effectiveness* and *efficiency*: users can make good choices quickly when the reasons for the recommendations are known.

- *Satisfaction*: users are pleased with the recommended items, saving time in their choices.

In this chapter we present an approach to build a recommender system capable of explaining its recommendations to a user thanks to the textual reviews written by other users. In order to train the model, we will take advantage of some descriptive terms used by the customers in their reviews. The model will be trained to find out the relationship between the textual reviews, represented using a BoW encoding (see Section 1.2.1), and the items they belong to. The BoW will be carried out considering only the mentioned descriptive terms, so we will have to preprocess the texts in order to filter out the irrelevant words. Once trained, the model will be able to recommend the most adequate items with respect to the terms included in a user query, as well as to elaborate an explanation based on those terms.

One important advantage of our approach is that we can apply our recommender system in different countries and languages, without the need to make changes to the system, as we will explain later.

Additionally, our approach does not require to have any previous knowledge of a user, given that we are not going to handle user profiles. Instead, our aim is to recommend items starting from a simple textual request in natural language. Therefore, one remarkable advantage of this approach is that our system is able to deal with the aforementioned *cold-start* problem. This can be particularly useful in contexts where items have lots of assessments/reviews, but each user reviewed only a very small number of items.

The rest of the chapter is organized as follows: in the next section (Section 3.2) we revise and comment some previous works related to our approach.

Then, in Section 3.3 we detail our proposed method and, hereafter, we describe an experimental setting in Section 3.4, where we introduce the datasets used and discuss the obtained results. Finally, the conclusions extracted from the work can be found in Section 3.5.

3.2 Related work

Recommender systems can be classified in two main classes attending to the kind of explanations they offer: they can show how the recommendation was elaborated, or they can justify it (Jia et al., 2020; Tintarev and Masthoff, 2012). The former are more transparent, allowing the users to understand the causes for an item to be recommended, while the justifications offered by the latter are typically supported by a neighborhood relationship between users and items. Transparent recommenders can be critically analyzed (scrutinized) by the users (Pu et al., 2012), so that they can make a better use of these systems, thus obtaining an improvement in their suggestions.

Zhang and Chen (2018) collected a comprehensive bibliography on explainable recommender systems, although there are not many works which use the text of the reviews. Almahairi et al. (2015) presented one of the first works incorporating textual reviews to improve recommendations (although not to explain them) using a *deep learning* approach, more specifically, recurrent neural networks.

The same kind of neural networks were used in the work of Costa et al. (2018), which was aimed at generating reasonably well-formed explanatory sentences together with the expected opinion of the user. In this sense, Bartoli et al. (2016) warn about the risk posed by automatic generation of reviews, since they can manipulate the users' opinions.

Dias et al. (2017) presented a more similar approach to ours, in which they create an explainable text-based recommendation system able operate in *cold-start* situations. The authors proposed a variation of the *word2vec* (see Section 1.2.1)) method in order to learn a common embedding both for the words of the reviews and the concepts (users and the items). However, their approach requires retraining the network for every new user, and the complexity to obtain explanations is not negligible, given that they have to compute the similarity between the user embedding and the rest of embeddings.

In this regard, it is worth noting the importance of the encoding method used to represent textual information. Let us briefly recall some of the available encoding methods described in Section 1.2.1.

Prior to the advent of deep learning in natural language processing, the most effective and widely used method for text encoding was the BoW. Let us recall This method consists of representing a text as the sum of the one-hot (see Section 1.2.1) vectors of the terms (or words) they are composed of, taken from a previously defined vocabulary. In other words, a text is represented as a vector with the length of the vocabulary, having values greater than 0 only in the corresponding positions of the terms present in the text.

This approach to represent texts has been refined using techniques such as TF-IDF or the use of n-grams, so it is still a widely used method that yields results comparable to those of other more sophisticated methods (Zhao and Mao, 2017). Indeed, there is no significant difference in some applications between using bigrams as terms in BoW, or word embeddings (Shao et al., 2018).

3.3 Formal Framework

In this section we explain our proposal for the construction of a *Text-based Recommender with eXplanations (TReX)*.

In the context of recommender systems we usually start from a set of users, $\mathcal{U}$, and a set of items or products, $\mathcal{P}$. These data are collected in a dataset, $\mathcal{D}$, such that there is a record for every interaction between any user $u \in \mathcal{U}$ with any product $p \in \mathcal{P}$, in which the user expressed his/her satisfaction by means of a score, together with a textual review, r, explaining the reasons for that assessment.

The scores are useless in our approach, as we will explain below, so we will take into account only the textual reviews. Thus, our dataset can be denoted as a set of triples:

$$(u, p, r) \in \mathcal{D}. \tag{3.1}$$

In particular, we want to focus in those contexts where most of the users have very few interactions with different products, thus hindering the construction of personalized recommenders. Instead of taking advantage of the (almost non-existent) consumption history, our approach will recommend products whose reviews are somewhat related to the user's query for recommendation. Moreover, this relationship is also used to yield explanations.
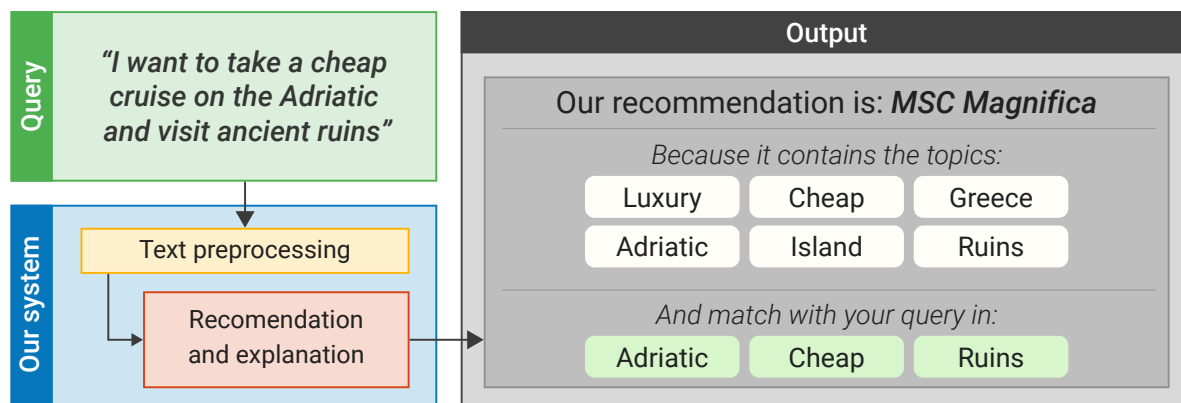
Figure 3.1: Workflow for recommending cruises using our approach. The input to the recommender system is a textual query in natural language. After some processing, it will eventually recommend *MSC Magnifica*, and explains the recommendation on the basis of the relationship with some terms included both in the query and in the reviews provided by other users for this product.

Our method, unlike the work of Costa et al. (2018), does not seek to generate sentences, which can sometimes result somewhat artificial, but to explain the recommendation based on the occurrence of some relevant terms in the user's query that are also relevant to the recommended products, as stated by other users in their reviews.

Let's take, for instance, the case of sea cruises: each cruise may have a lot of reviews, but it is unusual for a user to enjoy many cruises. In this hypothetical context, our system will operate as depicted in Figure 3.1.

In order to make this possible, our model must learn the relationship between the words in the review and the product they refer to. In other words, we will train the model to predict to which product belongs each review. The rationale behind this approach is to consider users' queries (asking for a recommendation) as a kind of *a priori* review; that is, we address the problem as: what would we recommend so that the user will eventually write a review with the terms included in his/her request?

Obviously, what we express in a request and in a review is rather different from a semantic point of view. Our hypothesis is that the difference is mainly due to the verbs used, but both reviews and requests are similar regarding the things they mention and their characteristics, that is, the *nouns* and *adjectives* used.
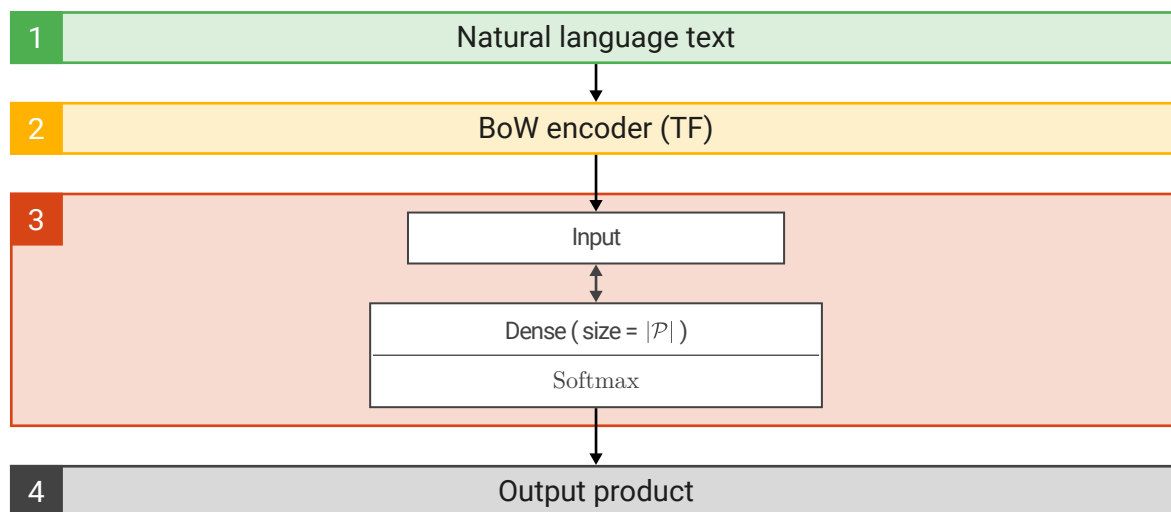
Figure 3.2: Architecture of the proposed system. Textual reviews, 1, encoded using BoW, 2, are processed using 3, a fully connected dense layer with a linear activation function, and with as many output elements as products, followed by a softmax layer. The output, 4, is a vector with the probabilities of belonging to each product.

3.3.1 Selection of relevant terms

The construction of the set of relevant terms is a key point to allow the model both to infer a good recommendation as well as to provide good explanations. Our proposal for this task consists of applying Part-of-Speech (PoS) tagging techniques (Schmid, 1994) to get rid of all words except nouns and adjectives. Then, we will consider as relevant terms a percentage of the most used nouns and adjectives in all the reviews. This percentage may be context-dependent so we recommend to determine it experimentally, in order to achieve an adequate trade-off between performance and explicability.

Once we have the set of relevant terms we can encode the textual reviews using BoW, in order to feed the model during training as well as during its operation, once trained. One important advantage of this approach is that we can apply our recommender system in different countries and languages, as we will show in Section 3.4.

For this purpose, we take advantage of the fact that computational linguistics provides reliable computer-based PoS tagging techniques for multiple languages.

3.3.2 Computing the recommendation

The proposed model is a classifier based on a multinomial logistic regression, as depicted in Figure 3.2. The model is trained with textual reviews, previously encoded using a bag of words approach, as explained above. Each review is encoded in a vector $\vec{t}$, where each component records the number of occurrences of the corresponding term in the text. This vector is then normalized using the L1 norm, thus gathering the probability of occurrence of each term in the textual review. This is also known as *term frequency (TF)* encoding.

The normalized BoW vector, $\vec{t}_{L1}$, is used as input to a fully connected layer with a linear activation function. The output of the layer is then transformed into a vector of probabilities using a *softmax* layer. Thus, the classifier outputs a vector whose dimension must coincide with the number of products, $|\mathcal{P}|$, that can be recommended:

$$\hat{\vec{y}} = \mathrm{softmax}(W\vec{t}_{L1} + \vec{b}) \tag{3.2}$$

where $\hat{\vec{y}}_p = \Pr(p|\vec{t})$, i.e., the p-th component of $\hat{\vec{y}}$ is the estimated probability of the text to be a review of the p-th product. We pose a multiclass learning task, where we learn the matrix W and the bias $\vec{b}$, and where the loss function to be optimized is the usual categorical cross entropy:

$$\mathcal{L}(\vec{y}, \hat{\vec{y}}) = -\sum_{p=1}^{|\mathcal{P}|} \vec{y}_p \cdot \log(\hat{\vec{y}}_p) \tag{3.3}$$

where $\vec{y}$ is the one-hot vector with the ground truth.

The benefits of such a simple model are two-fold: we expect to obtain a good generalization with no overfitting, and we can easily draw out explanations to justify the results of the classifier (the recommendation).

3.3.3 Explaining the recommendation

We have just seen that the output probabilities (3.2) of the products depend on the parameters learned, that is, W and $\vec{b}$. Taking into account that the bias of each product p, i.e., $\vec{b}_p$, is constant, we can focus on $W_{p,\cdot}$ (row p) to analyze the relevance of every linguistic term for the recommendation of the p-th product.

Let's detail how *TReX* proceeds with a toy example depicted in Figure 3.3, in the context of cruise recommendation previously introduced. For the sake of simplicity, we are going to consider only six relevant terms (*"cheap"*, *"island"*, *"adriatic"*, *"ruins"*, *"luxury"*, and *"greece"*), and only four different cruises that can be recommended (MSC Orchestra, MSC Magnifica, Silver Shadow and MSC Armonia).

In this setting, and starting from a query written in natural language (step 1), we filter out irrelevant words to keep only relevant terms, which are used to create the vector $\vec{t}$ containing the number of occurrences of each term (BoW encoding). In the given query we have only 3 out of 6 relevant words, appearing only once each one; thus, the corresponding components of $\vec{t}$ are 1's. This vector is then normalized using the L1 norm, yielding $\vec{t}_{L1}$ (step 2). The query, encoded as an L1-normalized 6-dimensional vector $\vec{t}_{L1}$, is then used as input to a multinomial logistic regression (step 3), which estimates the probability of the input text to be related to each one of the possible products.

In the event that the training process yielded the matrix W shown in Figure 3.3, the logistic regression (we have omitted the intercept vector, $\vec{b}$, for clarity) will yield an output vector with probabilities indicating that the most recommendable cruise is MSC Magnifica.

An interesting side-effect of this approach is that we can justify or *explain the recommendation* on the basis of the relevant terms found both in the user request and in the reviews of the recommended products. Having into account the parameters learned during the training of the model, i.e., the matrix W, we can finally provide a justification for the recommendation (step 4).

Thus, looking at the second row of the weight matrix, $W_{2,\cdot}$, and comparing it with the input vector, t, we observe that the relevant terms in the query are also very relevant for the corresponding product, MSC Magnifica, due to their sign (positive) and their magnitude (the highest in the row).

Therefore, this information can be used to provide an explanation in the following sense: *"MSC Magnifica is recommended because its reviews relate (with the highest probability) this product to* cheap, adriatic *and* ruins, *as requested by the user"*.
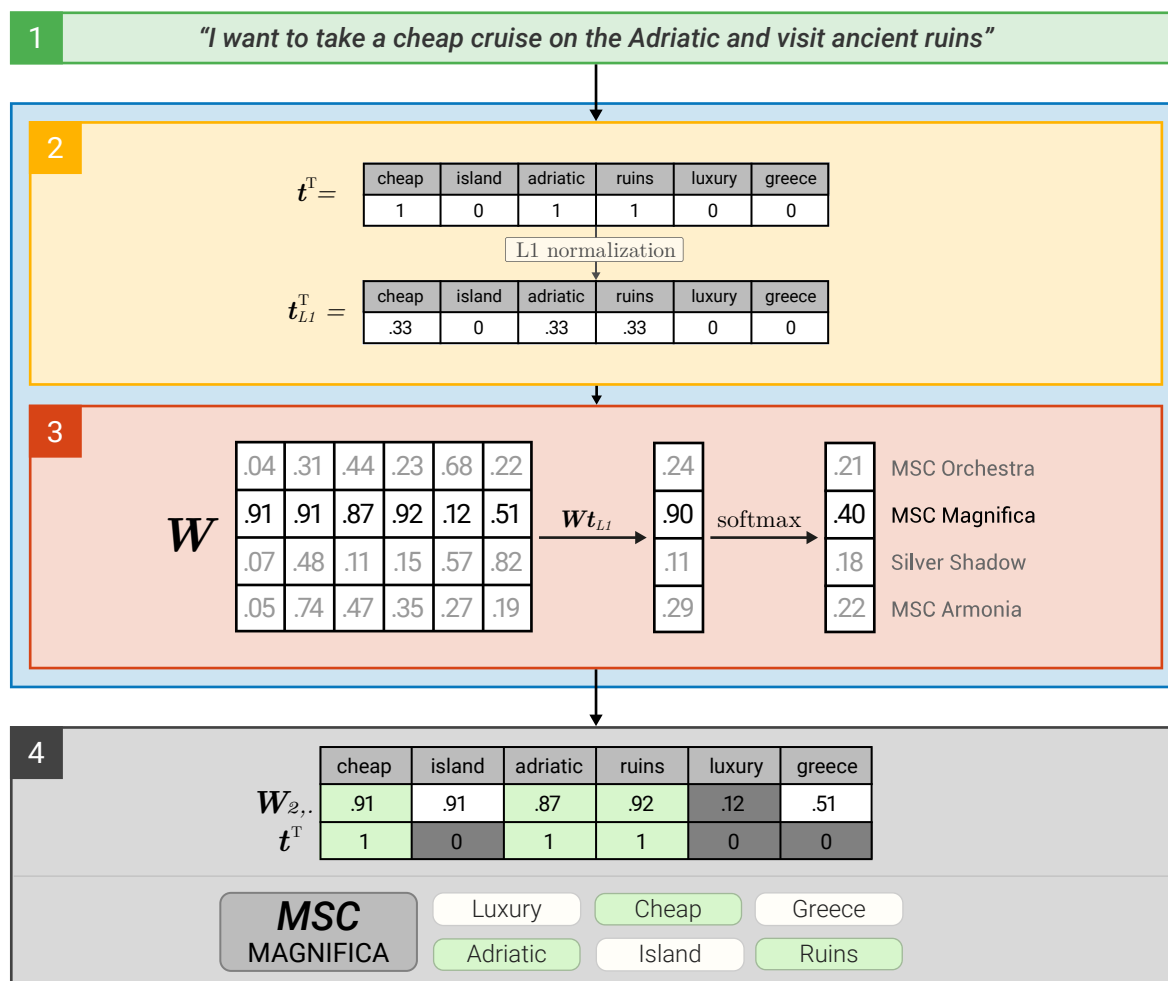
Figure 3.3: Detail of the recommendation and explanation tasks in a toy example. For simplicity, the bias vector b was omitted. The user's query, ①, is encoded in an L1-normalized vector, ②, which is then used to predict the output probabilities, ③. Setting side by side the probabilities and the terms in the user's query we can elaborate an explanation, ④.

3.4 Experiments

We carried out some experiments in order to validate our approach. For this we will use the restaurant datasets of TripAdvisor, which also include text. Restaurant recommendation is a task where the average number of reviews per product is much higher than per user. Therefore, this task fits the type of problems that our system aims to address.

City	Gijón	Barcelona	Madrid	Paris	New York
Abbreviation	GJN	BCN	MDR	PRS	NYC
Population	300K	1.6M	3.2M	2.1M	8.3M
Language	Spanish	Spanish	Spanish	French	English
# Reviews	39889	320236	474688	807535	826117
# Restaurants	148	1325	1634	3423	1988
Avg revs/rest	269.5	241.7	290.5	235.9	415.6
Popularity	6.8%	0.7%	1.3%	0.8%	1.9%

Table 3.1: Main characteristics of the five datasets after filtering out reviews with more than 2000 characters and restaurants with less than 100 reviews. It should be noted that, for each city, we only included reviews in its native language. *Popularity* represents the percentage of total reviews that have been written for the most popular restaurant.

In the rest of this section we describe how we pre-process the datasets, as well as the implementation details of our approach. Finally we present and discuss the experimental results, also illustrating how our proposal could be eventually used to build a restaurant recommender application with explanation capabilities.

3.4.1 Datasets

As previously mentioned, for this work we will use the TripAdvisor restaurant dataset, as it contains user reviews in text form. It is worth remembering that this data contains several cities, which were selected taking into account several factors, such as population, number of restaurants, geographical location and language. The choice of cities in different countries is important to prove that our approach is robust regarding the language used in the reviews.

On the other hand, selecting cities with different values of population, ranging from less than half a million up to more than eight million inhabitants, let us analyze the behavior of our approach in a variety of sizes of the datasets (the larger the population, the larger the number of restaurants and reviews).

Table 3.1 summarizes the most important characteristics of the datasets after filtering out examples whose restaurant had less than 100 reviews. Worth of mention is the number of restaurants in each dataset, which determines the size of the output layer of its corresponding model. Another noticeable characteristic is the average number of reviews per restaurant (Avg revs/rest), which is very similar for all the cities except for New York City, whose value is almost double than for the others.

Finally, the value shown as *Popularity* corresponds to the percentage of reviews of the most popular (i.e., reviewed) restaurant. The city of Gijón has the highest value in this characteristic with a large difference with respect to the rest of the cities. Our guess is that this is due to the fact that Gijón is the smallest city, the one with the smallest restaurant offer, and this fact could lead to a concentration of popularity in a few specific restaurants.

3.4.2 Restaurant recommendation: implementation and results

We devised an experimental setting based on the data downloaded from TripAdvisor in order to assess the performance of our approach when recommending restaurants. The textual reviews were preprocessed as follows: case normalization (all letters in lower case), elimination of punctuation symbols, lemmatization (transforming all the variations and inflected forms of words into its common lemma), and elimination of accent marks and numbers. We also removed long reviews, with more than 2000 characters, and restaurants with less than 100 textual reviews.

After the preprocessing, we reserved 10% of the examples in each dataset (3.1) for testing purposes, 80% for training and the remaining 10% for validation. The validation sets were used to seek for adequate hyperparameters for each model/dataset. The splits were made randomly, stratified by restaurant, in order to maintain the distribution as well as to ensure that any restaurant in the test/validation sets could be eventually recommended (unknown products cannot be recommended).

The implementation of our recommender, *TReX*, was made in Python using TensorFlow (Abadi et al., 2016). The text preprocessing techniques mentioned previously, including lemmatization and PoS tagging (needed to the BoW encoding), were applied using the spaCy library (Honnibal et al., 2020).

The optimization during the training was carried out with the Adam algorithm (Kingma and Ba, 2014) and an *early stopping* strategy. The hyperparameters of each model, i.e., the learning rate and batch size, were chosen ad hoc for each city after a grid-search on the corresponding validation subset.

On the other hand, the vocabulary of relevant terms used for BoW encoding was made up with 10% of the most used nouns and adjectives found in the textual reviews of each dataset/city. The value for this hypeparameter was chosen after a series of tests on the validation sets of several cities.

We compared the performance of *TReX* with a more complex classifier, whose architecture was based on a **Long-Short Term Memory (LSTM)** network (Hochreiter and Schmidhuber, 1997), much more powerful for **Natural Languaje Processing (NLP)** tasks such as the one at hand. The architecture of this classifier, labeled as LSTM2rst, consisted of an LSTM layer with 256 elements, connected to an output layer with the required number of elements (as many as restaurants in the corresponding dataset) and a softmax layer, in order to obtain a vector of probabilities.

The input encoding used for LSTM2rst was also more sophisticated: we used a *word2vec* (Mikolov et al., 2013) approach instead of the BoW encoding used by *TReX*. Caselles-Dupré et al. (2018) claim that recommender systems using *word2vec* encodings obtained from the texts of the task at hand yield better performance than those obtained from generic texts, so we trained our *word2vec* encoders with the (preprocessed) textual reviews, using a window size of 5 words and 300-dimensional output vectors.

The reason for comparing *TReX* with LSTM2rst, a more complex approach with a richer input representation, is to test if it is worth the trade-off between the possible penalty in performance of *TReX* and its ability to offer explanations for its recommendations.

Dataset	Baseline			TReX			LSTM2rst		
	P@1	P@5	P@10	P@1	P@5	P@10	P@1	P@5	P@10
GJN	6.8	17.6	25.7	**35.9**	**60.2**	**70.5**	34.8	58.4	69.6
BCN	0.7	3.0	4.8	28.1	46.5	54.5	**29.0**	**49.1**	**57.7**
MDR	1.3	4.4	6.7	**34.6**	53.8	61.0	34.4	**54.9**	**63.0**
PRS	0.8	2.1	3.1	23.7	38.5	44.9	**27.4**	**43.3**	**50.3**
NYC	1.9	5.9	8.8	38.8	57.6	64.8	**40.4**	**58.9**	**66.5**

Table 3.2: Precision@{1,5,10} in percentage obtained by the different systems. In bold are the best results for every city (rows) in every measure.

Table 3.2 displays the scores of Precision (@1, @5 and @10) obtained by *TReX* and LSTM2rst in the test sets of each city. We also included, as a *baseline* reference, the precision obtained when always recommending the most popular restaurants. This baseline recommender is clearly the worst, but the difference in precision between the smallest city (Gijón) and the rest is noteworthy.

In fact, we already noticed some peculiarities regarding the popularity when presenting the characteristics of the datasets in Table 3.1, where we indicated the percentage of reviews of the most popular restaurant of every city.

As we mentioned previously, we guess that popularity is more important in places with a limited offer of restaurants. Notice also that the scores in Precision@1 coincide with the percentage of reviews of the most popular restaurant in every city due to the stratified split of data.

On the other hand, the performance of LSTM2rst is, in general, better than that of *TReX*, as expected. The *word2vec* embedding learned from the data, together with a more sophisticated NLP approach used on the textual reviews by means of the LSTM network, make LSTM2rst more accurate than *TReX*.

However, the scores obtained by *TReX* are only slightly worse, despite using only 10% of the words in the textual reviews as input information. Moreover, it is straightforward to elaborate an explanation for the recommendations, as we already introduced in Section 3.3.3. We illustrate this ability in the following with a real example.

3.4.3 Explanation of the recommendation

Many recommender systems can only justify their recommendations by some kind of similarity (neighborhood) between users, products and reviews, as we mentioned in Section 3.2. Using deep learning based models, such as LSTM networks, allows us to obtain more complex and more accurate encoding approaches to project those users and products in a space where we can compute such similarity. But the justification of the recommendation based on these sort of distance measures is barely understandable by the users of the recommender system, who cannot scrutinize the recommendation.

Our proposal, however, learns the relevance of the most important linguistic terms which characterize the products, and this relevance, gathered in the matrix W of (3.2), is then used to elaborate explanations in the same linguistic terms appearing in the users' queries.

Figure 3.4 depicts a mock-up of a restaurant recommender application based on *TReX*. The application will be able to operate in different languages, justifying its recommendations, following the idea explained in Section 3.4.3 with the synthetic example of cruises.

For instance, let's look at Figure 3.4b, which exemplifies a request of recommendation in New York City; the query *Where can I eat the typical pastrami sandwich?* triggers a suggestion of the top n (4 in the mock-up) restaurants with the highest probability to be related to the relevant terms of the query.

Pointing to a given restaurant, the system can highlight some words of the query with an intensity proportional to their relevance for the recommendation, i.e., proportional to their weights in the corresponding row of W. Additionally, the user can see a word cloud built up depending on all the weights of the row which corresponds to the selected restaurant.

In the example of the figure we focused on the restaurant *Pastrami Queen*, so we can see that the terms *"pastrami"*, *"sandwich"* and, more importantly, *typical*, are relevant to describe this restaurant. The word cloud associated to this restaurant shows that other relevant terms are, for instance, *"Knish"* and *"Kosher"*, giving a clue of the food which is frequently mentioned in its reviews, and helping the user to take a better decision for choosing one of the recommended restaurants.

It should be noted that all the results shown in the mock-up depicted in Figure 3.4 were actually obtained from the output provided by *TReX* on the datasets of Gijón, New York City and Paris, respectively.

3.5   Conclusions

In this work we have shown an approach to make a recommender system able to provide explanations based only on the textual reviews of products. From these reviews we have devised a transparent recommendation system that learns the relevance of the terms that define the most important aspects of the products. These terms and their relevance are combined to provide explanations in a user-friendly manner, so that users can understand how the system has built the recommendation. Looking at the most relevant terms that define a product, the system also helps users to make the right decision.

The lack of complexity of our recommender does not hinder significantly its performance, as we have shown in a comparison with a more complex LSTM-based approach on datasets of different cities and sizes.

Additionally, the presented approach does not require any previous information about the users, so it is well-suited for *cold-start* scenarios. Our model is induced just from textual reviews of products.

Finally, we would like to highlight that our system can be directly applied to datasets in different languages without any change, provided that the language supports PoS tagging.

(a) Recommendation sample for Gijón.

(b) Recommendation sample for New York.

(c) Recommendation sample for Paris.

Figure 3.4: Mock-up of a recommender application with explanations. The results shown were actually obtained from true recommendations and explanations of our *TReX* in the three datasets/cities indicated.

Part III

Using images

From this point we are going to work with the other kind of unstructured data, the images. We are going to combine the use of Recommender Systems with images in four different proposals. In Chapter 4 we will pose a system capable of predicting the photo a user would take to explain recommendations, then in Chapter 5, a new way of creating embeddings to encode the images of a recommender system is presented, the proposal in Chapter 6 uses these embeddings to summarize clusters of users in an image, and finally, Chapter 7 presents a system capable of recommending restaurants based on the content of a given image.

Chapter 4

ELVis: *Explaining Likings Visually*

Justifying or explaining the predictions of a complex model, such as a Recommender System, is a topic that has become a priority for users and companies in recent years. Therefore, in the first work of this part we will explore the idea of learning personalised explanations for each user as if they were recommendations.

Nowadays, there are numerous platforms (such as TripAdvisor or Amazon) where users are allowed not only to leave reviews about a service or a product, but also to attach some photos to reinforce or justify their opinion.

For this reason, our main objective in this chapter is to predict which picture a user would take of a given item, because this predicted picture will be the best argument to convince him of the qualities of the product. With it we can explain the results of a Recommender System and therefore increase its credibility towards the user.

Once the model capable of predicting the most attractive image for a given user has been trained, we can estimate its distribution, so that a company can find out which aspects of its products stand out to customers.

4.1 Introduction

As mentioned in the previous work (Chapter 3), XAI is emerging as an important area of research interest since the GDPR demands transparency in systems that take decisions affecting people. As a good side-effect, explanations may help increase the trust of users in AI algorithms, since people rely not only on their efficacy but also on the degree of understanding of the process they follow. As a positive side-effect, explanations might help consumers have more faith in AI algorithms because they depend not just on their effectiveness but also on how well they comprehend the process they utilize.

One of the systems that suffers most from this lack of explanation are the aforementioned Recommendation Systems (Section 1.1), which offer recommendations of items to users and where *explainability* takes on special importance. There are various attempts to create explainable recommendation systems in the literature, including visualisations in the form of a tree, as in the case of Hernando et al. (2013) or the use of labels or tags of the recommended items Zheng et al. (2019).

In our case we want to exploit the popular saying *A picture is worth a thousand words* by accompanying each restaurant recommendation with an image that explains it. This image has to show the most relevant characteristics of the item for the user and has to justify why the recommendation has been made.

With this goal in mind, we present a method capable of learning to make personalised explanations from images called *ELVis* (*Explaining Likings Visually*). Our proposal does not aim to predict whether a user will like or dislike an item, this can be done by means of a *conventional* RS, our goal is, as can be seen in the Figure 4.1, to recommend pictures in order to create a personalised *cover* for each user of each of the recommended items, making the recommended items even more attractive. In order to perform the necessary experimentation, in this case we will make use of the TripAdvisor restaurant dataset previously described in Part I.
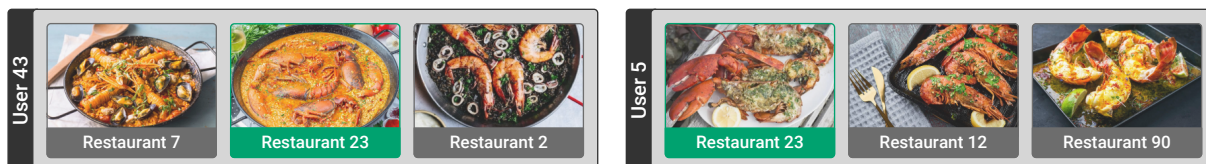
Figure 4.1: Three restaurant recommendations for two different users obtained through a given Recommendation System, the images have been selected using our system. Note how *ELVis* selects different images for the same item (23) depending on the target user.

From a formal point of view, we will predict the photo that the user would upload along with their review. The rationale behind this proposal is that, with their photos, users want to highlight the aspects of the items that most appeal to them (based on their personal tastes). They generally take the photographs to explain or justify to other users the reason for the score they have given the item, so their images are the best reason to convince them.

There is another aspect that we want to remark here, which is the usefulness of this proposal seen from the side of companies. Once we have a model to predict images, we can estimate the distribution of those predicted images.

This distribution can provide very useful information about the aspects of products that most attract the attention of the customers.

The rest of this work is organised as follows. Section 4.2 includes other proposals that deal with images and/or use data from popular online platforms. In Section 4.3 the aim is to set a formal framework for suggesting the picture a user would take of an item. The architecture of the proposed model is detailed in Section 4.4 and then, in Section 4.5 we discuss about the dataset and how it is pre-processed.

Once the model and the dataset are known, we can talk about the experiments to be carried out (Section 4.6) and the ways in which each of them will be evaluated (Section 4.7). Finally, the results obtained are presented in Section 4.8 and the conclusions drawn are reflected in Section 4.9.

4.2 Related work

This section includes a brief review of some approaches that use visual information to provide recommendations. On the one hand, we highlight some methods for video or movie recommendations that select the most appealing thumbnail or cover, respectively, some of which are based on popular platforms such as YouTube or Netflix.

On the other hand, we stand out some approaches for restaurant recommendations, with an special mention to another popular platform such as TripAdvisor. Finally, we present some state-of-art methods that use photos to provide recommendations, and the rationale of our approach compared to them.

4.2.1 YouTube

In the context of videos, YouTube is the largest RS in the industrial world. YouTube recommendations are described in (Covington et al., 2016), including details about its design and maintenance.

The complete system is a deep learning approach composed by two neural networks: one to generate a list of video candidates based on the users' activity history, and the other one to rank them in a personalised way. Note that in the second stage the authors proposed a deep collaborative filtering model where both videos and users are represented by means of rich feature sets of descriptors, instead of using matrix factorization (Koren et al., 2009), as in previous approaches.

Regarding video retrieval, Liu et al. (Liu et al., 2015) proposed a multi-task approach to automatically select query-dependent video thumbnails. They are selected from video frames, based on visual and side information. The authors extract visual information by means of a CNN (see Section 1.2.2) architecture, and side information from the query by a word embedding model (*GloVe*, see Section 1.2.1). Next, the two vector representations are mapped into a latent semantic space, in which the relevance of thumbnails can be estimated for the final selection.

4.2.2 Netflix

Netflix platform is endowed with a set of recommender tools. An overview of the algorithms used together with their business value is presented in (Gomez-Uribe and Hunt, 2016). Among them, we would like to highlight how Netflix provides personalized covers of the available contents (Netflix, 2016). The idea lies in exploiting movies' and users' information to select the picture that best represents a movie across all users.

Given the high diversity in users' preferences, Amat et al. (Amat et al., 2018) proposed to personalize the best picture per movie for each individual user. Their target was how to convince users to watch a movie, by showing them some visual evidence that supports the recommendation. The approach is based on contextual bandits and online machine learning, and it selects the images and makes the personalized recommendations as part of the same process.

Other works can be found in the literature applied to movie recommendations, which were evaluated on widely used datasets such as MovieLens and Netflix Prize. For example, Ortega et al. (Ortega et al., 2016) proposed a recommender system based on matrix factorization and collaborative filter to provide movie recommendations to groups of users.

4.2.3 TripAdvisor

TripAdvisor, as previously commented, is a very popular (not personalized) recommendation platform of the hospitality sector, in which users upload their opinions about restaurants, including ratings, text reviews and photos. In this context, deep learning networks have been used to improve users' experience by showing them the most appealing pictures, but not in a personalized manner (Amis, 2017).

The authors gathered the training data from the platform, and then manually selected thousands of *preference judgments*; that is, pairs to learn a ranking of photos for given a property. Regarding the model architecture, they used a *siamese network* built on the top of the ResNet50 (He et al., 2016), thus allowing to learn on pairs of photos.

Chu and Tsai (Chu and Tsai, 2017) designed a study in which restaurant attributes and users preferences are both represented by visual features, allowing to link content-based and collaborative filtering. The process with images uses standard embedding CNN procedures with additional ad hoc features. In order to deal with several photos, the authors use averaging or maximum aggregations.

Focusing on restaurant recommenders that do not use images, several works found in the literature are evaluated with TripAdvisor data. Some recent examples are a decision support model (Zhang et al., 2017) that recommends restaurants to tourists making use of social information, including online reviews and social relationships; and a recommender system (Zhang et al., 2018) that considers group correlations for both users and restaurants.

Finally, it is worth noting that there are also hotel recommenders that use TripAdvisor. For example, Nilash et al. (Nilashi et al., 2018) presented a fuzzy-based approach that uses multi-criteria ratings extracted from online reviews to provide hotel recommendations. Another example is the one proposed by Liu et al. (Liu et al., 2019) that recommends hotels taking into account the time in which the reviews were made by users and, thus, possible changes in their behaviors and tastes.

4.2.4 Photos and recommendations

To our knowledge, the first attempt to use photos to provide recommendations is the content-based RS presented in (He and McAuley, 2016), called VBPR (Visual Bayesian Personalized Ranking). It is a factorization system in which the description of the restaurants (items) are the features learned by a CNN from one single image of the item.

Kang et al. (Kang et al., 2017) seek to extend the previous contribution in (He and McAuley, 2016), showing that recommendation performance can be significantly improved by learning *fashion aware* image representations directly, i.e., by training the image representation (from the pixel level) and the RS jointly.

In a more recent work, Tang et al. (Tang et al., 2019) focused their attention on the lack of robustness that affects to VBPR. In this sense, the authors proposed an approach based on adversarial learning to obtain a robust multimedia recommender, which was evaluated on Pinterest and Amazon datasets.

4.2.5 Rationale of the approach

Other recent works have explored the use of images in the context of RS; however, to the best of our knowledge, there are no attempts, in the literature or commercially, able to predict the photos that the user would take with the main aim of providing explainable recommendations. Our target is to provide a personalised explanation drawn from the core of the RS that learns the interest of users in items. For this purpose, users' photos are employed.

Regarding other systems that recommend images, such as the one previously described (Tang et al., 2019), there are two main differences that deserve to be highlighted. First, images handled in our proposal are photos uploaded by the users of a platform to show their opinion about an item, instead of being uploaded by its responsible in commercial terms; thus, we have to face a challenge in terms of the quality of the images and the variety of objects and information displayed in them. Secondly, photos in our case can be organised by the user that uploaded them or by the item they correspond; thus, obtaining a complex structure of images that we have to handle properly. Finally, our objective is to give an explanation, which makes us explore the set of images in a peculiar way.

With respect to restaurant recommenders, users that look for restaurants in popular platforms are interested, not only in the rating given by other users and their comments, but also in their photos, taken mainly of food, restaurant atmosphere and restaurant location. For this reason, some RS found in the literature, such as the work presented in (Chu and Tsai, 2017) previously mentioned, use this kind of information. However, all of them use photos to compute a personalised list of recommendations, while our target is to predict photos as a way to provide personalised explanations.

4.3 Formal framework

At the beginning of Chapter 2, we explained that in a collaborative recommender platform we usually have a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$ interacting to form pairs as shown in the Equation 1.1 or 1.2.

However, in this work we will interpret a set of images taken by users as an element to highlight certain aspects of the items. Formally we will have the following collection of sets:

- photos($\vec{u}$): photos taken by user $\vec{u} \in \mathcal{U}$,

- photos($\vec{it}$): photos of item $\vec{it} \in \mathcal{I}$, and

- photos($\vec{u}, \vec{it}$): photos($\vec{u}$) $\cap$ photos($\vec{it}$).

Our interest is to be able to detect the photos taken by a given user since we understand that they are representative of that user's tastes. Therefore, we assume to have a set of labeled pairs

$$(\vec{u}, \vec{f}) \rightsquigarrow \begin{cases} 0, & \vec{f} \notin (\vec{u}) \\ 1, & \vec{f} \in (\vec{u}) \end{cases} \tag{4.1}$$

where $\vec{u} \in \mathcal{U}$ denotes a user, $\vec{f} \in (\vec{it})$ is a photo of an item, and the label point out the *authorship* of the photos. Thus, the label will be 0 to indicate that the photo was not taken by user $\vec{u}$ and 1 otherwise.

As usual in RS, we are going to estimate the probabilities for positive labels. The main difference is that, in this work, labels' examples represent the authorship of the photos.

We assume that a user takes photos of an item trying to reflect the most important characteristics that make the user like/dislike the item. Thus, we want to learn a procedure to select the photo of an item $\vec{it}$ that best represents the tastes of a user $\vec{u}$, that is,

$$\vec{f^*} = \operatorname*{argmax}_{\vec{f} \in \text{fotos}(\vec{it})} \Pr(\vec{u}, \vec{f}). \tag{4.2}$$

Therefore, we aim at solving a binary classification task to estimate $\Pr(\vec{u}, \vec{f})$. The rationale is that once we learn to predict the probability of a photo to be taken by a user $\vec{u}$, we can apply that model to select photos taken by other users, but with a high probability to be also taken by $\vec{u}$. This will allow us to show to $\vec{u}$ the most adequate photos of new items, those that the user would have probably taken.

This idea can be generalized to a group of users by defining a compatibility function of a photo, $\vec{f}$, with a set of users $\mathcal{S} \subseteq \mathcal{U}$ according to their tastes by

$$\phi(\mathcal{S}, \vec{f}) = \sum_{\vec{u} \in \mathcal{S}} \Pr(\vec{u}, \vec{f}). \tag{4.3}$$

Therefore, the photo $\vec{f^*}$ that best *explains the tastes of a group $\mathcal{S}$ of users* regarding a given item $\vec{it}$ can be obtained by

$$\vec{f^*} = \operatorname*{argmax}_{\vec{f} \in \text{fotos}(\vec{it})} \phi(\mathcal{S}, \vec{f}). \tag{4.4}$$

4.4 Topology of the network

In this section, we present the network employed to induce the distribution of $\Pr(\vec{u}, \vec{f})$. Taking into account that we have to solve a binary classification task, the optimization of the *binary cross-entropy* loss function was carried out using a model that learns on pairs of users and photos $(\vec{u}, \vec{f})$. Figure 4.2 depicts the architecture of the proposed model (*ELVis*).
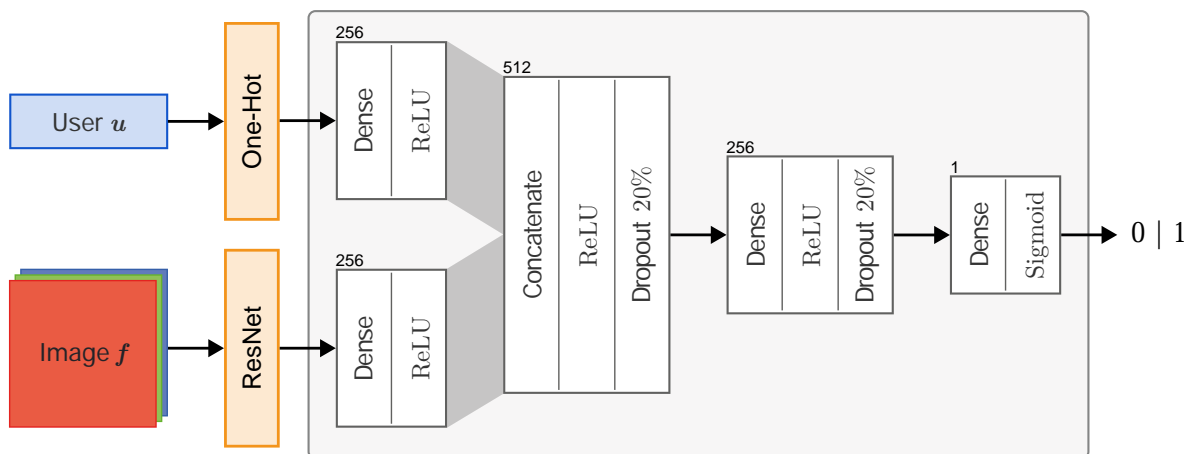
Figure 4.2: Topology of the network employed in *ELVis* to learn $\Pr(\vec{u}, \vec{f})$ from labeled pairs of users and photos $(\vec{u}, \vec{f})$.

The input data of *ELVis* are codified as follows:

- User $\vec{u}$. Users are represented by a *one-hot* codification, and then mapped into a 256-dimensional embedding.

- Image $\vec{f}$. Photos are codified using a Convolutional Neural Network; more specifically, the convolutional base of the Inception-ResNet-v2 model (Szegedy et al., 2017) with weights pre-trained on ImageNet[1]. The embedding provided by the CNN, composed of 1,536 deep features, is next mapped into a vector with 256 elements.

Once the input data are codified as previously described, the two vectors are concatenated and further processed by a sequence of different layers that include Fully Connected (FC), Rectified Linear Unit (ReLU) (Nair and Hinton, 2010), and Dropout (Srivastava et al., 2014). The purpose of these layers is to learn a nonlinear function able to determine if a given photo was taken by a given user. The authorship is given in terms of the joint probability, $\Pr(\vec{u}, \vec{f})$, so finally a sigmoid activation function is used to produce a probability output in the range $[0, 1]$.

We came up with this architecture after numerous empirical tests using other networks with variations in the number and size of the layers.

---

[1]https://keras.io/api/applications/inceptionresnetv2/

TOP	Gijón (338)			Barcelona (3023)			Madrid (4578)		
	RND	CNT	*ELVis*	RND	CNT	*ELVis*	RND	CNT	*ELVis*
1	4.3%	2.7%	8.9%	4.0%	1.6%	11.8%	3.6%	1.6%	11.9%
2	8.3%	5.9%	16.3%	7.9%	4.0%	22.2%	7.5%	3.6%	20.3%
3	11.8%	7.7%	20.1%	11.8%	6.1%	29.3%	11.3%	5.9%	27.9%
4	15.7%	11.2%	26.6%	15.8%	9.2%	34.9%	14.9%	8.8%	33.5%
5	19.6%	15.7%	29.9%	19.9%	12.2%	39.8%	18.7%	11.9%	38.8%
6	23.7%	18.9%	35.2%	23.7%	15.9%	44.9%	22.4%	15.1%	43.2%
7	27.9%	21.6%	40.2%	27.7%	20.1%	49.0%	26.1%	18.5%	47.0%
8	32.5%	24.0%	42.9%	31.8%	23.6%	53.0%	29.9%	22.4%	50.6%
9	36.9%	27.2%	46.7%	35.9%	27.9%	56.3%	33.6%	26.5%	53.8%
10	40.9%	35.5%	52.1%	39.9%	32.8%	59.7%	37.3%	31.1%	57.2%

TOP	New York (4230)			Paris (4625)			London (9176)		
	RND	CNT	*ELVis*	RND	CNT	*ELVis*	RND	CNT	*ELVis*
1	3.8%	1.6%	11.6%	4.6%	1.9%	13.9%	3.4%	1.7%	11.5%
2	7.4%	3.7%	20.1%	9.3%	4.3%	22.5%	6.9%	3.5%	19.3%
3	11.2%	5.6%	26.9%	13.8%	6.9%	29.7%	10.3%	5.4%	25.5%
4	14.9%	8.0%	32.4%	18.3%	10.3%	35.8%	13.7%	7.8%	30.9%
5	18.6%	11.3%	36.8%	22.8%	13.5%	42.0%	17.1%	10.6%	35.7%
6	22.3%	14.2%	41.4%	27.3%	17.4%	47.6%	20.5%	13.9%	39.9%
7	26.0%	18.3%	45.3%	31.9%	22.6%	52.3%	24.0%	17.2%	43.8%
8	29.7%	22.3%	49.2%	36.4%	27.4%	56.5%	27.5%	20.5%	47.4%
9	33.5%	26.2%	52.4%	40.8%	33.1%	60.4%	30.9%	24.3%	50.1%
10	37.3%	30.1%	55.3%	45.2%	39.3%	64.4%	34.2%	28.2%	53.1%

Table 4.2: Percentage of test cases in top-n positions in the six cities (the larger, the better). The values in parentheses are the number of test cases considered for this experiment after filtering out those restaurants and users with less than 10 photos in the training and test sets, respectively.

To check the performance of *ELVis* we carried out evaluation experiments on test sets of the six cities. Table 4.2 shows the percentage of test examples ranked in top positions for the six cities. We filtered out restaurants with less than 10 photos in the test in order to avoid the optimistic bias (see Section 4.7) of the top-n measure, and users with less than 10 photos in the training to ensure a reasonable amount of information to learn from.

The scores are quite similar in all the cities, and there are big differences between the three methods. The baseline CNT (*centroid*) is the worst in every row of the table, which means that the vectorial representation of photos obtained by the CNN is not guided by any semantic rule related to the users' taste.

Looking at the top-10 scores, *ELVis* is around 20 percentage points better than RND (*random*) in all cities except Gijón, where we only achieved around 12% of improvement over RND. This is mainly because of the smaller size of its dataset. The best scores were obtained in the data from Paris, the second largest dataset used in our experiments.

4.8.2 Analysis regarding the amount of users' information

We also compared the performance of *ELVis* and the two baselines regarding the amount of information available for training. For this purpose, we have tested the models with 100 different test sets for each city. Each test set was built as described in Section 4.7 but using only pairs $(\vec{u}, \vec{f})$ where the user has x or more photos in the training set, with $x = \{1, 2, \ldots, 100\}$. Figures 4.5 and 4.6 present the median percentile values of the test photo for each test set, where the X axes represent the threshold used to filter out users regarding their amount of photos in the training set.

The variation in the amount of training information is barely relevant for the baselines. *Random* keeps stable around the 50%, while the *centroid* is even worse, being always above that value. *ELVis* exhibits by far the best performance. In fact, the performance of *ELVis* increases when the model is applied to users of whom we have more information, as expected. This is reflected in the graphics, where the percentile score decreases (the lower the percentile, the better performance) as the test sets are built up of users with more photos (i.e., higher values in the X axis).

However, it is important to consider the number of cases available in order to test the performance of the baselines and our model. Obviously, the more photos we require to include a user in a test set, the smaller (fewer users) it is. For this reason, the graphics in Figures 4.5 and 4.6 include the number of test cases (secondary Y axes and green lines).
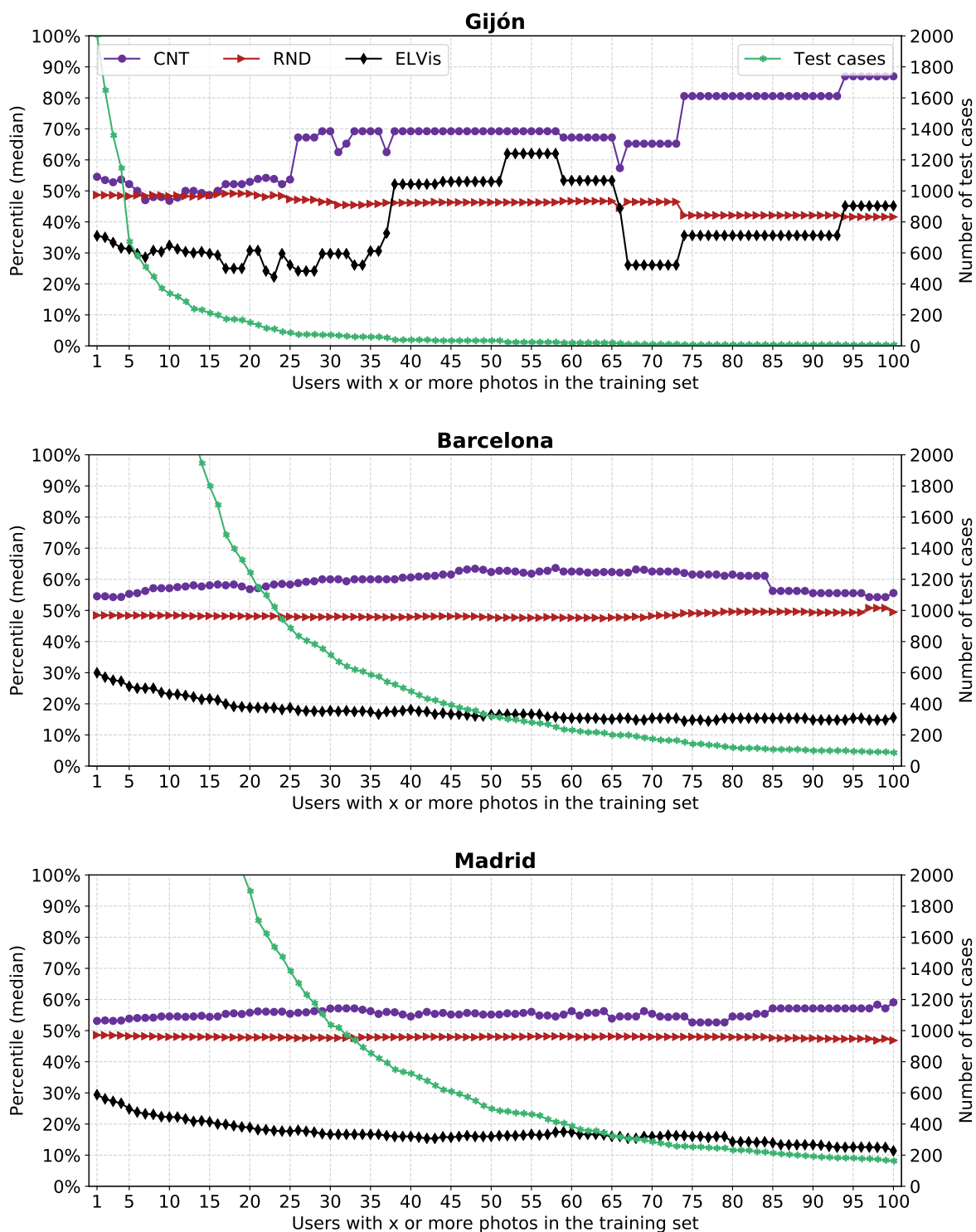
Figure 4.5: Average percentiles of the test photos, in the three Spanish cities, for users with different number of photos in training. The green line represents the number of cases available below 2,000 (see the vertical axes in the right side). We appreciate that the lack of test cases in Gijón leads to an irregular behavior of *ELVis* for users with 35 photos or more in the training set.
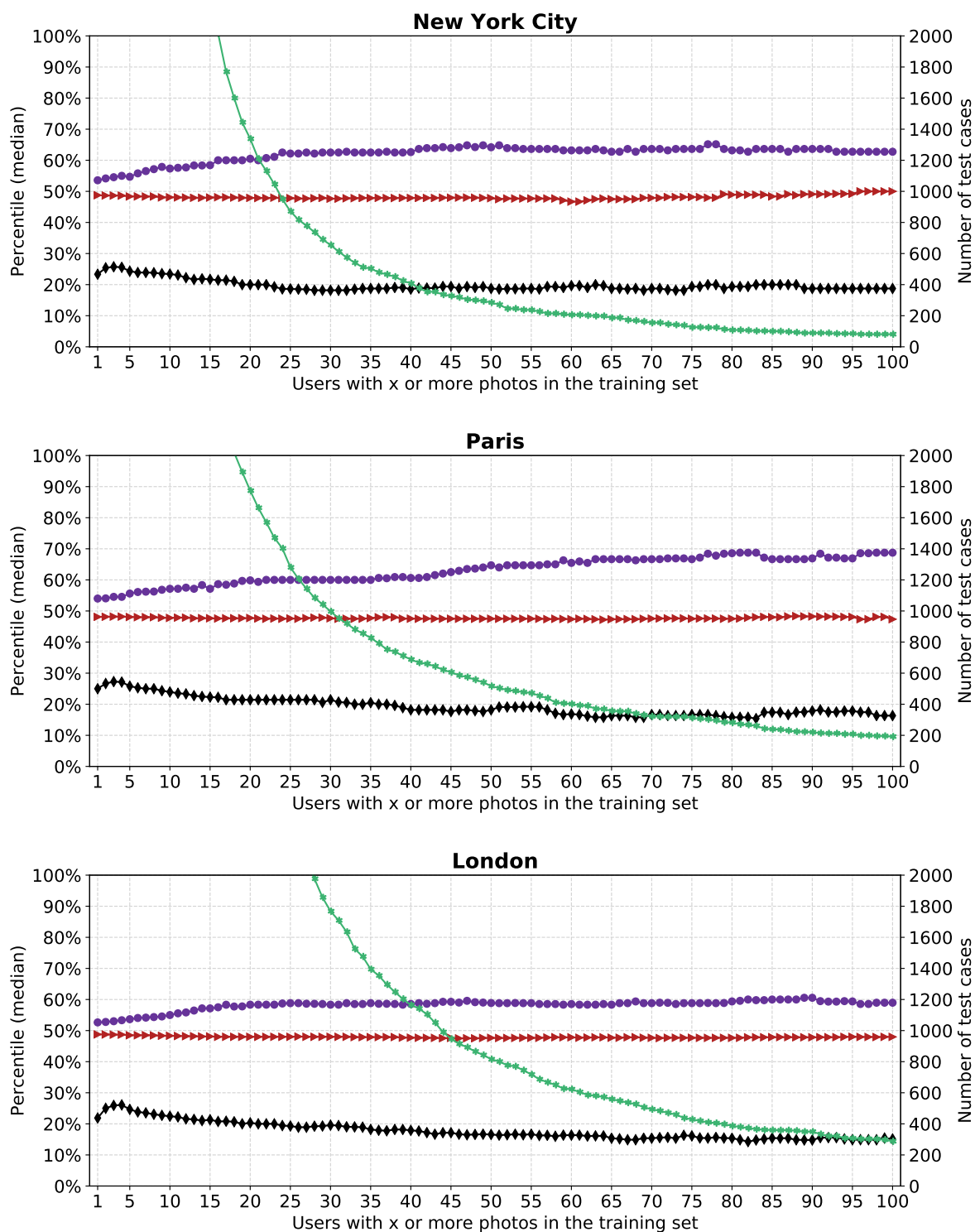
Figure 4.6: Average percentiles of the test photos, in New York, Paris and London, for users with different number of photos in training. The green line represents the number of cases available below 2,000 (see the vertical axes in the right side).

Notice that when the test sets include users with few photos (left side of the graphics) the median of the percentile increases for *ELVis*, but it is still much better than the baselines. This indicates that the performance is good enough even for users in a situation near to a *cold-start*. In the next section we propose how to use our system for users in strictly *cold-start* situations, where we have no information at all.

The small number of cases in the test sets also explains the strange behavior of the *centroid* and *ELVis* in the city of Gijón: there is only a significant amount of test cases when including users with less than 25 photos. Consequently, their behavior in the Gijón dataset is not representative from that point on.

4.8.3 The tastes of all users of a city about a restaurant

Figure 4.7 shows the 30 photos of "El Perro que Fuma" (one of the favorite restaurants of the group of authors of this work in Gijón) available at the time of downloading the data. They are ordered by equation (4.3) using all the users of the Gijón dataset

Figure 4.7: Set of photos of the restaurant *"El Perro que Fuma"* sorted using the equation 4.3 based on the preferences of all TripAdvisor users from Gijón.

If we restrict the set of users to only those who took pictures in the restaurant, the results are similar. If we look at the top-3 of the ranking, these are the variations: the first place would be for the photo that was in the ninth position before, the second would be the same, and the preferred photo for all clients (the one that occupied the first place) descends only to the third place.

Something sensibly different happens when we use the set of clients who visited the restaurant but declare that they do not like it. The favorite photo for them is the one that occupies the position 27 of 30. Curiously, this photo depicts the exterior area of the restaurant, taken from the opposite sidewalk and where you can barely see its facade, being hidden behind parked cars and a group of customers from the terrace of the restaurant.

We could use this approach to explain a recommendation for a new user with no historic information at all (*cold-start*). Provided we have a recommendation by any means to the new user, we could explain it using the top ranked photo(s) for the group of known users whose taste regarding that restaurant matches the recommendation (either positive or negative) made for the new user.

4.9 Conclusions

In this work we present a method to explain the recommendations to the users of an RS; in particular, those in which the users share not only their tastes, but also their photos of items. We built a learning system (*ELVis*) capable of predicting the photo of an item that the user would take in case of an eventual interaction with that item. In other words, the photo that reflects the most appealing aspect for the user. Users' photos serve to highlight those aspects.

The reason for pointing out users' photos as an element of explanation is that users disseminate photos to support their opinions in a way that seems unappealable. Thus, we believe that nothing can convince other users more than an image that could have been taken by themselves.

This method to convince users requires to learn a binary classification task aimed at finding out the authorship of photos. Formally, it works as an image recommender, but it has a peculiarity that marks the essence of the problem addressed in this paper. First of all we deal with photos (some of low quality) taken by users. Secondly, the photos are taken from a specific item. In other words, the photos have two dimensions that must be taken into account: their authorship and the item they portray.

To illustrate the performance of *ELVis* in a real-world scenario, we used data taken from the TripAdvisor platform of six different cities. The results, compared with a pair of baseline methods, are excellent.

On the other hand, a comment on the evaluation of this system could be made. As with all RS, it could be argued that a fair evaluation should probably be established with a field test, asking users to assess the photos proposed by the model. The point is that the photos suggested by the model can be accepted by users as representative of their preferences, even better than those photos really taken by the user. As future work, we plan to consider how to apply synonymy of images to the context of this paper.

Last but not least, there is a side effect of *ELVis* that we would like to highlight. We have seen that it allows to determine the most representative photos for a group of users within a set of photos. For example, the available photos of a restaurant can be *democratically* ordered according to the preferences of the users. This option allows, for example, the owner of a restaurant to see the aspects of her business that stand out the most, not only the customers of the premises, but also all the possible customers. We believe that this is an important tool and, thus, its development is also part of our future work.

Chapter 5

Sem: Semantics of Images

The aforementioned Collaborative Filters (Section 1.1.1) are a type of RS that try to find matches between users based on the ratings they have previously made. This approach gives good results, but it degrades when there are few interactions to learn from. The alternative would be to observe some features of the users that could be linked to their tastes. However, specific information on users or items is often not available. In this new work, we explore how to exploit the photos of items taken by users. These photos are frequent in social network interactions and have rich semantics in this context.

Our aim is to assign similar meanings to the photos of items with which the same group of users interacted. For this purpose, we define a multi-label classification task from images to sets of users. The classifier uses a general-purpose Convolutional Neural Network to extract the basic visual features, followed by additional layers necessary to accomplish the learning task.

To evaluate our proposal we compared it with CF, using the two categories of data available (restaurants and POIs). According to the experimentation carried out, the poor results achieved by CF are outperformed by our proposal, which takes into account the visual and taste semantics of the available photos.

5.1 Introduction

Typically, the datasets that constitute a learning task for a Recommender System include references to users, items, and a relationship that must be learned to extend. Usually, this relationship is an evaluation (implicit or explicit) of the users towards the items. So the idea is to suggest an item i to a user u as long as other users with similar tastes to u also like i. This is the approach of the already mentioned *Collaborative Filters*. To implement this idea, it is necessary to have a sufficient number of matches of user-item interactions. In practice, this method cannot be used with low densities of the user-item interaction matrix.

In this kind of scenarios, there is usually no other information about the users (age, interests, occupation...) or the items beyond their interaction, so CF have to learn latent characteristics of each user in order to build the RS. To achieve this, it is necessary to have a large amount of information about the behaviour of each user so that the recommendations made are not trivial. Usually, the number of interactions per user is very small, which generates the aforementioned *cold-start* problem (Section 2.2).

In this chapter we deal with a problematic situation like the one described above. The density of user-item interactions is very low (on average, each user had contact with only 1.9 items) and, additionally, we do not have specific features of either users or items. However, people use social networks to try to find out if some items may or may not be to their liking. We do it because we have available a very special relationship between users and items: the photos of some items that some users take and share.

The kernel of our proposal is a mapping function (semantics) from a set of photos to a Euclidean space. Let us recall that *semantics* studies the meaning of the expressions that we use to communicate. While these expressions are typically linguistic, the human experience also relies on signs beyond words such as visual stimuli. In this work, we will see how images can be given meaning in the context of an RS. In particular, a set of items' photos shared by users will be analyzed. Notice that photos contain very valuable information both about users and the items they interact with, as people only take photos of what is relevant to them. Our goal is to understand what these photos mean and what they can reveal about users' tastes.

This photo mapping function is what we refer to as *semantics*, but we could simply call it embedding or projection. The key point is that this semantic function must translate photos into vectors in a useful way.

Figure 5.1: Two images of pizza that elicit different reactions from users.

Images are usually encoded by the convolutional base of a general-purpose Convolutional Neural Network, typically pre-trained on ImageNet (Deng et al., 2009). In this manner, the meaning of an image is linked to the visual features that allow to detect its contents.

Unfortunately, this is not very useful for taking into account users' tastes. To illustrate this problem, let us consider the two photos of pizzas depicted in Figure 5.1. Based on their content, these photos are similar; however, they do not provoke similar reactions from users. Therefore, they should not have similar semantics (i.e., they will not be *synonymous* or near neighbors in the map). In other words, the user reactions go beyond the labels included in the photos themselves; in fact, we do not have ingredient labels of the photos because they are not necessary for the problem at hand.
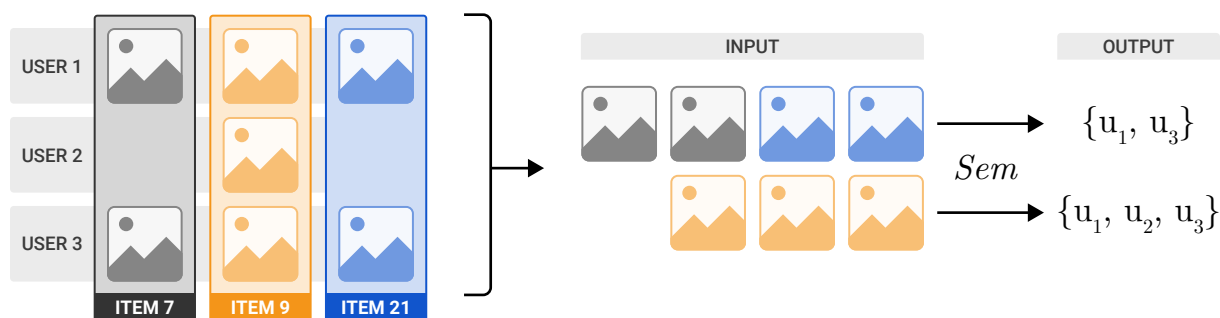
Figure 5.2: Data structure of this work. Each of the photos in an item will be tagged with all users who have interacted with the item. Note that the images of items 7 and 21 will have the same coding.

From a formal point of view, the aim of this work is to show how to learn our semantic function, *Sem* (see Figure 5.2). The goal is to grasp the following idea: the photos of items that were interacted by the same group of users should have the same semantics.

For this purpose, we try to assign a subset of users to each photo, those who have inter-

acted with it. Note that *Sem*, in order to learn these new embeddings, it starts from those provided by a CNN, which it will transform until it achieves its objective: to predict the group of users associated with each image (by means of a multi-label learning problem).

The rest of the work is organised as follows: Section 5.2 includes an overview of some relevant publications related to the topics discussed here. Section 5.3 describes in depth our interpretation of semantic mapping in this context. Then, in Section 5.4, we formally describe the approach of this work, i.e., how to learn the oursystem function, which was implemented with the network architecture described in Section 5.5. In the experimentation (Section 5.8) the systems will be evaluated using the metrics defined in Section 5.7 under the datasets detailed in Section 5.6. Section 5.9 closes the work with the main conclusions.

5.2 Related Work

When we face problems with images as input data, it is very common to use pre-trained CNNs as generic feature extractors (Nanni et al., 2017), obtaining the so-called *deep features*. Unlike traditional hand-crafted features that represent basic image properties, deep features represent semantic information for a given learning task (e.g., semantic similarity in image classification refers to the target class). In the context of RS, the concept of image semantics must be defined for the problem at hand. This is the case of the work presented by Guo et al. (2019), who define the semantics of an image through all the objects within it, against most approaches that use a single object. For this purpose, the authors proposed two Attention Networks that combine the feature embeddings of all the fine-grained image objects to provide a recommendation.

There are other interesting works in the recent literature that take advantage of visual data to provide recommendations. He and McAuley (2016) proposed a visual Bayesian personalized ranking, which uses the deep features extracted from product images by means of a pre-trained CNN. This work was improved by Kang et al. (2017), who proposed to learn the image embedding along with the RS.

More recently, Neve and McConville (2020) proposed a method for reciprocal RS, which are those based on social platforms that connect people with people. The method consists of a siamese CNN, with an ad-hoc architecture, trained to identify images that fit users' preferences.

As different convolutional backbones can be used as image feature extractors as part

of a visual-based RS, Deldjoo et al. (2021) analyzed three popular CNNs along with four visual-based RS. The experimental results proved that a deeper CNN ensures high recommendation performance.

Despite the common adoption of CNNs to represent visual data in RS tasks, Ferwerda and Tkalcic (2018) suggested to use traditional visual features, such as hue and saturation. These visual features were combined with content features in order to predict the users' personality in the social network Instagram. Kawattikul (2018) also opted to use traditional features rather than the deep features extracted by CNNs. The proposed method uses a simple weighting technique to combine both images and text descriptions. In particular, a shape-based representation is extracted from product images and a LSTM representation is obtained from product descriptions.

Focused on other characteristics of RS, Dominguez et al. (2019) performed a comparative study about the impact of several algorithms on relevant aspects such as *explainability* and users' trust. In particular, they compared black algorithms, such as the deep neural networks, with other transparent but less reliable methods. For their part, Díez et al. (2020a) proposed a framework to estimate the authorship probability of photos, which can be used to visually explain the recommendations of an RS.

In the context of the hospitality sector, we can highlight the approach available on TripAdvisor, the popular platform in which the most appealing pictures are selected to be shown when looking for restaurants and hotels (Amis, 2017). The method is based on preference learning and uses the convolutional base of a ResNet50 (He et al., 2016). Standard embeddings computed by CNNs are also used in (Chu and Tsai, 2017) to provide restaurant recommendations based on visual features. Yang et al. (2015) designed their own CNN to learn the similarity distance metric between food images. Their food preference learning approach can be used as part of a restaurant RS.

Smart tourism destination is the topic considered by Figueredo et al. (2018), who proposed a solution capable of detecting tourist preferences using images from social media networks along with CNNs and fuzzy logic. Sertkan et al. (2020) presented an approach based on fine-tuned CNNs to represent travel behavioral patterns. As a result, they are able to determine a tourist profile from a collection of user's pictures. More approaches related to travel RS can be found in the review provided by Chaudhari and Thakkar (2020).

To delve further into the field, we refer the interested reader to a recent survey focused on recommender systems that take advantage of multimedia data in different domains, such as tourism or food (Deldjoo et al., 2020).

As can be seen, in previous works the usage of semantics in RS is characterized by the use or learning of traditional semantic representations, only based on content or visual features, which is not a bad strategy since the content of an image partly represents the user's tastes or preferences. Another point in common between those works is the creation of user profiles that summarize the system's knowledge of them. The quality of these profiles increases as the number of user interactions increases; however, in *cold-start* situations these systems may not have enough information to have reliable profiles.

The semantics that we present in this article goes beyond what has been seen in the works we have just discussed, since our semantics will take into account both the visual content of the images and the interactions that users have had with the items, which is a novelty. In addition, the semantics defined will allow us to work in *cold-start* scenarios where no previous user information is available because it can work with just a photo.

5.3   Semantics

A common strategy for building an RS is to project both users and items into a common Euclidean space $\mathbb{R}^k$. These projections, embeddings or mappings, which must be learned as functions, require an initial vector representation for both items and users. When no representation is available, artificial encodings are used.

In particular, one-hot codification is one of the most prevalent artificial encodings due to its simplicity. Let us recall that one-hot codification for an object of index i (in the set of users or items) is represented by a binary vector where all the components are zero except the one with index i, which is one.

The situation described above is found in matrix factorization methods, which are a class of collaborative filters. In this case, the RS is constructed by estimating the affinity between users and items as the inner product between the projections of their one-hot representation in a vector space $\mathbb{R}^k$. The main disadvantage of this type of method is that it is not possible to make recommendations for users or on items that have not been included in the learning process: they do not have an initial one-hot representation and, therefore, they do not have a projection in $\mathbb{R}^k$.

The alternative to this approach is to have an *external* vector representation for both users and items. This is the case of *content-based* RS, which commonly use not only the features of the items, but also a representative profile of the users that can be built with enough user data.

In contrast to these classical approaches, we propose a method that: (1) does not depend on any initial artificial encoding, being able to work in *cold-start* situations; and (2) does not require item features or user profiles. More specifically, we assume that, for each user, we have a small set of photos that capture their tastes. In this scenario, we can calculate the semantic mapping function of their photos. As a result, whenever we intend to summarize the user tastes, we could use the centroid in $\mathbb{R}^k$ of their photos codified by this mapping function.

On the other hand, for each item, we have a set of points in $\mathbb{R}^k$ formed by the mapping of the photos that users took of it. Correspondingly, we intend for the Euclidean representations of these photos to be close. Therefore, the core point of this research is the definition of a semantic mapping function that makes everything work correctly.
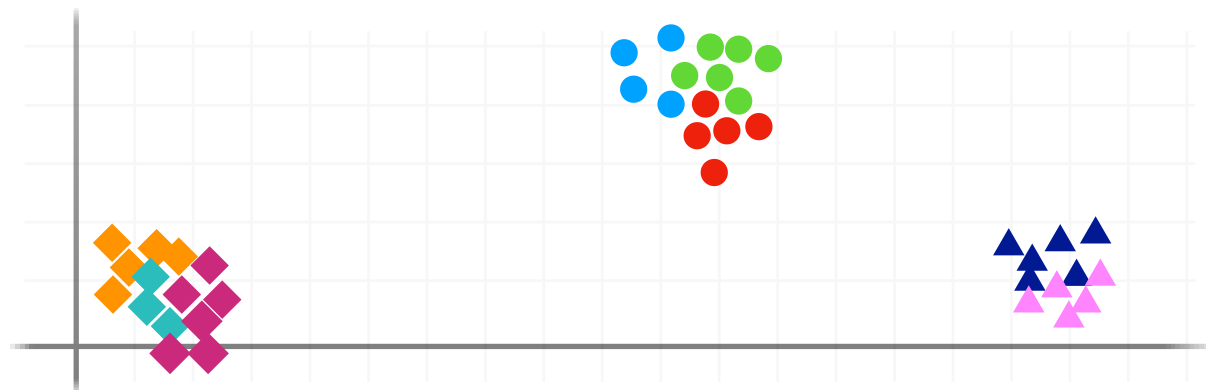
Figure 5.3: Example of an ideal semantics of photographs expressed as two-dimensional vectors. Notice that the points of the same color and shape correspond to the semantics of photos of the same item.

As stated in Section 5.1, the fundamental idea of this article is that the photos of items with which the same group of users interacted should have the same semantics. This principle leads us to the following:

- The photos of the same item should have the same or very similar semantics, regardless of their content.

- The photos of two items, A and B, should have similar semantics when almost all users who interacted with A also interacted with B.

In practical terms, we aim at having a semantics of photos like those shown in Figure 5.3. As can be observed, there is a group of photos of three items, represented by circles (bottom left), with very similar semantics (i.e., the same group of users interacted with these three items). Another group of users seems to have interacted with other three items, represented by squares (top right). Finally, there is another group of users who interacted with two items, represented by triangles (bottom right). Summarizing, we assume that similar photos are the ones that are close using the Euclidean distance. However, we will see that this definition of similarity is not necessarily the best.

The following section describes the method we propose to learn how to generate these projections or semantic mappings as well as the similarity measure we will use.

5.4   Formal framework

Let us consider a set of users $\mathcal{U}$, a set of *Items*, and a set of images , photos that users took from items. We register their relationships in a matrix $\vec{M}$ with binary components with one row per user and one column per image (see Figure 5.4). We assume that $\vec{M}(u, i) = 1$ whenever user u had an interaction with the item photographed (by u or by another user) in image i. In all other cases, $\vec{M}(u, i) = 0$.

Notice that the columns $\vec{M}(\cdot, i)$ are equal for all the images of the same item. Therefore, we can encode items by groups of equal columns of $\vec{M}$. On the other hand, these columns can also be understood as a set of users.

From the point of view of an RS, two items A and B with equal (or almost equal) column representations, are items with the same (or almost the same) interactions with users. Thus, A and B would play the same role in a recommendation environment.

Having all this into account, we define

		item 1			item 2			item n
		$\mathbf{img}_{1,1}$	$\mathbf{img}_{1,2}$	$\mathbf{img}_{1,3}$	$\mathbf{img}_{2,1}$	$\mathbf{img}_{2,2}$	. . .	$\mathbf{img}_{n,1}$
$M =$	**user** 1	1	1	1	0	0	. . .	1
	user 2	0	0	0	1	1	. . .	1
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	user m	1	1	1	0	0	. . .	0

Figure 5.4: Matrix $\vec{M}$ with the relationships between users, items, and images, provided $m = |\mathcal{U}|$ and $n = |Items|$.

$$h :\longrightarrow \{0,1\}^{\mathcal{U}}, \ i \longmapsto h(i) = \vec{M}(\cdot, i). \tag{5.1}$$

That is, for each image i, $h(i)$ is an encoding of the item photographed in i. We would like to emphasize that

$$\langle h(i), h(j) \rangle = \langle \vec{M}(\cdot, i), \vec{M}(\cdot, j) \rangle = |\vec{M}(\cdot, i) \cap \vec{M}(\cdot, j)|. \tag{5.2}$$

In other words, the inner product of the projections of the images i and j is the number of common users who interacted with the items of these two images.

This is exactly the idea that we want to formalize when defining a semantic (mapping) function. To make it operative, we factorize (using a deep network presented in Section 5.5) the function h into a function from images to a Euclidean space (the continuous part) followed by a linear multi-label classifier; that is, a binary classifier for each component indexed by a user. In this way, we arrive to the key definition of this work.

The *Semantics of Images* (*Sem*) is given by a Euclidean mapping from which h can be obtained using a multi-label classifier (*ml*). In symbols,

$$h :\xrightarrow{Sem} \mathbb{R}^k \xrightarrow{ml} \{0,1\}^{\mathcal{U}}. \tag{5.3}$$

The *semantics* of an image i is defined by $Sem(i)$ and, according to equation (5.2), the *similarity* is defined by the inner product.

It should be noted that, as the number of users available in some recommendation scenarios can be large, in order to simplify the multi-label classifier complexity, we are going to selected a percentage of the *top active* users (those with more reviews available in the training set) to act as target labels in the learning task.

5.5 Network architecture

This section presents the network employed to solve the multi-label learning task previously described, see Figure 5.5.

Sem receives as input an image i, which is first encoded by means of a densely connected CNN (DenseNet) (Huang et al., 2017).
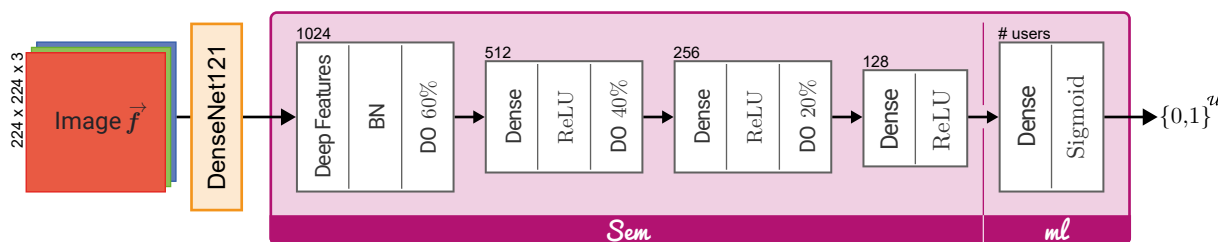
Figure 5.5: *Sem* architecture.

In particular, we used the convolutional base of a DenseNet-121 pre-trained on ImageNet[1], thus obtaining a general-purpose 1024-dimensional embedding. Once the input image is encoded, we pass it through a batch normalization layer (BN) (Ioffe and Szegedy, 2015) in order to improve the speed and stability of the learning process.

Then, the output of the BN is passed through three processing blocks of sizes 512, 256, and 128. Each block is composed of a hidden fully connected layer (FC) and a rectified linear unit (ReLU) (Nair and Hinton, 2010) as activation function to achieve the non-linearity. Notice that the output of the BN layer and the first two processing blocks are followed by a dropout layer (DO) (Srivastava et al., 2014) in order to reduce the amount of overfitting. Finally, to solve the multi-label problem that we are facing, the output block of our network is an FC layer, with size $|\mathcal{U}|$, followed by a sigmoid (σ) activation function to achieve a probability value for each user $u \in \mathcal{U}$.

The model was trained by minimizing the binary cross-entropy loss, using the Adam optimizer (Kingma and Ba, 2014). With the aim of increasing the performance, we added weights to the loss function making five times more relevant to fail a 1 than a 0. This makes the model more accurate in predicting the correct users in each image and it will also have more leeway to predict other users that it also considers relevant.

By removing the last two layers of the trained model, we can take the 128-feature vector ($k = 128$ in equation (5.3)) obtained in the third block to represent the semantics captured from input images. These are the so-called *Sem* features.

The entire process described in this section is reflected in Figure 5.6, in which all the images of the same item will have the same output (users who interacted with it) in the multi-label classification problem.

---

[1] https://keras.io/api/applications/densenet/#densenet121-function
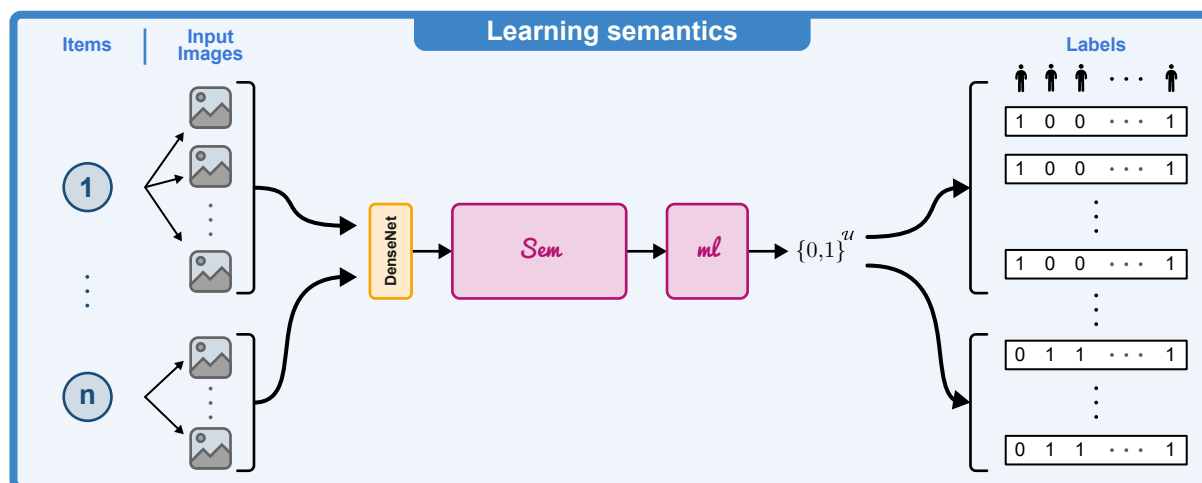
Figure 5.6: Illustration of the formal framework to learn the semantics of the input images, which in this example are the photos of a set of items. Each input image is processed to obtain a binary vector of dimension the number of users, with a one in the positions corresponding to the users who interacted with the item and a zero otherwise.

5.6   Dataset

The adequacy of the approach presented in this work was assessed using the datasets from the two available categories or domains (Part I), i.e. restaurants and POIs.

In each domain, two kinds of experiments were performed in order to, first compare our system with a traditional collaborative filter, and second to analyze the performance of our system recommending items to previously unseen users. The latter was tackled to check the answer to the *cold-start* problem, where the systems should try to recommend items to users without any previous information.

With the aim of performing these two experiments, we need to split each dataset into training/dev/test partitions keeping in mind that, in the first experiment (collaborative filter) all users and items must be in the training set.

However, the second one (*cold-start*) requires all restaurants in training, but not the users, who must be distributed among the partitions keeping a group of unseen ones for the test set. The reason behind these restrictions is explained in Sections 5.8.1 and 5.8.2.

5.6.1 Restaurant reviews

The first domain focuses on restaurants (items), customers (users) and photographs of dishes taken by them.

Each customer on the set uploaded one or more *reviews*, each in a different restaurant, including some photographs of their experience (notice that we only consider the reviews with pictures). Since our objective is to acquire the semantics of images in terms of users' tastes, we filtered out all the images whose main content is not food.

In this case, we interpret that users had a positive interaction with the restaurants they visited. We could have used another criterion but, to defend this point of view, we can see recommendations as suggestions of restaurants that should be visited.

5.6.2 Points of interest

The second domain focuses on points of interest (POIs) in cities. In this domain, visitors (users) go to one or more POIs (items) and take pictures of them. As in the restaurants case, we consider that users had a positive interaction with the items they visited.

Although we might think that all the photos of the same POI are the same or very similar, not all visitors take the same photographs because their interests are not always the same. For example, when someone visits a cathedral they may be more interested in the exterior architecture of the building, the interior stained glass windows, the holy images, or the frescoes painted on it. Therefore, the perception of the same POI by several visitors may be different.

It is also worth noting that all the photographs taken in a POI, in most cases, can be unambiguously related with only one item, whereas in the restaurant domain, very similar food pictures can be taken in more than one item. That means that you will not find a picture of the Eiffel Tower on another POI, but you will find almost the same picture of pizza in multiple restaurants.

5.7 Evaluation

Once the semantic (mapping) function is defined (Section 5.4), we must devise a method to evaluate it. For this purpose, let us assume that a test user shows a small set of photos capturing their preferences.
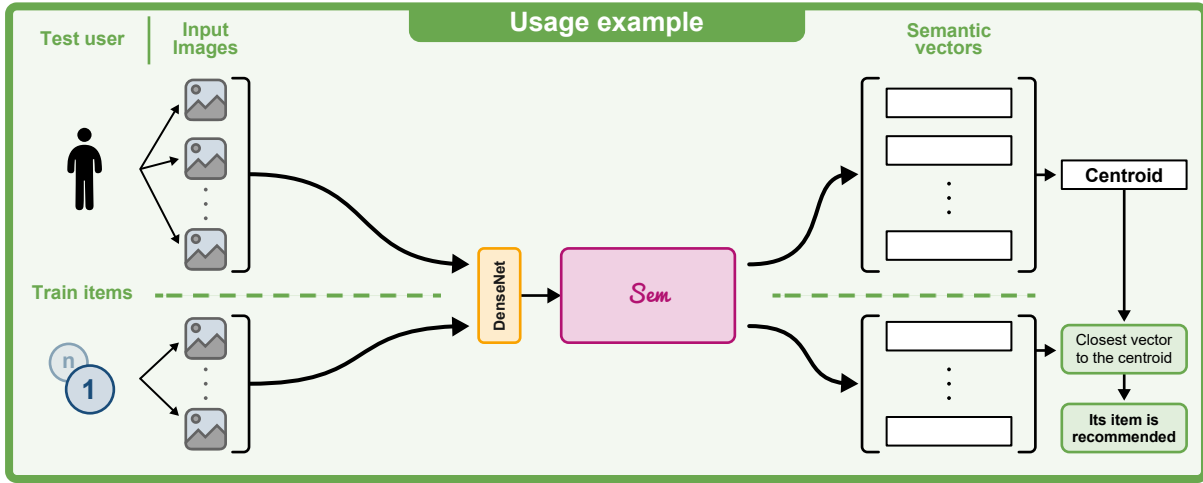
Figure 5.7: Illustration of a usage example of the validation procedure, in which the semantic vectors of the images of a new user are computed with the network trained in Figure 5.6. The centroid of these semantic vectors is calculated and compared to the semantic vectors of the photos considered during the training phase. The closest photo to the user's centroid corresponds to the item that will be recommended.

First we calculate the semantics of each photo, which presumably should be similar. Next, we compute the centroid in $\mathbb{R}^k$ with the intention of summarizing the preferences of the user. If the semantics is correctly estimated, in the restaurant domain, for example, the most *similar* photo to the centroid should be one taken in a restaurant that the user would like to visit. Notice that, in this semantic context, the most similar will be the one with the greatest dot product with the centroid, see equation (5.2).

This procedure will yield a list of items ordered from highest to lowest affinity. Then, if the first item of the list is one the user has actually interacted with, we count a hit (Figure 5.7); doing this for all the test users will return us the Top 1 accuracy. Following the same procedure, but with the first five items of the list, we will obtain the Top 5 accuracy and so on, applying the formula

$$\text{Top-N accuracy}(Y, \hat{Y}) = \frac{1}{m} \sum_{i}^{m} 1_{y_i \cap \hat{y}_i^N \neq \emptyset}, \tag{5.4}$$

where Y and $\hat{Y}$ are the sets of visited and recommended items, respectively, for all the m users of the dataset, y_i is the set of visited items for user i, $\hat{y}_i^N$ refers to the Top-N recommended items for user i, and, finally, 1_p will be 1 when the predicate p is true and 0 otherwise.

We will also calculate the Precision and Recall measures in order to further analyze the quality of the recommendation of these rankings using the following formulas:

$$\text{Prec@N}(Y, \hat{Y}) = \frac{1}{m} \sum_{i}^{m} \frac{|y_i \cap \hat{y}_i^N|}{|\hat{y}_i^N|} = \frac{1}{m} \sum_{i}^{m} \frac{|y_i \cap \hat{y}_i^N|}{|N|} \tag{5.5}$$

$$\text{Rec@N}(Y, \hat{Y}) = \frac{1}{m} \sum_{i}^{m} \frac{|y_i \cap \hat{y}_i^N|}{|y_i|} \tag{5.6}$$

Table 5.1 shows, for a user i, multiple simulated examples of visited restaurants (y_i), Top-3 of restaurants that the system could predict as most related to this user ($\hat{y}_i$) and the behavior of Top-3 accuracy, Prec@3 and Rec@3 metrics in those situations.

y_i	$\hat{y}_i^3$	$|y_i \cap \hat{y}_i^3|$	$|y_i|$	Top-3 accuracy	Prec@3	Rec@3
r_{71}, r_{43}	r_{34}, r_{54}, r_{129}	0	2	0	0	0
r_{311}, r_3, r_{198}	r_{92}, r_{198}, r_{311}	2	3	1	0.67	0.67
r_{203}	r_{48}, r_{203}, r_7	1	1	1	0,33	1
r_{57}, r_{32}	r_{286}, r_{30}, r_{37}	0	2	0	0	0
r_{47}, r_8, r_{93}	r_{143}, r_1, r_{93}	1	3	1	0.33	0.33
r_{81}, r_{32}	r_{81}, r_{327}, r_{32}	2	2	1	0.67	1
$r_{27}, r_9, r_{111}, r_{41}, r_8$	r_{41}, r_9, r_{111}	3	5	1	1	0.6

Table 5.1: Multiple simulated examples, for a user i, of visited restaurants (y_i), Top-3 of restaurants that a system could predict as most related to this user ($\hat{y}_i$) and the behavior of Top-3 accuracy, Prec@3 and Rec@3 metrics in those situations.

- Prec@N shows the proportion of recommended restaurants which are relevant for the user, that is, if 3 restaurants are recommended to a user, the only values that can be obtained are 0, 0.33, 0.66, or 1, depending, respectively, on whether the user actually visited 0, 1, 2, or 3 of the recommended restaurants. However, the vast majority of users have written reviews in only 1 or 2 restaurants. Therefore, even if the system outputs the best recommendation possible, in some cases would be impossible to obtain a perfect result for this metric, as it happens, for example, in the third and sixth case of Table 5.1.

- Rec@N computes the proportion of relevant restaurants which have been predicted. If 3 restaurants are recommended for a user who actually likes 5, the values that can be obtained are 0, 0.2, 0.4, or 0.6, depending on whether the user actually visited 0, 1, 2, or 3 of them.

In other words, even if the best possible recommendation is given, it is not possible to reach the maximum in Prec@3 as can be seen in the last example of Table 5.1.

- Top-N accuracy is a metric independent of the number of restaurant predictions and the number of visited restaurants by the users, so for each case a 1 will be obtained if any of the restaurants visited for the user is among the recommended ones and a 0 in the opposite case.

Thus, each of these measures will have a more appropriate scope of application: i) Prec@N and Rec@N are more suitable for comparing the performance of different recommender systems with each other, since the particular characteristics of each dataset affect all the systems equally, and ii) Top-N accuracy is more suitable for assessing the quality of recommendations in a given dataset.

Let us recall that the semantics learned does not have to simply focus on the content of the photo; Convolutional Neural Networks do that very well, as explained in Section 5.1. In our case, the semantics learned goes further, since it not only takes into account the content of the photographs, but also the set of restaurants or points of interest that a user visited.

For example, in the points of interest domain, there are many tourists visiting Notre Dame Cathedral and the Eiffel Tower when they go to Paris. If we have a system that obtains semantics based only on the content of the photographs (such as a CNN), it will not be able to recommend the Eiffel Tower to a user who has visited the Notre Dame Cathedral, since their photographs are not similar. However, our system will be able to do so because the semantics it learns also takes into account the places visited by the users.

5.8 Experimentation and results

Once the case study and the evaluation have been defined, we can start with the experimentation. As said at the beginning of the section, we want to perform two experiments: i) comparing our system with a traditional collaborative filter and ii) analyzing the performance in a *cold-start* scenario. In both experiments we are going to test, not only the embedding obtained by our system, but also that produced by the pre-trained convolutional base of a DenseNet-121 (used as input in our model). This will allow us to test whether the semantics we are trying to learn actually performs better than traditional content-based encoding.

To do so, we only need to remove our system (*Sem*) from the evaluation procedure showed in Figure 5.7, using directly the encoding of the DenseNet as semantic vectors. To find the closest vector to the centroid, in this case, we need to change the dot product for the Euclidean distance due to the nature of the ImageNet problem that this type of pre-trained CNNs usually solve.

The first experiment also requires the creation of a collaborative filtering system to compare with. For this purpose, we have created a model based on the projection of users and items in a new latent space in which the probability of interaction between them is computed. This model will receive users and items encoded as one-hot vectors and, by calculating embeddings, it projects them in a common 64-dimensional space.

It then concatenates both vectors, which are fully connected to another 64-dimensional layer (using a ReLU activation function). This layer is then connected to a single cell output-layer that predicts the probability of interaction between users and items by applying a sigmoid function. This particular method of solving collaborative filtering tasks is the so called *Neural network-based Collaborative Filtering* (He et al., 2017).

We cannot use the same evaluation procedure explained above since we do not have images so, in order to obtain the affinity sorted list of items, we will use the CF model to predict, for each test user, the probability of interaction for all the items. The rest of the evaluation procedure stays intact.

It is worth noting that we use the same experimental procedure for all the systems we are going to compare (*Sem*, DenseNet, and a collaborative filter). First, the dataset (restaurants or POIs) is divided into training, dev, and test partitions as appropriate (following the constraints of each of the two experiments). Then, using the training and dev sets of Barcelona (as it is a medium-sized city), we performed a grid-search of hyperparameters and architectures for each system and in both experiments, in order to find the optimal combination for the problem to be solved.

Finally, with the best combination for each system, we trained all the systems joining the training and dev subsets. The test subset was not seen in the training procedure and will only be used to perform the final evaluation described in the preceding section.

5.8.1 Experiment 1: Comparison against a collaborative filter

In this first experiment we want to evaluate the performance of the semantics learned by our *Sem*, following equation (5.3), against a traditional recommender system such as a collaborative filter. We also incorporate to the comparison the embedding generated by a Convolutional Neural Network in order to show that taking into account only the content of the photographs yields worse results. Recall that the semantics learned by our system considers not only the content of the photographs, but also the places visited by the users. In this comparison, the embeddings generated by the DenseNet network will be used to generate recommendations in the same way as those generated by *Sem* (see Figure 5.7).

| | Dataset | Training + Dev | | Test |
		Reviews	Density	Reviews
Restaurants	Gijón	6341	0.26	530
	Barcelona	47403	0.03	5486
	Madrid	62353	0.03	7208
	NYC	72725	0.02	7463
	Paris	87014	0.02	9486
	London	160288	0.01	15254
POIs	Barcelona	38009	0.24	4321
	NYC	65345	0.22	9126
	London	70425	0.10	7820

Table 5.2: Dataset division in training/dev/test for the first experiment. Density represents the percentage of 1s in the user-item interaction matrix.

To carry out this first experiment, the datasets were separated into training and test sets, ensuring that all users and items were present in the training set. This requisite is essential for the operation of collaborative filters, although it is not necessary for our system. To perform this division, the reviews of users who have interacted with just one or two items are forced to belong to the training set (this is necessary since the training set will be further subdivided to obtain a validation set); for the remaining users, those who have reviewed three or more items, their interactions are divided between the training and test sets. The characteristics of the resulting subsets obtained after this split are shown in Table 5.2. As can be observed, the number of reviews used as test is approximately 10% of the total number of reviews.

If we focus on the densities of the user-item matrix, in the restaurant domain the densities are extremely low, between 0.01% and 0.03% (with the exception of Gijón, a city that

is also very different in size with respect to the rest of the cities), while in the points of interest domain the densities vary between 0.10% and 0.24%.

Top N accuracy results obtained for this experiment can be seen in Table 5.3. If we focus on the accuracy when recommending a single item (Top 1) we see that *Sem*, the system presented in this work, obtains the best performance on all datasets in both domains. In contrast, the collaborative filter obtains the worst results, which is not a surprise since the user-item matrix hardly presents any interactions. The performance of the collaborative filter in the POI domain for Barcelona may be striking, since it obtains a 62.49% when in the rest of the cases it is barely close to 10%. Analyzing this result, we have observed that this high percentage is due to the fact that, unlike the rest of the cities, there is a POI (the Sagrada Familia) that has a much higher number of reviews than the rest of the POIs of that city, which facilitates the success in the recommendation.

	Dataset	Top 1			Top 5			Top 10		
		CF	DNet	*Sem*	CF	DNet	*Sem*	CF	DNet	*Sem*
Restaurants	Gijón	7.79	12.31	**24.12**	12.56	26.13	**33.42**	15.58	37.94	**41.46**
	Barcelona	1.93	7.80	**14.43**	4.90	15.99	**24.49**	6.70	20.44	**30.73**
	Madrid	2.41	10.62	**12.38**	5.00	20.21	**21.59**	6.71	26.31	**27.10**
	New York	4.57	15.76	**16.56**	7.92	**28.00**	28.00	10.03	**34.14**	33.11
	Paris	0.21	7.46	**9.98**	2.55	15.28	**20.40**	4.46	19.68	**26.25**
	London	0.69	7.55	**10.77**	3.64	16.84	**23.90**	6.29	21.99	**29.44**
POIs	Barcelona	62.49	80.89	**81.83**	74.00	**94.06**	91.52	86.55	**96.66**	93.95
	New York	7.28	69.61	**88.28**	76.41	91.80	**93.30**	86.85	**95.59**	95.01
	London	14.99	59.56	**70.32**	46.82	**81.96**	81.79	57.72	**88.04**	84.72

Table 5.3: Top N accuracy results (in percentage) between the Collaborative Filter, and the embeddings learned by DenseNet and *Sem*. Best results are in bold.

If we ask our system to give us a list with five recommendations (Top 5), it achieves the best result in all the cities in the restaurant domain. Regarding the POI domain, there is a remarkable improvement when using the DenseNet embeddings in the city of Barcelona, while *Sem* is the best in New York and both systems obtain similar results in London. There is also a considerable improvement in the performance of CF when using POIs, although with considerably worse results than the other two systems (DenseNet and *Sem*).

We believe that this improvement is due to the fact that the POI problem is a bit simpler for several reasons: 1) the number of POIs is smaller than the number of restaurants in the same city (in Table 2.2, it can be seen that cities have approximately 10 times

| Dataset | | Prec@1 | | | Prec@5 | | | Prec@10 | | |
		CF	DNet	*Sem*	CF	DNet	*Sem*	CF	DNet	*Sem*
Restaurants	Gijón	7.79	12.31	**24.12**	2.56	5.23	**6.73**	1.66	3.84	**4.27**
	Barcelona	1.93	7.80	**14.43**	0.98	3.20	**4.94**	0.67	2.05	**3.12**
	Madrid	2.41	10.62	**12.38**	1.00	4.04	**4.33**	0.68	2.63	**2.73**
	New York	4.57	15.76	**16.56**	1.59	5.60	**5.61**	1.01	**3.42**	3.32
	Paris	0.21	7.46	**9.98**	0.51	3.06	**4.10**	0.45	1.97	**2.65**
	London	0.69	7.55	**10.77**	0.73	3.37	**4.79**	0.63	2.20	**2.96**
POIs	Barcelona	62.49	80.89	**81.83**	14.83	**19.19**	18.55	8.76	**10.07**	9.62
	New York	7.28	69.61	**88.28**	15.33	18.97	**19.22**	8.87	**10.15**	10.01
	London	14.99	59.56	**70.32**	9.40	16.67	**16.75**	5.83	**9.10**	8.78

Table 5.4: Precision results (in percentage) between the Collaborative Filter, and the embeddings learned by DenseNet and *Sem*. Best results are in bold.

more restaurants than POIs); 2) the density of the user-item matrix in the POI domain is higher than in the restaurant domain, so there are more data available during the training stage; and 3) when a tourist visits a new city, there are certain points of interest that are almost always visited.

If we focus on the performance when 10 items are recommended (Top 10): our system obtains the best results in all the datasets of the restaurant domain except one (New York); and in the POIs domain, DenseNet obtains the best result in Barcelona and London, while both approaches obtain a very similar result in New York.

If we want to see which representation is more effective when trying to answer the question *how many of the recommended items are relevant?*, we have to look at Table 5.4 showing the results in Precision. We can see in the table that the semantics learned by *Sem* is the best in the vast majority of the tests performed, so we can conclude that our semantics is more accurate in the recommendations than the other methods. The Pecision values are quite high in the POI domain when only one place is recommended; this reinforces the previous statement that this domain is simpler than the restaurant domain. It is also observed that the Precision values in both domains decrease for all systems as the number of recommended items increases. This is because the average number of interactions per user is very low (around 1.9) and as the list of recommended items increases, it is inevitable that items that are not relevant are introduced in the recommendation.

If we want to answer the question: *How many of the relevant items have been recommended?*, then we have to look at Table 5.5 where we show the results on the Recall

measure.

In this case we see that when only one item can be recommended, *Sem* is the one that obtains the best results. As the number of recommended items increases, the embedding obtained by DenseNet is equal in performance to that obtained by *Sem*. This happens mainly in the POI domain and we believe, as we mentioned before, that this may be due to the fact that it is a simpler domain.

Dataset		Rec@1			Rec@5			Rec@10		
		CF	DNet	*Sem*	CF	DNet	*Sem*	CF	DNet	*Sem*
Restaurants	Gijón	7.47	11.89	**22.82**	12.02	24.21	**30.84**	14.57	34.97	**38.22**
	Barcelona	1.63	7.59	**12.76**	4.20	15.38	**21.69**	5.64	19.49	**27.20**
	Madrid	2.19	10.32	**11.43**	4.48	19.38	**19.52**	5.86	**25.02**	24.19
	New York	4.21	15.49	**15.79**	7.32	**27.31**	26.41	9.32	**33.09**	31.12
	Paris	0.15	7.29	**8.99**	2.13	14.70	**17.96**	3.77	18.77	**23.02**
	London	0.61	7.40	**9.90**	3.22	16.35	**21.84**	5.51	21.17	**26.74**
POIs	Barcelona	59.28	76.97	**77.30**	69.64	**88.47**	85.64	81.59	**91.02**	87.92
	New York	6.40	64.97	**80.79**	70.57	85.16	**85.56**	80.11	**89.06**	87.70
	London	14.32	56.61	**64.76**	43.27	**76.57**	75.42	53.12	**81.96**	78.21

Table 5.5: Recall results (in percentage) between the Collaborative Filter, and the embeddings learned by DenseNet and *Sem*. Best results are in bold.

In summary, *Sem* presents in general the best performance, although when a large list of recommendations is made in the simplest domain, the embeddings provided by a DenseNet are comparable and even better in some cases. What is clear after seeing these results is that the performance of the collaborative filter is quite far from the performance of our system, being significantly worse (using 0.05 as p-value) in all cases applying the Bonferroni-Dunn test (Dunn, 1961).

5.8.2   Experiment 2: Performance in *cold-start* situations

In this second experiment we intend to analyze the performance of our system when asked to make recommendations for users who were not present in the training set. Collaborative filters are not able to make recommendations to such users and, for this reason, this system is not used in this experiment.

To carry out an experimental evaluation, we split the datasets into training, dev and test sets. The division was made in two steps.

Dataset		Training + Dev			Test			
		Users	Avg it/usr	Popularity	=1	=2	=3	≥ 4
Restaurants	Gijón	3330	1.61	8.65	833	136	37	46
	Barcelona	20527	2.04	1.52	4683	873	336	506
	Madrid	26670	2.05	2.54	5990	1196	442	700
	NYC	34746	1.81	3.39	8015	1583	619	699
	Paris	38511	1.99	0.57	8876	1528	558	872
	London	78685	1.73	0.83	19557	3078	1035	1480
POIs	Barcelona	17411	1.83	41.96	3906	955	398	522
	NYC	27558	2.04	21.51	5802	1560	691	1082
	London	32022	1.86	14.69	7450	1623	616	876

Table 5.6: Dataset division in training/dev/test for the second experiment. The test set is showed in terms of the number of users with 1, 2, 3, o ≥ 4 items reviewed. Notice the popularity relevance in the POIs dataset.

First, we randomly selected half of the users for the training set and the other half for the dev and test sets; and then, we moved to the training set those users in the other two sets with reviews on items that only appear in them.

Thus, we guarantee that there are no references to unknown items in the dev and test sets. In Table 5.6 we show a description of the sets after performing the split. Notice that we show the users of the test set broken down by the number of reviews they have since we will focus the analysis of the results on seeing the performance of the different algorithms based on the number of registered user interactions.

We also show the percentage of users who visited the most popular item in the training set. For example, in the set of POIs of Barcelona, 41.96% of users visited the Sagrada Familia (something that, as mentioned in the previous section, affects the results obtained to a certain extent). It can be seen that the most popular items in the restaurant domain have a much lower percentage than the most popular items in the POI domain.

Figure 5.8 depicts the Top N accuracy performance of the embeddings (*Sem* and DenseNet), broken down by the number of reviews of the tests users. We also include under the name *Popularity* the results obtained by recommending on the basis of the number of reviews, i.e., ordering items (restaurants or POIs) from the most to the least reviewed (popular). Provided we have already shown in the previous experiment that the three methods compared present a similar behavior in Precision and in Recall to that of Top N accuracy, in this experiment we will only show the graphs obtained in the latter measure, in order to ease the reading of the document.
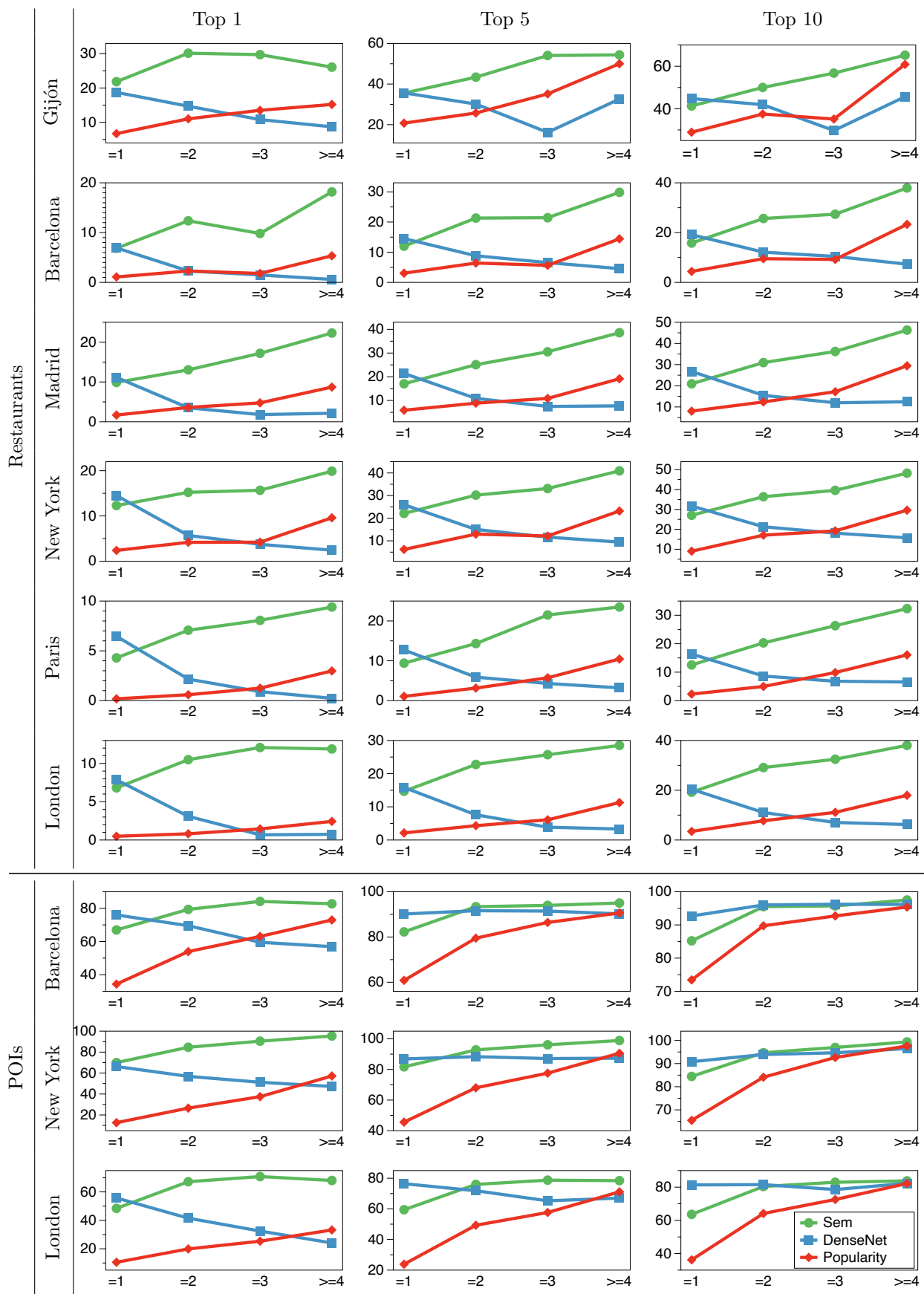
Figure 5.8: Top N accuracy of the three methods analyzed in the second experiment. The horizontal axis represents the number of reviews of each user.

Moreover, both Precision and Recall scores are not bounded as usual in these problems (e.g., it is impossible for a user interacting with less than N items to achieve a maximum Prec@N, as well as for a user interacting with more than N to achieve a maximum Rec@N), but the Top N accuracy does not present this drawback, which makes it easier to compare the performance obtained between the different datasets.

The most relevant aspect that can be appreciated in these graphs is that, for the representation obtained by *Sem*, the greater the number of user interactions (or number of reviews), the better the performance, outperforming the other two systems. With only one interaction, it looks like the DenseNet encoding is enough to obtain a slightly better result than *Sem*, but not significantly.

If we increase the number of interactions, the DenseNet encoding performance becomes worse and worse in most cases. This is because when you have a single image, for example of a pizza, if you look only at the content (as DenseNet does), it is normal to recommend pizza restaurants. The problem starts when the user has taken photos in more than one restaurant (pizza an sushi for example).

In this case, when the centroid is computed using both image embeddings, the obtained vector represents an average of two contents, resulting in a bad recommendation. Our system, however, takes into account the other places the user has visited in addition to the content, making it capable of knowing that the users who ate pizza and sushi, for example, also tend to go to Mexican restaurants. This additional knowledge makes *Sem* capable of producing better recommendations as can be seen in the graphs.

Another behavior shown in the graphs is that recommending the most popular items is rarely a good option. It is true that its performance increases with the number of user interactions, but this is explained by the fact that, with the increase of interactions, the probability of visiting a popular item (restaurant or POI) also increases. In the POI domain, this method achieves some good results, being some of them at the same level as our system. This is because of the already commented peculiarities of the domain in junction with, as shown in Table 5.6, the immense popularity of some places that are always visited by many tourists.

In short, when we have a situation where the users provide only images of one item, the *Sem* or DenseNet encodings can be used almost indistinctly. If the user have interacted with more than one item in the past, using the embeddings from *Sem* is the best option, while the most popular item should not be recommended under any circumstances.
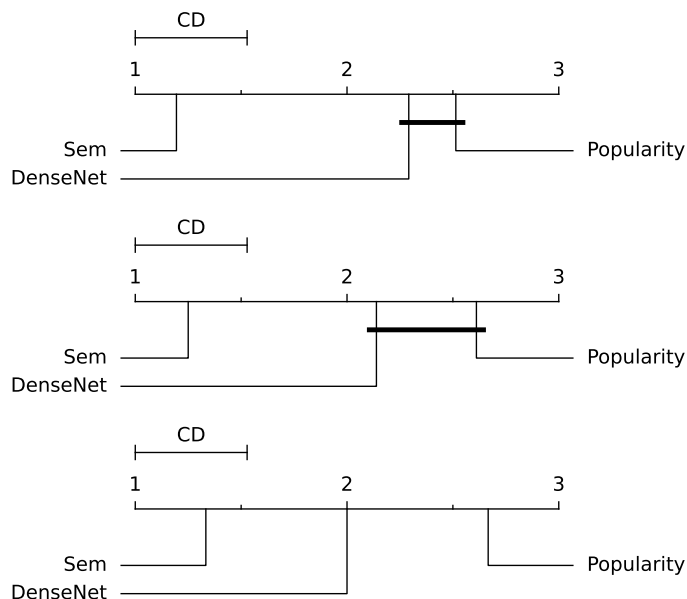
Figure 5.9: *Cold-start* Bonferroni-Dunn tests for Top 1 (up), Top 5 (middle) and Top 10 (bottom). Our system performs significantly better than the others (p-value of 0.05) with a Critical Difference (CD) of 0.53. The other two systems only differ significantly in Top 10.

Figure 5.9 shows the significantly better performance of *Sem* against the other methods, using a Bonferroni-Dunn test. Therefore, we can conclude that *Sem* is the most competitive in *cold-start* situations and can also benefit from the scarce information in these situations.

5.9   Conclusions

This work presents a method to learn a photo mapping (embedding or projection) in $\mathbb{R}^k$. In particular, we deal with photographs taken by users on RS items. From a practical point of view, this means that we have users, items, and relationships between them. First, we have a binary relationship that we can understand as an interaction between users and items. The second relationship that we contemplate is the one given by the photos that users take of items.

The projections we are looking for must be such that, the photos of items with which the same set of users interacted, lead us to very similar vectors in $\mathbb{R}^k$. With the set of photos taken by a user we can build their profile in a RS. Furthermore, this can be done every time a new user appears because their photos lead us to assign them a place in $\mathbb{R}^k$ that represents their tastes.

The same happens with items: the photos that users take of them allow defining a vector with their features. This is a key point as photos allow us to treat the original dataset as a *content-based* RS. Remember that we do not normally have personal information about users beyond their interactions. With no age, job, gender, or any other characteristic related to their tastes, a CF is the only possible approach to building a RS. In this way we can overcome the so-called *cold-start* problem. Also, when the number of interactions is low, collaborative filters cannot provide acceptable returns, but content-based RS can.

To evaluate our proposed method, we used two tourism datasets with different domains: restaurants (six cities) and points of interest (three cities). In both cases, we have verified that the performance of the collaborative filters is dramatically low while we achieve good results with our proposal. Additionally, we compared our semantic approach against a general-purpose embedding given by a pre-trained CNN. The empirical results support our hypothesis: a CNN embedding is not enough to reflect the users' tastes. The method proposed in this paper, *Sem*, take advantage of users to learn a multi-label classification task involving their tastes.

Chapter 6

SummImg: *Summarizing with Images*

Photo summaries are designed to make it easy to navigate through a large volume of images. They must include the relevant aspects of the data source and at the same time cover its diversity. In this chapter we deal with the TripAdvisor restaurant dataset, in particular with the photos users take at the restaurants they visit. We will encode the information contained in the photos following the same procedure as in the previous work (Chapter 5), i.e. taking into account the relationships between users and restaurants. The result will be a reduced set of photographs that will allow, in some way, to reconstruct the behavior of the users by generalizing the original data. To do this, we must be able to capture the essence of user tastes in a summary of the gastronomic offer of a city. The proposal includes an experimental study using data from five cities: Paris, New York, Madrid, Barcelona, and Gijón. The results are overwhelming in favor of the method proposed here for constructing the visual summary.

6.1   Introduction

When dealing with large volumes of data, for example, in RS, it is difficult to distinguish the information that is relevant from that which is not. Valuable information is hidden not only in volume, but also behind an intricate web of relationships. Moreover, these data may have a wide variety of types of information that, in addition to quantitative evaluations, may include opinions expressed with texts and/or photos.

In this work we present a method to build an understandable and useful short description of a specific case of these tangles of data: the restaurants of a city along with the opinions of their customers. The idea is to *explain*, with a simple visual *summary*, the gastronomic offer of a place that can have tens of thousands of restaurants with hundreds of thousands of opinions and photos taken by users. To give this explanation we will use small collections of photos selected for this purpose.

A first characteristic of the summaries that we are going to present is that they are based on a type of clustering that we could qualify as *sociological*, different from those that can be conceived based on content. We are not interested in grouping pizzerias or restaurants with certain regional food. Instead we will consider that two restaurants are *similar* if the sets of users who visit them are also similar. They do not have to offer the same type of food, but they must be *interchangeable in a recommendation* to be visited.

We will work with the photos that users share after visiting a restaurant, not only to understand what caught their attention, but also to generate the final summary. As can be seen, in this work we are going to take the *Sem* coding learnt in the research presented in Chapter 5 further by using it in this case to generate, using images, a summary of the users' tastes. This coding, given its nature, allows us to group photos that were taken in the same (or similar) restaurant, or photos that are visually similar.

Summaries should allow us to draw a visual panorama of a large volume of complex data. The objective is to facilitate the navigation of users who seek to assimilate a large amount of information. For this reason, the reduced number of images that we are going to select must include, on the one hand, the most relevant aspects and, on the other, photographs that represent the diversity of the whole.

This proposal will be illustrated using the TripAdvisor restaurant dataset (Section 2), however, the methods presented could be adapted to other contexts with slight or even no modification. What is essential is to have datasets as used in RS, i.e. to have users, items and user reactions to the items expressed through photographs.

After reviewing some related works in Section 6.2, we present the formal framework of our proposal in Section 6.3. The key point is the representation of the restaurants by the set of users who visited them. This leads us to define the similarity between restaurants as the cardinal of their intersection in Section 6.3.1. The consequence will be that the restaurant clusters will contain, as we wish, interchangeable elements in a list of suggestions to visit. But this will be explained in more detail later.

The next step consists in determining, for each cluster of restaurants, the photos that define them (Section 6.3.2) and that will be used as a visual summary. Here we will need a deep CNN to encode each photo, allowing us to group the photos of similar restaurants and giving coherence to the whole method.

In Section 6.4 we discuss a procedure for evaluating the method devised for summarizing. The key idea is to assess whether the restaurant summary is valid or not (and to what extent) to reconstruct the entire data set; that is, the visiting behavior of users. Using this approach, in Section 6.6 we report an exhaustive set of experiments carried out to check the adequacy of our proposal. Comparisons of the results show the key role of the deep network used to encode the photos, making a significant difference. Finally, Section 6.8 closes the manuscript with the main conclusions.

6.2 Related work

6.2.1 Summarization

Summary algorithms try to find a small subset of objects (sentences, images, videos, sounds) that covers the information of a large set of those objects. The aim is to cover both the diversity of the initial set and the representativeness of what is selected as a summary. They must also eliminate redundancies, as it is essential that the summary be small (Nenkova and McKeown, 2011).

When it comes to images, summaries are a selection of a few images. The purpose is to provide a short description of a collection of images. As in other cases, summaries are useful for making it easier to navigate through a (normally large) collection of images.

Most of the summarization work was done with text documents, see for instance (Gambhir and Gupta, 2017). In this case, the algorithms are usually classified as abstractive (they construct sentences that summarize the content of the document) or extractive (they select some representative sentences). In the case of images, the abstractive approach does not make sense (except perhaps in very special cases). Selective methods remain and, as in the case of texts, the summarizers include some clustering that requires the definition of similarity (Yang et al., 2013a).

Evaluation of summaries

The evaluation of summaries is a controversial issue. In many cases, the evaluation is intended to be carried out through user satisfaction levels or with a relevancy score. In both cases they are not very objective measures. It has even been stated that the lack of consensus somehow slows down progress in this field (Lloret et al., 2018).

When summarizing texts, perhaps the problem of evaluation is more complex. The reason is that the semantics of the sentences of the summary must be compared with that of the original text, which is frankly difficult. An illustrative example of this phenomenon can be seen in (Lloret et al., 2018) where SummEval is presented: a set of resources for summarization and evaluation.

An alternative point of view, which appears especially when it comes to summarizing collections of images, is the *reconstructive* approach. It even also appears when summarizing documents, as in (Chu and Liu, 2019). The idea is to assume that the effectiveness of the summary is reflected by its ability to reconstruct the original set or each individual image in the set (Yang et al., 2013a). In this case, the images are described by means of a dictionary of objects that can appear in them. In some way, each image is represented as sentences using a bag of words approach.

The reconstruction idea is extended to transformer-based encoder and decoder structures, see for instance (Li et al., 2020) dealing with multi-document summarization.

In the context of RS, there is a natural way of understanding reconstruction, which is what we present in this manuscript. The original set records the interaction between users and items. Then, the summary could be evaluated by the degree to which it allows us to reconstruct user behavior.

6.2.2   Summarization in Recommender Systems

As above mentioned, summarizing is closely related to the concept of similarity. The overall idea is to pick one representative element from each group of similar articles. Note that, in RS we may cluster users or items.

The similarity of users (or items) involved in RS has been intensively studied. In (Gazdar and Hidri, 2020) the authors explore some similarity measures for users in order to determine the set of users having the same behavior with regard to a given subset of items.

Let us recall that users and items play a dual role in RS; thus, the similarities can be employed for both entities.

Trying to improve the performance of an RS, (Bag et al., 2019) explore the use of the *Jaccard* similarity. On the other hand, in (Amer et al., 2021) we can find a thoroughly discussion about combinations of similarity functions devised to improve the performance of an RS. However, our point of view in this research is to use similarity to summarize the data collected by an RS. Therefore, similarity is used to cluster the available items.

In (Qian et al., 2019) using *Weibo microblogging* data, the aim is to summarize events using representative texts and images. For this purpose, a co-clustering algorithm is introduced to group text and images considering their relation with users. Images are grouped according to their *visual similarity*.

There is another interesting paper about presentation of the data of an RS (Gil et al., 2018). The authors introduce VisualRS. The objective is to present the information of an RS system in a visual and navigable way, although there is no intention of summarization.

6.2.3 Dealing with restaurants

As mentioned above, in this work we use TripAdvisor data on restaurants in five cities around the world. It is the largest social network about restaurants, hotels and, in general, tourist activities. The photographic information shared on this platform has been previously studied. For example, in (Giglio et al., 2020) the authors used a collection of photographs to understand the perception of luxury by hotel users.

In (Chu and Tsai, 2017), the authors presented a hybrid RS about restaurants. They used visual features to represent both users and restaurants in addition to a collaborative filtering approach. A general purpose CNN was employed to extract features from images with additional ad hoc features. Probably their key point is the method used to deal with several photos, the authors use averaging or maximum aggregations instead of a semantic approach as we introduce in this work.

At (Díez et al., 2020b) we dealt with photos of users taken in restaurants and then shared on TripAdvisor. In that article we focused on authorship: we estimated the probability that a photo was taken by a user. The objective was to provide with each personalized recommendation to a user, the photo that would have been most likely taken by that user.

The photo would then act as an explanation for the recommendation and the interest of the user in the suggestion would be increased. Finally, let us quote (Sun et al., 2019). This paper includes a survey about the use of *side information* on RS. It is interesting in order to obtain a general perspective of the topic.

6.3 Formal framework of the proposal

From the formal point of view, we deal with a set of *users* ($\mathcal{U}$), a set of *restaurants*, and a simple relationship between them: *visited*. We could consider other relationship, such as valuation (implicit or explicit); in that case, we would only have to slightly modify the method described in this work.

As stated in Section 6.1, a key element in this work is a set of photos. Users may or may not contribute photos (one or more than one). We will use them as a fundamental source of communication. A central reflection is that we understand that users take photos (and share them on a social network) of places that especially attract their attention. Therefore, photos carry an important message about the behavior of the users.

For each city we will consider *datasets* whose elements are triples of a user, a restaurant (*rest*), and a list of photos. In symbols,

$$(user, rest, list\ of\ photos). \tag{6.1}$$

Let us remember that all the available information is the data from which an RS is built. Thus, in order to grasp the core idea, it will be especially useful for us to represent restaurants by the sets of users who visited them:

$$\text{representation}(rest) = \big(\text{visited}(u, rest) : u \in \mathcal{U} \big) \in \{0, 1\}^{\mathcal{U}} = \mathbb{Y}. \tag{6.2}$$

This is probably one of the key points of this work. In the experiments reported in Section 6.6, to represent restaurants using (6.2), we will not use the complete set of users $\mathcal{U}$ (it would be unfeasible due to its size), but a sample: 25% of the most active users in each city.

Figure 6.1 depicts the steps of the proposal to obtain the visual summary that is described in the following sections. First, we present how to obtain a selection of groups of restaurants.
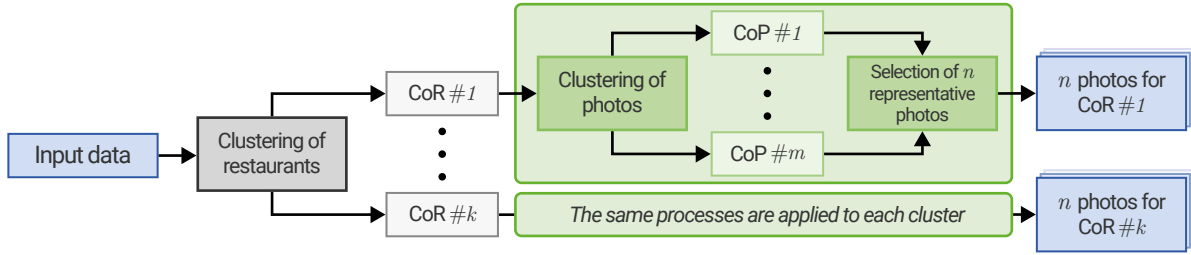
Figure 6.1: We propose this workflow to obtain a visual summary: a reduced set of representative photos for each cluster of restaurants (CoR), see Section 6.3.1. The photos are encoded using the embedding output, $\hat{\mathbb{Y}}$, provided by the neural network architecture shown in Figure 6.2. Finally, the set of photos is chosen from each cluster (green area) following the procedure explained in Section 6.3.2. CoP stands for Cluster of Photos.

To do this, we will define a similarity between restaurants and then we will build a hierarchical clustering. Next, we will see how each cluster can be described through a reduced set of photographs that will constitute the summary we are looking for. Then, we will discuss how the summary can reconstruct the behavior of users, which will suggest a method to evaluate the summary.

6.3.1 Similarity and clustering of restaurants

To build a cluster, we need a *similarity*. In this case we are going to define a function that sets up how interchangeable two restaurants are in a list of recommendations to visit.

We are going to use the following definition. For a couple of restaurants $rest\_1$ and $rest\_2$, their *similarity* (sim) is given by the dot product of their vectorial representation (6.2) or, alternatively, the number of users who visited both restaurants. In symbols,

$$\mathrm{sim}(rest\_1, rest\_2) = \langle rest\_1, rest\_2 \rangle = |rest\_1 \cap rest\_2|. \tag{6.3}$$

Note that this function is different from the *Jaccard similarity* where the above expression is divided by the cardinal of the union. For our proposes this is not a good idea. Let's remember again that the idea is that similar restaurants can replace each other in a list of suggestions, since they are visited by a *similar* set of users.

Using this function, we will build a hierarchical clustering. But we are only interested in the most prominent clusters of similar restaurants. The clustering was performed with an agglomerative algorithm with *linkage complete*, which stops when the merge of the available groups has a similarity below the 5th% percentile of restaurant similarities.

That is, 95% of the similarities yield a cluster merge. Thus, we try to avoid merging groups of restaurants with little similarity. Note that if all the clusters were joined we would end up with only one group, which would not be informative.

6.3.2 Photographs to symbolize clusters

Once we have the clusters of restaurants, we have to find a method of representing them through a small list of featured photographs. Remember that this phase corresponds to the green area in Figure 6.1.

To do this, we need to understand, in a certain sense, the meaning of the photos, and to select the most representative of each group of similar restaurants. These two steps are following described in depth.

Photography embedding

In this case, we will start from the coding learned in the previous work (Chapter 5) applying slight modifications to better adapt it to the problem. We will have, therefore, an embedding for each of the images in a Euclidean space where they will have some semantic meaning. Unlike the previous work, in this case we will not obtain the encoding using the output of an intermediate layer of the model, we will use the output layer ($\hat{\mathbb{Y}}$).

The objective of this encoding is to be able to map each photo to a point close to those assigned to other photos of the same restaurant and to visually similar photos. To achieve this, we propose a neural network (very similar to the one presented in the previous work) that seeks to detect the restaurant where each photo was taken, remembering that each restaurant is represented by a binary vector that encodes the set of users who have visited them (6.2). In symbols,

$$\text{Embedding} : Photos \longrightarrow Restaurants\{0,1\}^{\mathcal{U}} = \mathbb{Y}. \tag{6.4}$$

At this point we can see that the representation of restaurants and the way we define their similarity (6.3) is very important. The embedding assignment could be wrong for a photo by not associating it with the corresponding restaurant. It could fail any of its components.
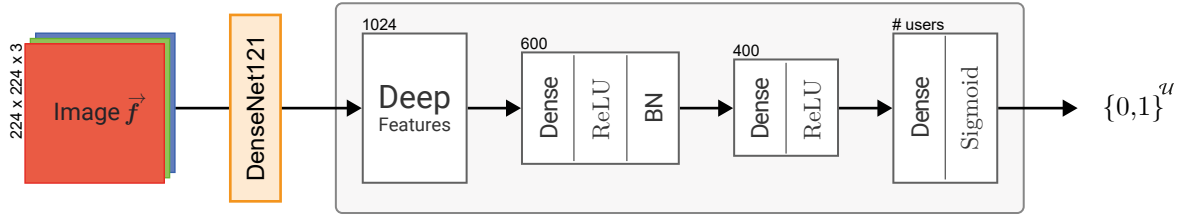
Figure 6.2: Network used to learn the embedding (6.4) from photographs to the set of estimations of codes of restaurants, $\hat{\mathbb{Y}}$.

But we hope that the embedding will associate the photo to a point at $\{0,1\}^{\mathcal{U}}$ very close, not only to its restaurant but also to other similar restaurants; that is, to those of its cluster (see Section 6.3.1).

The embedding of a certain photo is a vector whose components can be thought of as users. In practice, the component of a user u will be the probability that the photo belongs to a restaurant visited by u. These probability vectors belong to a space that we will call $\hat{\mathbb{Y}}$ below. In addition, we will use this symbol to refer to the entire method presented in this manuscript.

$$\text{Embedding}(photo) \in \hat{\mathbb{Y}} \cong Restaurants\{0,1\}^{\mathcal{U}} = \mathbb{Y}. \tag{6.5}$$

Returning to the definition of the embedding (6.4), from a formal point of view, it can be seen as a *multi-label* classifier. Figure 6.2 depicts the deep learning network used to build this function. This network applies the convolutional base of a DenseNet (Huang et al., 2017), pre-trained on the ImageNet dataset (Huang et al., 2017), to convert an input RGB image into a 1024-feature vector. The rest of the architecture is composed of fully connected (Dense) layers of different sizes, along with rectified linear unit (ReLU) (Nair and Hinton, 2010), and batch normalization (BN) (Ioffe and Szegedy, 2015) layers. Finally, a sigmoid activation function is applied to obtain the estimation of restaurant codification.

Finally, we want to highlight that from an abstract point of view we have described how to represent the photos in a Euclidean space where the similarity is given by the scalar product, as in restaurants (6.3).

$$(\hat{\mathbb{Y}}, \langle \cdot, \cdot \rangle) \tag{6.6}$$

Selection of photographs

Let Cl be a cluster obtained following the procedure introduced in Section 6.3.1, and let Ph be the set of photographs of the restaurants of Cl available in the set of triples (6.1).

To select the representative photos of Cl we are going to calculate a clustering of its photos (Ph), see Figure 6.1. We will use the same hierarchical method used to do the restaurant clustering detailed in Section 6.3.1, with the same parameters: an agglomerative hierarchical clustering with *linkage complete*. The algorithm, again, will stop when the merging of available clusters has a similarity less than the 5th percentile of the similarities in Ph, (6.6).

We consider only the n photo clusters with more elements. In each cluster we take the most similar photo (using the dot product, as in (6.3)) to the centroid. In the experiments reported in Section 6.6, we will consider $n \in \{1..5\}$.

6.4   Evaluation

Suppose the summary is a set of photos p_j. To assess its quality, we will measure to what extent it can replace the entire collection of photos. The central idea is to measure to what extent the summary allows us to reconstruct the behavior of users, which in this context will be the list of restaurants visited.

To give a precise formulation we will consider that the users are described as a set of photographs f_i. The reconstruction will be done in two steps. The first is to determine the user's photo that is most similar to one of those in the summary:

$$j* = \underset{i,j}{\operatorname{argmax}} \; Similarity(f_i, p_j). \tag{6.7}$$

The second step consists of associating the user with the cluster of the restaurant where p_{j*} was taken, the restaurant $rest(p_{j*})$. According to the summary, we will understand that the user's habits include the restaurants in that cluster. Let us recall that clusters are built in such a way that their components can be interchangeable in a list of recommendations to visit ((6.2), (6.3)). Therefore, it seems reasonable to accept a cluster as a useful description of users' behavior.

Finally, we define the quality of the summary as the proportion of users for whom it is true that the restaurant cluster (with photos more similar to theirs) contains part of the restaurants visited by users. In symbols,

$$rest(p_{j*}) \cap rest(u) \neq \emptyset.$$

The whole reconstruction scheme can be depicted by

$$u \rightsquigarrow p_{j*} \rightsquigarrow rest(p_{j*}) \rightsquigarrow cluster.$$

As can be seen, the definition of similarity plays a fundamental role. In fact, as we indicated at the end of the previous section, we consider that a method to extract a summary of the set of photos must contain a space (in which to embed the photos) and a definition of similarity between photos in that space.

6.5 Datasets

In this work we will use the TripAdvisor restaurant dataset (Chapter 2), specifically data from five of the cities. Specifically, we use three Spanish cities: Barcelona (population: 1.6 million) and Madrid (population: 3.2 million), the two largest in the country; and Gijón, a medium-sized city of about 300,000 inhabitants. We also used data from other large cities around the world, such as New York (8.3 million), and Paris (2.1 million). The table below shows the figures for the cities after applying the data cleaning explained below.

	All data				Training data			
	#Users	#Rest.	# Reviews	#Photos	#Users	#Rest.	# Reviews	#Photos
Gijón	4,203	373	6,475	14,241	2,297	373	3,917	8,748
Barcelona	25,230	3,236	49,024	105,638	14,339	3,236	32,229	70,842
Madrid	33,222	3,707	65,191	141,649	18,512	3,707	41,714	92,287
New York	43,581	3,733	74,806	130,992	24,147	3,733	46,392	82,886
Paris	46,794	6,764	88,406	171,296	27,648	6,764	60,260	118,017

Table 6.1: Basic statistics of the dataset used in our experiments.

The raw data was cleaned up by applying some filters: firstly, we keep only the most recent review for each pair user/restaurant.

If a user reviewed a restaurant several times, we filter out the oldest reviews. We also removed all photos without food.

We used the 25% more active users (those with the largest number of reviews) to build the vectorial representation (6.2) of the restaurants. Next, we eliminated restaurants with less than five users (from those used for encoding) and also those that did not have at least five photos.

The clustering of restaurants described in Section 6.3.1 was carried out with the resulting data. Then, we split the dataset of each city into a training and a test set. As explained in Section 6.4, we will estimate the ability of our approach to summarize the gastronomical offer of a city. Thus, we split the data with respect the users, so that the information in the test set corresponds to users never seen during the training stages. The split is made to retain approximately 50% of users in each partition and guaranteeing that all the restaurants appearing in the triplets (6.1) of the test set also appear in the training set.

6.6 Experiment description

In this section we will describe the experiments performed in order to test the performance of our system. Let us recall that the evaluation (see Section 6.4) is measured in terms of accuracy, considering a hit case when the assigned cluster contains at least one restaurant visited by the user. All methods start from the grouping of restaurants as explained above.

Our proposal ($\hat{\mathbb{Y}}$) summarizes each group of restaurants by selecting n images. Then, each user in the test set is assigned the group with the photo most similar to those provided by the user, (6.7).

We compared the performance of $\hat{\mathbb{Y}}$ with several variants obtained by ablation of its two main components; that is, removing the network to encode the photos and using DenseNet image encoding directly ($\mathbb{D}$), replacing the cluster selection method with a random choice ($\hat{\mathbb{Y}}_{rnd}$), or applying both modifications ($\mathbb{D}_{rnd}$). Table 6.2 summarizes the scores achieved by all these variants.

On the other hand, we also tested a baseline approach ($\mathbb{B}$) that simply selects the cluster of restaurants with the largest number of photos in the training set, thus not depending on any image encoding.

	Photo encoding	Cluster selection
Our proposal ($\hat{\mathbb{Y}}$)	Network output	Highest inner product
Variant 1 ($\hat{\mathbb{Y}}_{rnd}$)	Network output	Random selection
Variant 2 ($\mathbb{D}$)	DenseNet vector	Closest (Euclidean distance)
Variant 3 ($\mathbb{D}_{rnd}$)	DenseNet vector	Random selection
Baseline ($\mathbb{B}$)	N/A	Largest cluster

Table 6.2: Different approaches for the assignment of a cluster of restaurants to a user. The baseline approach assigns the cluster with the largest number of photos, so it does not depend on their encoding.

In a sense, this method uses a kind of *popularity* measure to assign the cluster of restaurants.

In order to test the robustness of our approach we tried a range of values for some parameters of the experiments. Thus, we run the experiments considering a range of photos to represent each cluster, $n = [1..5]$.

With respect to the training of $\hat{\mathbb{Y}}$, we used a grid-search on the training dataset of Barcelona that yielded a learning rate $\alpha = 5 \cdot 10^{-4}$ with linear decay down to $1 \cdot 10^{-5}$, a batch size $b = 1024$, and the weights for the weighted loss $w_0 = 1$ and $w_1 = 5$. The network was trained using an early stopping strategy with a maximum of 4000 epochs.

6.7 Results

Table 6.3 presents the results obtained for each of the five cities studied. Remember that the evaluation method is described in Section 6.4 and represents the proportion of times that the models are able to reproduce the behavior of the users of the test set. To ease the reading of these results, we have expressed these proportions as percentages.

In the first column of the table appears the name of the city. The table is split in two parts. The left hand side shows the scores obtained when the photos were selected using the clustering method. The right part reproduces the results when the photos were randomly chosen. See Section 6.3.2. As can be observed, the clustering method is clearly superior.

Each row shows the results for a dataset. We have distinguished the scores obtained with users who have at least one photo in the test set (≥ 1), from those that contain two or more, above three or at least for.

The column labeled by "k" indicates the number of cases in each situation in the test.

The following columns gather the scores by the number of images of each cluster that make up the summary, see Section 6.3.2. An improvement can be seen for each model by increasing this number (left to right in the table). The exception is the $\mathbb{B}$ model that chooses the largest cluster without further consideration and is, therefore, independent of the number of photos in the summary.

Another reason to improve the scores is the number of photos available from each user (column "k"). The more knowledge we have of users, the better we are able to reconstruct their behavior. In the table we can see this from top to bottom.

The scores show that the image encoding is of crucial importance for the task at hand. Our proposed encoding, $\hat{\mathbb{Y}}$, maps each photo into a space taking into account the users who visited the restaurant where it was taken. In a sense, the model obtained by our neural network architecture generalizes the latent features of the gastronomic offer of the restaurant, and that made an specific group of users to visit it.

The DenseNet encoding ($\mathbb{D}$), on the contrary, has nothing to do with the taste of users, it is an encoding devised to achieve good performance in general purpose computer vision tasks, as object recognition. We used the DenseNet as a starting point but our posterior processing has proven to be essential in order to achieve an adequate summary of representative images of restaurants regarding the users' tastes.

City	k	$\mathbb{B}$	USING PHOTO CLUSTERING										RANDOM SELECTION OF PHOTOS									
			1		2		3		4		5		1		2		3		4		5	
			$\mathbb{D}$	$\hat{\mathbb{Y}}$	$\mathbb{D}$	$\hat{\mathbb{Y}}$	$\mathbb{D}$	$\hat{\mathbb{Y}}$	$\mathbb{D}$	$\hat{\mathbb{Y}}$	$\mathbb{D}$	$\hat{\mathbb{Y}}$	$\mathbb{D}_{rnd}$	$\hat{\mathbb{Y}}_{rnd}$	$\mathbb{D}_{rnd}$	$\hat{\mathbb{Y}}_{rnd}$	$\mathbb{D}_{rnd}$	$\hat{\mathbb{Y}}_{rnd}$	$\mathbb{D}_{rnd}$	$\hat{\mathbb{Y}}_{rnd}$	$\mathbb{D}_{rnd}$	$\hat{\mathbb{Y}}_{rnd}$
Gijón	≥ 1	36.1	3.4	35.3	6.3	33.6	9.5	39.6	11.2	41.3	12.6	42.0	2.8	20.1	2.5	27.4	3.1	29.7	3.1	31.1	3.7	32.0
	≥ 2	58.6	6.6	53.2	10.0	50.5	11.2	57.7	13.9	59.2	15.7	59.2	2.1	34.7	2.4	44.1	3.9	45.6	4.2	46.8	4.2	48.0
	≥ 3	69.6	9.6	65.6	12.0	64.0	12.8	70.4	18.4	71.2	20.0	71.2	3.2	48.0	4.0	55.2	5.6	56.8	4.0	56.8	4.8	56.8
	≥ 4	73.8	15.4	72.3	16.9	70.8	16.9	76.9	18.5	76.9	21.5	76.9	3.1	55.4	3.1	58.5	4.6	56.9	3.1	56.9	4.6	56.9
Barcelona	≥ 1	4.6	1.9	9.7	2.8	12.9	3.3	14.2	3.8	15.1	4.1	15.3	1.0	3.3	1.2	6.8	1.5	8.3	1.8	9.6	1.9	10.4
	≥ 2	10.5	2.6	16.1	3.8	22.1	4.2	24.2	4.6	25.8	5.1	26.1	1.4	5.8	1.6	12.7	2.1	14.4	2.4	17.0	2.7	18.3
	≥ 3	14.8	2.9	19.1	3.9	27.2	4.1	29.7	4.5	31.9	5.5	31.9	1.8	7.2	1.8	16.0	2.3	16.9	2.7	19.9	3.3	21.4
	≥ 4	17.4	3.8	20.6	4.7	30.4	5.2	34.3	5.5	37.0	6.5	37.2	0.9	7.3	1.4	16.8	2.4	17.9	2.8	22.2	3.8	23.1
Madrid	≥ 1	12.0	2.3	9.4	3.5	13.2	4.1	14.5	4.6	15.2	4.9	16.1	0.8	1.2	0.9	6.3	1.1	5.5	1.6	5.3	1.8	5.5
	≥ 2	22.5	3.7	16.3	5.0	22.6	5.8	24.2	6.3	25.4	6.6	27.2	0.9	1.2	0.8	10.6	0.9	9.4	1.8	9.4	2.2	9.4
	≥ 3	31.1	4.8	19.3	6.6	26.8	7.0	29.9	7.8	31.5	8.1	33.9	1.0	1.3	0.9	12.2	1.2	11.4	1.9	11.4	2.4	11.2
	≥ 4	36.0	6.6	21.8	8.4	30.8	8.9	33.9	9.2	36.0	9.6	38.7	1.3	1.1	1.2	14.0	1.3	13.0	1.8	13.0	2.1	12.7
NYC	≥ 1	9.6	3.8	13.0	4.5	16.4	5.4	18.1	6.2	18.0	6.7	18.6	1.6	8.6	1.8	10.7	2.0	13.1	2.1	12.8	2.2	16.2
	≥ 2	17.6	5.2	20.3	5.8	24.9	6.7	26.5	7.1	25.8	7.7	26.8	1.9	14.6	1.8	17.8	2.1	21.5	2.4	20.9	2.4	26.3
	≥ 3	22.9	5.8	25.3	6.4	30.9	7.7	32.9	8.2	31.8	9.0	32.7	2.0	17.7	2.0	21.3	2.5	25.1	2.7	24.8	3.0	31.0
	≥ 4	28.0	5.7	29.4	6.1	34.9	7.5	36.6	8.0	35.3	8.7	35.9	2.0	21.5	2.5	25.6	3.2	29.2	3.3	29.0	3.8	35.5
Paris	≥ 1	3.0	1.8	9.4	2.5	10.9	2.9	11.8	3.1	11.9	3.3	12.2	0.7	2.6	0.9	5.8	1.1	6.7	1.4	7.5	1.7	7.7
	≥ 2	6.7	2.7	17.0	3.5	19.0	3.7	20.9	3.9	21.1	3.9	21.6	0.7	4.0	1.0	10.8	1.0	11.3	1.2	12.7	1.6	12.9
	≥ 3	9.4	3.2	21.6	3.8	24.3	3.9	26.7	4.2	26.9	4.3	27.7	0.8	4.4	0.9	13.7	0.9	14.0	1.0	15.9	1.4	15.6
	≥ 4	12.7	3.2	25.9	3.8	29.1	3.8	32.2	4.0	32.1	4.1	33.8	0.5	4.3	0.8	16.5	0.9	17.4	0.9	18.7	1.8	18.3

Table 6.3: Experimental results in percentage (see Section 6.4). The scores were obtained with data of the five cities indicated in the leftmost column. To check the robustness of the procedure, we split the test elements by the number of photos available of each user (column "k"), as well as by the number of representative photos for each cluster, varying from 1 to 5 (numbered columns). We also compared two methods for the selection of the representative images in each cluster of restaurants: by clustering of images (left half of the table) and by random selection (right half). The results obtained in each configuration for these approaches are labeled as: the baseline procedure ($\mathbb{B}$), using clustering or random selection of images encoded with Densenet ($\mathbb{D}$ and $\mathbb{D}_{rnd}$ respectively) or encoded by our proposed neural network ($\hat{\mathbb{Y}}$ and $\hat{\mathbb{Y}}_{rnd}$ respectively); see Section 6.6.
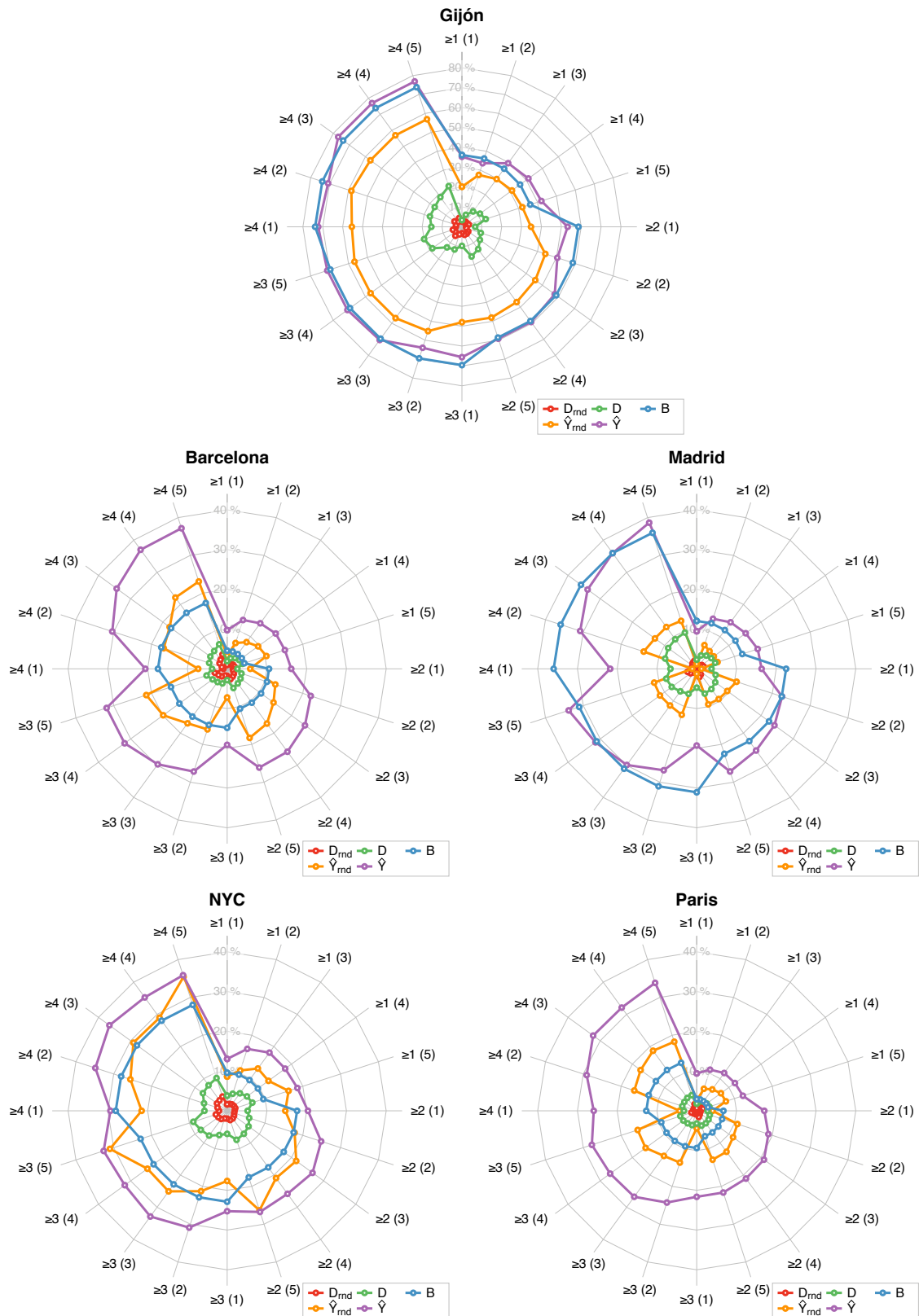
Figure 6.3: Radar charts of the results of each city, see Section 6.4. The axis represent number of photos. For example, "≥ 2 (3)" stands for the scores achieved for users with at least 2 photos and summaries built with 3 photos for cluster.
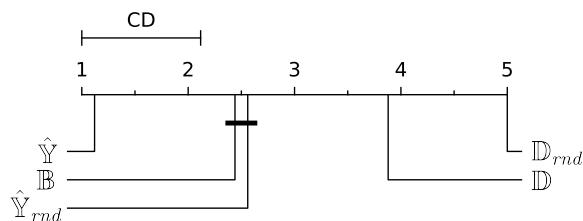
Figure 6.4: Bonferroni-Dunn test with $\alpha = 0.05$.

Clearly, the best scores are achieved by $\hat{\mathbb{Y}}$, attaining the best performance in almost all the cases, while $\mathbb{D}_{rnd}$ is the worst approach. We carried out a Bonferroni-Dunn test with $\alpha = 0.05$ to test whether the differences between these approaches were statistically significant. The results of the test, graphically depicted in Figure 6.4, indicate that all differences are statistically significant except those between $\mathbb{B}$ and $\hat{\mathbb{Y}}_{rnd}$.

The surprising good results of the majority model ($\mathbb{B}$) are due to the fact that in some cities the visits of the customers are not uniform. This is the case of Gijón (due to its small size) and Madrid (with a lot of tourists who follow the advice of the guides and visit the same places). To graphically illustrate the results, we have devised one radar chart for each city (Figure 6.3).

6.8   Conclusions

In this work we present a method to summarize the information of an RS dataset. In this case, the interactions between users and items include photos that will play a key role. In fact, the summaries we build are a reduced set of photos that contain the condensed information from the dataset. To illustrate our method, we have used restaurant datasets taken from TripAdvisor. In this context, the visual summary that we created supposes a short description, in a few photos, of the gastronomic offer of a city. When dealing with summaries, it is not trivial to establish the evaluation method that should be used to measure their quality. In this case, we have chosen to contrast the ability of the summaries to be able to reconstruct and generalize the gastronomic behavior of the users.

We have carried out an experimental ablation study of the components of the proposed method. The result is that performance plummets if we skip any of the steps detailed in the manuscript. The key piece of the proposal is the encoding of users' photos. For this purpose, we use a deep network that takes into account not only the visual characteristics of the photos, but also the relationship between the restaurants where they were taken and the users who visited them.

We think the approach introduced can be useful in the treatment of RS datasets with multimedia elements that arise from the interaction between users and items. The definitions of similarity used, which is the centerpiece, can be extended to other types of data with relative ease.

Chapter 7

VisualRec: Visual-based recommendation

Adding up all that has been learned in previous chapters, in this chapter we present a brief work were we pose a restaurant recommendation mobile application. Its aim is to recommend places that serve a similar type of dish to the one a user indicates by uploading an image. This is useful when we travel to another city and need recommendations of establishments where we can taste a dish we particularly like. By uploading an image of the dish in the application, we would obtain a list of recommended restaurants in the area almost immediately. As in most cases we do not have photos of our favorite dishes, or we simply want to be advised, the application will also show us a series of images with suggested dishes, which mainly highlight those typical in the gastronomy of the current location, which may be unfamiliar to the user and at the same time appetizing. To carry out this process, several Convolutional Neural Networks will be created and used to determine whether or not there is food in a photo, to extract the main characteristics of the dish from the image and then to recommend a list of restaurants.

7.1 Description of the application

The main objective of the mobile application proposed in this work is to offer restaurant recommendations to users who arrive in a new city for tourism, work or other reasons. Traditionally, existing applications recommend restaurants based on a textual search or based on similar sites visited previously. In our case, the user does not have a history nor does he/she have to know the names of restaurants or traditional dishes of the place, therefore, this traditional approach would not be valid. To solve the problems described above, our application will have as a novel element the way of obtaining these recommendations, since the starting point will be a single image provided by the user as a search term. This functionality is intended to emulate a user request of the type *"I want to eat something like the dish shown in the picture"*. Figure 7.1 depicts the steps followed by our application.
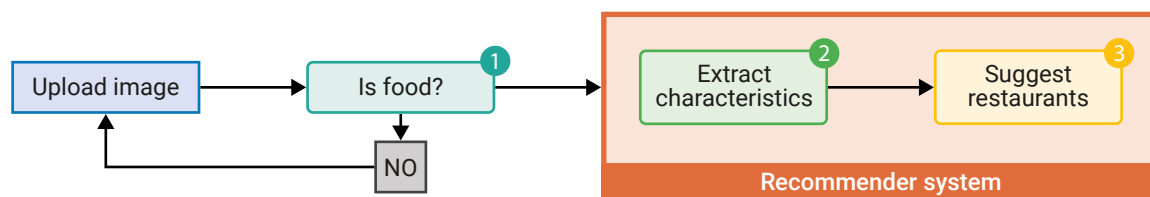
Figure 7.1: Restaurant recommendation procedure from the uploading of an image to the final recommendation.

The user not only will be allowed to use their own images, but also different example images will be suggested on the main screen of the application. These images will be grouped by different criteria (local food, Italian food, fast food, . . .) in order to facilitate the recommendation for cases where the user does not have images, does not know the names of the dishes or simply for inspiration. It is important to highlight the criterion *local food*, as it includes a series of images of dishes that should vary according to the user's geographical location, always showing typical local dishes, which is very useful if the local gastronomy is unknown.

The application will also provide access to information published on the Internet about the recommended restaurants. Thus, the user will be able to quickly view pictures taken by other customers of the restaurant as well as read their reviews. The application will recommend a list of restaurants to the user based on the content of the image. To do this, it will be necessary to know which are the most typical dishes of each establishment and once we know them, we will recommend only those restaurants specialized in the dish or content of the image given by the user following a procedure like the one represented in Figure 7.2.
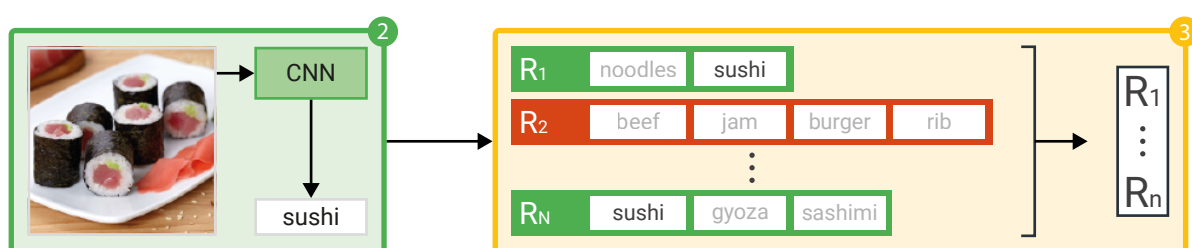

Figure 7.2: Details of the procedure used for the generation of recommendations. First the content of the user image is extracted using a Convolutional Neural Network. Then, we recommend the restaurants where the same content or dish is offered. Please note that this two steps correspond to the last ones in Figure 7.1.

The idea for the creation of this application arose after noticing that many of the mobile apps we use today have image search capabilities, but, to our knowledge, none were known to exist in the field of restaurants. Recently this year there have been some advancements in this regard by Google, but there is not much information about it (Google I/O'22).

Examples of such applications include Google Lens, AliExpress or Amazon which, using this technology, allow finding web results or consumer goods using only the image provided by the user.

7.2 Proposed system

In order to achieve the functionalities described above, it is necessary to use the following AI techniques:

- Convolutional Neural Networks:

    1. Classify image in food / no food.
    2. Obtain the content or dish of the input image.

- Recommender systems

    3. To generate, from an image, a list of recommended restaurants.

The usefulness of these techniques during the recommendation process can be seen in Figure 7.1, each of the above points corresponding to the blocks of the recommendation process. We have decided to create a different model for each of the three desired functionalities. Each of them will be detailed in the following.

7.2.1 Model 1: Classify image in food/no food

Initially, our application has to verify that the user has entered an image with food, this is necessary to avoid erroneous or confusing recommendations by the system. Therefore, we have to create a model capable of processing images at the input and solving a binary classification problem at the output, i.e. predicting whether or not there is food in the image.

We have chosen to use an existing Convolutional Neural Network to which transfer-learning (Yang et al., 2013b) and fine-tuning will be applied in order to avoid creating an architecture from scratch. This implies taking advantage of an existing architecture that has good results solving similar problems and then adjusting it for our specific problem.

Considering that our models have to run on a mobile device as fast as possible, we have chosen to use an existing network with few layers and parameters designed to run on mobile devices. The network in question is *MobileNetV3Small* (Howard et al., 2019) and from it, we will only keep the convolutional part (feature extraction) to which we will add a series of layers in order to solve the binary classification problem. The resulting model can be seen in Figure 7.3.
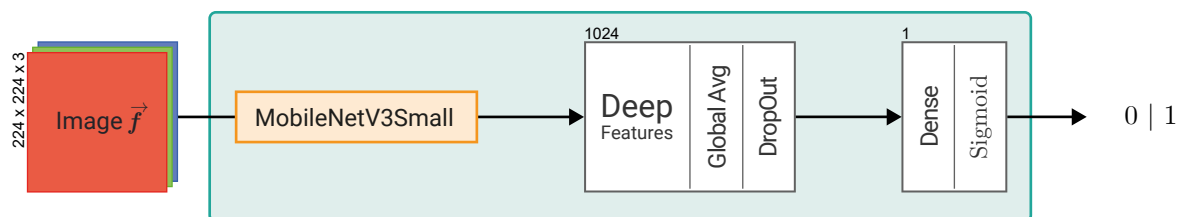
Figure 7.3: First model architecture. The input 224×224 color images are processed using the comvolutional part of the MobileNetV3Small. The output is transformed using a series of layers in order to reduce the dimensionality, control the overfitting and obtaining the final prediction respectively. *Global Avg* stands for a Global Average Pooling. The size of each layer can be seen on its side.

Once we have defined the architecture of the model, we need a dataset that allows us to train it and that should be formed by pairs of the type:

$$(image, \ food?). \tag{7.1}$$

In this case we are going to manually label the more than 18000 images of the restaurants for the city of Gijón (Chapter 2). After this process we will know the restaurant where these images were taken and if they have food or not. The dataset has been divided into training, validation and test in order to be able to perform various hyper-parameter tests and, finally, the best combination of hyperparameters was used to generate and export the model to be utilized in the mobile application.

It should be noted that this model will only be used with those images uploaded by the user through their gallery, the images suggested by the application do not require to pass this filter, as they have been previously selected to ensure that they all contain food.

7.2.2 Model 2: Extraction of food types from the image

After making sure that all the images we work with are of food, we can continue with our goal: recommending restaurants based on the type of food contained in a given image.

To achieve it, we need to know, not only the type of food contained in the image provided by the user, but in which restaurants that style of food is served (see Figure 7.2). This information will allow us, depending on the type of food that appears in the input image, to select the most suitable restaurants to be recommended.

The easiest way to obtain the plate of food present in an image is to train a model to solve this task. To create it, we will follow the same philosophy as in the previous case, taking advantage of an existing CNN and modifying the final part to adapt it to our problem.

As can be seen in Figure 7.4, the only difference with the previous model lies in the output layer, where we now intend to solve a multi-classification problem instead of the previous binary classification.
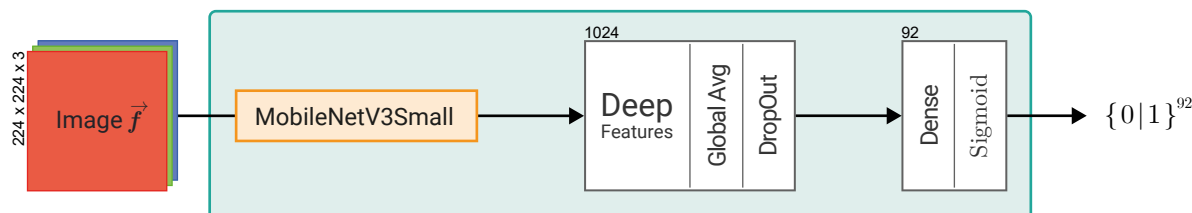

Figure 7.4: Second model architecture.

To train this model, a set of 92 different types of food has been created, each containing approximately 300 images. Initially, a list was defined with all the types of food that we wanted to learn, taking into account the gastronomic diversity all around the world. Finally, we develop a software capable of download images from Google for each type of food on the list, which made possible to create a dataset in a simple way. As in the previous case, the data were divided into training, validation and test and, after hyper-parameter testing, the model was trained with the best known combination.

From this point we are already able to know the type of food in an image (step 2 in Figure 7.2), but we still need to know the same information for each of the restaurants in the city (step 3). Using the above model on the dataset used in Section 7.2.1 we can extract this knowledge using Algorithm 1.

Algorithm 1: Classes of food in each restaurant.

forall *restaurant r* **do**
  Remove non-food images from r with Model 1;
  forall *image i from r* **do**
    Predict, using Model 2, the food classes in i;
  end
  Select the most frequent classes in r;
end
Result: {(restaurant, popular food classes)}

Once the classes of each restaurant are known, we can make recommendations following the process shown in Figure 7.2. Bearing in mind that our application is going to be executed on a mobile device, it is not ideal to carry out so much workload on it, so we have chosen to create a third model in charge of returning the list of the most likely restaurants directly from an image, thus avoiding the whole intermediate procedure.

7.2.3   Model 3: Obtain restaurant recommendations

Taking advantage of the information extracted in the previous step, we can create a dataset that will be used to train the final system. This new dataset will be created from the dataset used in the first model, i.e. the one that was formed by pairs as indicated in Equation (7.1).

Since our objective is to know, for each of the images in the dataset, a list of restaurants where the dish in the image is served, it is necessary to transform this dataset using Algorithm 2.

Algorithm 2: Final model dataset creation.

Remove non-food images with Model 1;
forall *image i* **do**
│ Extract food types using Model 2;
│ Obtain restaurants where the same food is served;
end
Result: {(image, restaurant list)}

Finally, we will need a model capable of learn these relationships between images and restaurants, which we will solve following the same philosophy as in the previous scenarios. It must be taken into account that in this case we will try to solve a multi-label problem, since each image in the input may be associated with more than one restaurant at the output of the model, specifically with all those where the dish in the image is served.

Once the model has been trained, we can export it to the mobile device where, after running it for an image, it will return a list of the most likely restaurants.

7.3   Mobile application

In Figure 7.5 we can see the essential sections of the application. The screenshot on the left-hand side shows the main screen, where we can see different suggestions (local food, Asian food, . . .) as an inspiration. The idea is that, depending on the user's location, the local food suggestions will be tailored to the specific geographical area. The rest of the suggestions do not require any modification and could remain static for the rest of the locations.

When we click on one of the suggested images (sushi in this example), the application runs Model 3 on the image, obtaining a list of restaurants as shown in the screenshot on the right-hand side of Figure 7.5. In addition to seeing the recommendations, we can also sort them by affinity (probability returned by the model), stars, name or proximity. If we click on any of the recommended restaurants, the application will open the restaurant's TripAdvisor website, where we can consult more information about the establishment.
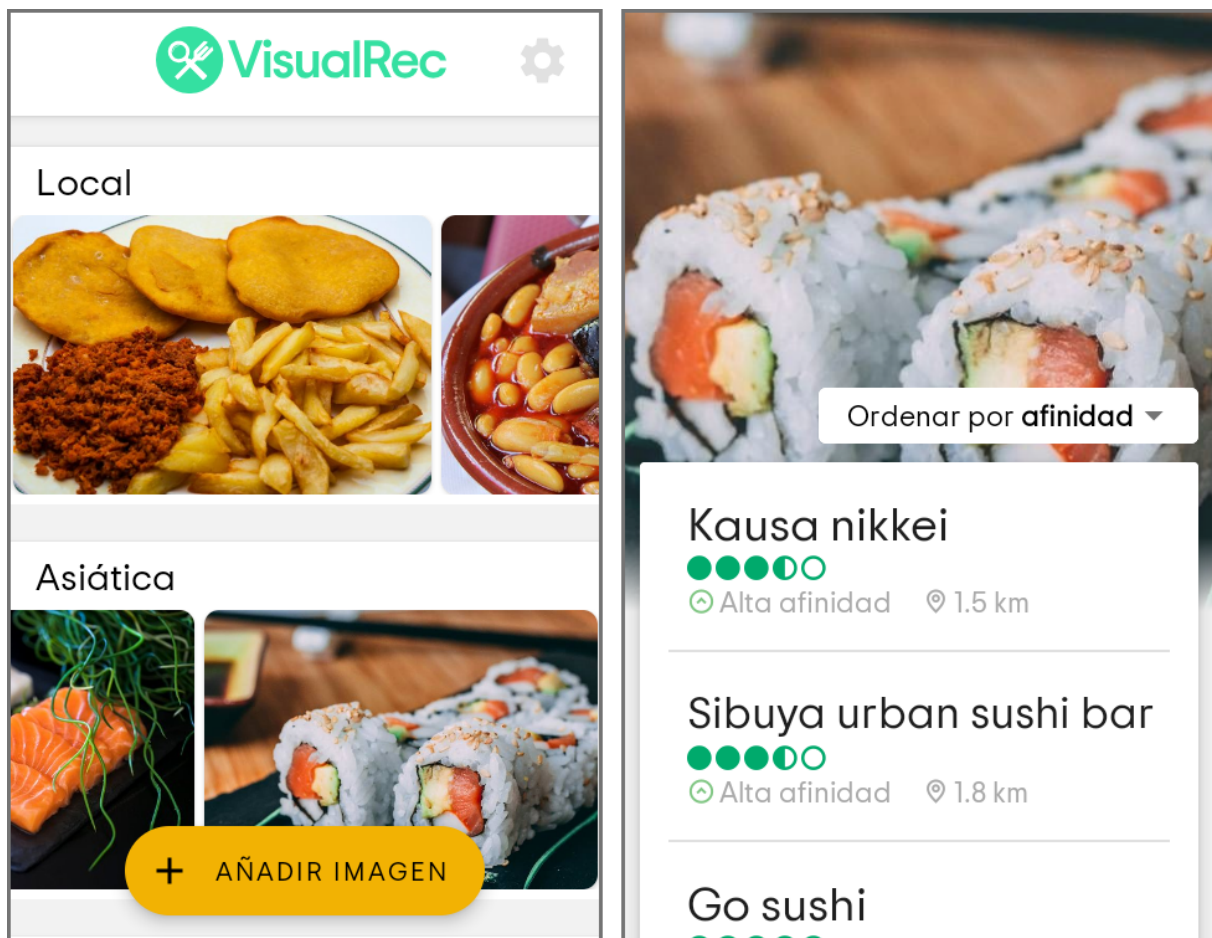
Figure 7.5: Real application screenshots.

If we decide to upload a photo from our gallery instead of choosing an image from among those suggested, the application, before showing us the list of recommendations directly, applies Model 1 on the image in order to verify that it contains food. If it does not, a warning message is displayed and, if it does, the recommendation is made following the same steps as described above.

7.4   Technologies and tools used

The application has been developed for the Android operating system (given the accessibility of its tools), therefore the development software Android Studio will be used. To train both the Convolutional Neural Networks and the recommendation system, we used the Keras (Chollet et al., 2015) library on TensorFlow (Abadi et al., 2016) in the programming language Python (Van Rossum and Drake, 2009).

Once the models had been trained and, in order to be able to perform inference on a mobile device, they were transformed into the appropriate format to be able to run them using the Android TensorFlow Lite library, applying the default optimisations to simplify them and reduce their size.

7.5 Conclusions and future work

In this chapter we have presented an application capable of making restaurant recommendations based simply on a photo of food, which may have been provided by the user or suggested by the application itself.

To achieve our goal, we have created and trained models from scratch in order to solve our specific problems. In total, three different networks have been created and trained, which indicate, given an image, whether it has food, the type of food and finally the restaurants where the food is served. In order to train each of the models, an appropriate data set for each problem had to be created.

In addition, a recommendation algorithm has been devised which, based on the content of the image, is capable of searching for the restaurants that are most closely related to the dish in the image.

As future work, the first improvement proposed is to reduce the number of models by combining 1 and 2 in a single system, so that we would have a single model capable of indicating whether or not the photo has food in it and, if so, telling us its type.

Another improvement would be the automation of local food suggestions. Currently, these are created manually for the specific city you are working with, but it is thought that they could be obtained automatically based on the location. Finally, in order to add some transparency, the featured dishes of each restaurant could be indicated in the list of recommendations, so that the user could verify the accuracy of the system's recommendations.

Part IV

Publications and conclusions

Chapter 8

Publications

All the works presented in this thesis have been published or submmited to specialized journals and conferences. The details of each submission/publication are shown in the following section.

8.1 Works presented in this thesis

8.1.1 TReX : Text-based Recommender with eXplanations

Sistema de Recomendación con Explicaciones Basadas en Texto

- **Authors:** Pablo Pérez, Antonio Bahamonde, Oscar Luaces & Jorge Díez
- **Conference:** CAEPIA (Conference of the Spanish Association for AI)
- **Acceptance year:** 2021
- **Status:** Published and presented
- **Details:** Preliminary version of the work presented in this document.

Text-based Recommender System with Explanatory Capabilities

- **Authors:** Pablo Pérez, Antonio Bahamonde, Oscar Luaces & Jorge Díez
- **Journal:** KAIS (Knowledge And Information Systems)
- **Impact factor (2021):** 2.531
- **Rank (2021):** Q3 in Computer Science, Artificial Intelligence & Q3 in Computer Science, Information Systems
- **Submission year:** 2022
- **Status:** Under review
- **Details:** Work presented in Chapter 3. Extension of the previous work.

8.1.2   ELVis: Explaining Likings Visually

Towards explainable personalized recommendations by learning from users' photos

- **Authors:** Jorge Díez, Pablo Pérez, Oscar Luaces, Beatriz Remeseiro & Antonio Bahamonde
- **Journal:** Information Sciences
- **Impact factor (2020):** 6.795
- **Rank (2020):** Q1 in Computer Science, Information Systems
- **Acceptance year:** 2020
- **Status:** Published
- **Details:** Work presented in Chapter 4.

8.1.3   Sem: Semantics of Images

Users' photos of items can reveal their tastes in a recommender system

- **Authors:** Jorge Díez, Pablo Pérez, Oscar Luaces, Beatriz Remeseiro & Antonio Bahamonde
- **Journal:** Information Sciences
- **Impact factor (2021):** 8.233
- **Rank (2021):** Q1 in Computer Science, Information Systems
- **Submission year:** 2020
- **Status:** Under review
- **Details:** Work presented in Chapter 5.

8.1.4   SummImg: Summarizing with Images

Summarizing with Photos the Items in a Recommender System

- **Authors:** Pablo Pérez, Jorge Díez, Beatriz Remeseiro, Oscar Luaces & Antonio Bahamonde
- **Journal:** Pattern Recognition
- **Impact factor (2021):** 8.518
- **Rank (2021):** Q1 in Computer Science, Artificial Intelligence
- **Submission year:** 2021
- **Status:** Under review
- **Details:** Work presented in Chapter 6.

8.1.5   VisualRec: Visual-based recommendation

Recomendación de Restaurantes Basada en Contenido Visual

- **Authors:** Pablo Pérez, Cristina Cuesta & Jorge Díez
- **Conference:** CAEPIA (Conference of the Spanish Association for AI)
- **Acceptance year:** 2021
- **Status:** Published and presented
- **Details:** Work presented in Chapter 7.

8.2   Related works developed during the thesis

8.2.1   RecSys 2020 Doctoral Symposium

Taking Advantage of Images and Texts in Recommender Systems: Semantics and Explainability

- **Authors:** Pablo Pérez
- **Conference:** RecSys (The ACM Conference on Recommender Systems)
- **GGS Rating:** Class A
- **Acceptance year:** 2020
- **Status:** Published and presented
- **Details:** Early stage of this thesis.

8.2.2   User encoding for clustering in sparse RS

User Encoding for Clustering in very Sparse Recommender Systems tasks

- **Authors:** Pablo Pérez, Jorge Díez, Oscar Luaces & Antonio Bahamonde
- **Journal:** Multimedia Tools and Applications
- **Impact factor (2021):** 2.577
- **Rank (2021):** Q2 in Computer Science, Theory Methods
- **Acceptance year:** 2021
- **Status:** Published

8.2.3   Transparency and Scrutable Movie Recommendation System

Sistema de Recomendación Transparente y Escrutable para Recomendaciones Personalizadas

- **Authors:** Antonio Rodriguez, Pablo Pérez, & Jorge Díez
- **Conference:** CAEPIA (Conference of the Spanish Association for AI)
- **Acceptance year:** 2021
- **Status:** Published

Chapter 9

Final conclusions

Recommender Systems are of vital importance in an extremely connected world where the amount of information generated by each user increases constantly. The vast majority of platforms that we use every day have some kind of recommender that helps us distinguish between the irrelevant content and the one that really matters. Thanks to them, users obtain greater satisfaction and companies obtain higher revenue (among other benefits).

These systems are traditionally trained by making use of historical user interactions (purchases made, songs or videos played, news read, ...) since there is usually no other information available. Currently, many platforms also allow users to extend their opinions with text and/or images (unstructured information), which we believe can be used when creating or improving a Recommender System.

After an introduction (Chapter 1) to the basic terms and problems we wish to solve in this thesis, in Part I we create and analyze several datasets which included unstructured information in the form of text and images, based on public reviews from the TripAdvisor website. The works developed in the scope of this thesis are detailed in Parts II and III, the former devoted to take advantage of textual information and the latter to those using information contained in images.

The first work presented (Chapter 3) poses a text-based Recommendation System where explainability and transparency take on special importance. To achieve this, we start from a simple neural network with a single hidden layer and no activation function that, once trained, allows not only to recommend restaurants with a high hit rate, but also to yield explanations to justify the origin of these recommendations to the end user. The results show that our proposal performs almost as well as more complex options that do not have explanation capabilities.

The second work of this thesis (Chapter 4) presents a system capable of predicting the authorship of given photo. Using this system, when a user enters the website of a restaurant, we can show her only those photos taken by other users that could have been taken by her (they have a similar style or content to the ones the user has already taken).

This personalized selection of images will make the user even more convinced of a given recommendation. The results show that our system is able to identify the author of a photo with good accuracy despite the wide range of possibilities.

In Chapter 5, we present a new way to create embeddings for images from which recommendations can be made. Unlike traditional mappings (obtained through a pre-trained CNN), our system does not learn similar embeddigns for images with the same content, in our case two images will be similar if they have been taken in a restaurant visited by the same (or almost the same) group of users. After performing several experiments on various datasets and comparing with traditional systems and encodings (Collaborative Filters and pre-trained CNNs) the results support our hypothesis, a traditional encoding is not enough to take into account users' tastes, some method like our proposal is needed.

Based on the mapping learnt in the previous work, in Chapter 6 we present a system capable of summarizing the gastronomic offer of a city in a few images. Starting from all the images in the dataset, the system is able to create groups of users according to their tastes and to select, for each one, the images that best represent them. The results show that the system is able to create summaries from which the original dataset can be reconstructed to some extent, which is a traditional evaluation method in this type of problems.

Finally, in Chapter 7 we present a mobile application capable of generating restaurant recommendations based only on images of food. Thanks to what we have learned in the previous works, in this proposal we create a system equipped with several convolutional networks capable of detecting food in an image and if that is the case, identifying what kind of food it contains and finally recommending a restaurant.

After analyzing all the results obtained, it can be concluded that the main objective of the thesis has been achieved, namely, to take advantage of the unstructured information (text and images) that can be found in many Recommender Systems datasets. We have improved, not only the recommendation results, but we have also managed to add explainability and transparency. We have also created specific embeddings to map the input information in this type of problems to allow us to generate useful summaries for recommendation tasks. Our intention to continue exploring the possibilities that textual information can bring to the field of Recommender Systems. In this sense, we have several works in progress at the moment. In the future we will also explore ways of including text and images in a single system by combining all the knowledge gained from the works presented.

Conclusiones finales

Los Sistemas de Recomendación son de vital importancia en un mundo cada vez más conectado y donde la cantidad de información generada por cada usuario aumenta constantemente. La gran mayoría de plataformas que utilizamos día a día poseen algún tipo de recomendador que nos ayuda a distinguir entre el contenido irrelevante y el que de verdad tiene importancia. Gracias a ellos, los usuarios obtenemos una mayor satisfacción y las empresas mayores ingresos (entre otros beneficios).

Estos sistemas se entrenan, tradicionalmente, haciendo uso de la interacciones históricas de los usuarios (compras realizadas, canciones o vídeos reproducidos, noticias leídas, ...) puesto que no suele existir más información disponible para hacerlo. En la actualidad, muchas plataformas permiten también que los usuarios extiendan su opinión con texto o imágenes (información no estructurada), los cuales, creemos que pueden utilizarse a la hora de crear o mejorar un Sistema de Recomendación.

Tras realizar una introducción (Capítulo 1) a los términos básicos y problemas que deseamos resolver en esta tesis, en la Parte I del documento creamos y analizamos varios conjuntos de datos con información no estructurada en forma de texto e imágenes partiendo de reseñas públicas de la web TripAdvisor. Los diversos trabajos propuestos se detallan en las Partes II y III, siendo la primera la responsable de sacar provecho a la información textual y la segunda a la contenida en las imágenes.

El primer trabajo presentado (Capítulo 3) propone un Sistema de Recomendación a partir de reseñas textuales donde la explicabilidad y transparencia cobran especial protagonismo. Para lograrlo, se parte de una red neuronal simple de una sola capa oculta y sin función de activación que, una vez entrenado permite, no solo recomendar restaurantes con una tasa de acierto elevada, si no que también generar explicaciones para justificar al usuario final el origen de dichas recomendaciones. Los resultados demuestran que nuestra propuesta rinde casi igual que opciones más complejas que no poseen capacidades de explicación.

El segundo trabajo de esta tesis (Capítulo 4) presenta un sistema capaz de predecir, a partir de una foto, el usuario que la ha realizado. El objetivo que se persigue es poder utilizar este sistema para, cuando un usuario entra en la web de un restaurante, mostrarle solo aquellas fotos que, aunque no las ha realizado, podría haberlo hecho (poseen un estilo o contenido similares a las realmente tomadas por el usuario). Esta selección personalizada de imágenes hará que el usuario se convenza aún más de una recomendación dada. Los resultados muestran que nuestro sistema es capaz de identificar al autor de una fotografía con cierta precisión a pesar del gran abanico de posibilidades existente.

En el Capítulo 4 presentamos una nueva forma de crear embeddings para imágenes a partir de los cuales se pueden realizar recomendaciones. A diferencia de las codificaciones tradicionales (obtenidas mediante una CNN preentrenada), nuestro sistema no aprende embeddigns similares para imágenes con el mismo contenido, en nuestro caso dos imágenes serán similares si han sido tomadas en restaurantes similares (visitados por los mismos usuarios). Tras realizar diversos experimentos en varios conjuntos de datos y compararnos con sistemas y codificaciones tradicionales (Filtros Colaborativos y CNNs preentrenadas) los resultados apoyan nuestra hipótesis, una codificación tradicional no es suficiente para tener en cuenta los gustos de los usuarios y se necesita un método como el presentado.

Partiendo de la codificación aprendida en el trabajo anterior, en la Capítulo 6 presentamos un sistema capaz de resumir en pocas imágenes los gustos gastronómicos de una ciudad. Partiendo de todas las imágenes del conjunto de datos, el sistema es capaz de crear grupos de usuarios según sus gustos y de seleccionar, para cada uno, las imágenes que mejor los representen. Los resultados muestran que el sistema es capaz de crear resúmenes a partir de los cuales se puede reconstruir en mayor o menor medida el conjunto de datos original, lo cual es un método de evaluación tradicional en este tipo de problemas.

Finalmente, como último trabajo de la tesis, en la Capítulo 7 presentamos una aplicación móvil capaz de generar recomendaciones de restaurantes partiendo únicamente de una imagen de comida. Gracias a todo lo aprendido en los trabajos previos, en esta propuesta creamos un sistema dotado de varias redes convolucionales capaces de: detectar si hay comida en una imagen, en caso afirmativo identificar que tipo de comida contiene y finalmente recomendar un restaurante.

Tras analizar todos los resultados obtenidos, se puede concluir que se ha conseguido el objetivo principal de la tesis, sacar provecho de la información no estructurada (texto e imágenes) que acompaña a muchos datasets de Sistemas de Recomendación. No solo se han mejorado los resultados de recomendación, también hemos conseguido añadir explicabilidad y transparencia o hemos creado codificaciones específicas para este tipo de problemas que también nos permiten crear resúmenes. Nuestra intención es continuar explorando las posibilidades que la información textual puede aportar al campo de los Sistemas de Recomendación, es por esto que tenemos varios trabajos en desarrollo en la actualidad. En el futuro se explorarán formas de incluir texto e imágenes en un único sistema combinando todos los conocimientos extraídos de los trabajos presentados.

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, 2016.

AliExpress. Aliexpress. `http://play.google.com/store/apps/details?id=com.alibaba.aliexpresshd`. Access date: 15-06-2022.

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154, 2015.

Carlos Alonso, Pablo Pérez-Núñez, Oscar Luaces, Jorge Díez, Beatriz Remeseiro, and Antonio Bahamonde. Tripadvisor points of interest, December 2021. URL `https://doi.org/10.5281/zenodo.5749846`.

Fernando Amat, Ashok Chandrashekar, Tony Jebara, and Justin Basilico. Artwork Personalization at Netflix. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 487–488, 2018.

Amazon. Amazon compras. `http://play.google.com/store/apps/details?id=com.amazon.mShop.android.shopping`. Access date: 15-06-2022.

Ali A Amer, Hassan I Abdalla, and Loc Nguyen. Enhancing recommendation systems performance using highly-effective similarity measures. *Knowledge-Based Systems*, 217: 106842, 2021.

Gregory Amis. Improving TripAdvisor Photo Selection With Deep Learning. https://www.tripadvisor.com/engineering/improving-tripadvisor-photo-selection-deep-learning/, 2017.

Android. Android studio. `http://developer.android.com/studio`. Access date: 28-04-2022.

Sujoy Bag, Sri Krishna Kumar, and Manoj Kumar Tiwari. An efficient recommendation generation using relevant jaccard similarity. *Information Sciences*, 483:53–64, 2019.

Sachin Banker and Salil Khetani. Algorithm overdependence: How the use of algorithmic recommendation systems can increase risks to consumer well-being. *Journal of Public Policy & Marketing*, 38(4):500–515, 2019.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, Dennis Morello, and Fabiano Tarlao. Best dinner ever!!!: Automatic generation of restaurant reviews with lstm-rnn. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 721–724. IEEE, 2016.

Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 459–468, 2017.

Robin Burke, Michael D. Ekstrand, Nava Tintarev, and Julita Vassileva. Preface to the special issue on fair, accountable, and transparent recommender systems. *User Modeling and User-Adapted Interaction*, 31(3):371–375, 2021. doi: 10.1007/s11257-021-09297-5.

Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 352–356, 2018.

Kinjal Chaudhari and Ankit Thakkar. A comprehensive survey on travel recommender systems. *Archives of Computational Methods in Engineering*, 27:1545–1571, 2020.

François Chollet et al. Keras: The Python Deep Learning library, 2015. URL `https://keras.io/`.

Eric Chu and Peter Liu. Meansum: a neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232, 2019.

Wei-Ta Chu and Ya-Lun Tsai. A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web*, 20(6):1313–1331, 2017.

Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proceedings of the 23rd international conference on intelligent user interfaces companion*, pages 1–2, 2018.

Paul Covington, Jay Adams, and Emre Sargin. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM Press, 2016.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM Conference on Recommender systems*, pages 39–46, 2010.

Yashar Deldjoo, Markus Schedl, Paolo Cremonesi, and Gabriella Pasi. Recommender systems leveraging multimedia content. *ACM Computing Surveys*, 53(5):1–38, 2020.

Yashar Deldjoo, Tommaso Di Noia, Daniele Malitesta, and Felice Antonio Merra. A Study on the Relative Importance of Convolutional Neural Networks in Visually-Aware Recommender Systems. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3967, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Charles-Emmanuel Dias, Vincent Guigue, and Patrick Gallinari. Text-based collaborative filtering for cold-start soothing and recommendation enrichment. In *AISR2017*, 2017.

Vicente Dominguez, Pablo Messina, Ivania Donoso-Guzmán, and Denis Parra. The effect of explanations and algorithmic accuracy on visual recommender systems of artistic images. In *24th International Conference on Intelligent User Interfaces*, pages 408–416, 2019.

Olive Jean Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64, 1961.

Jorge Díez, David Martinez, Amparo Alonso-Betanzos, Oscar Luaces, and Antonio Bahamonde. Metrical representation of readers and articles in a digital newspaper. 09 2016.

Jorge Díez, Pablo Pérez-Núnez, Oscar Luaces, Beatriz Remeseiro, and Antonio Bahamonde. Towards explainable personalized recommendations by learning from users' photos. *Information Sciences*, 520:416–430, 2020a.

Jorge Díez, Pablo Pérez-Núñez, Oscar Luaces, Beatriz Remeseiro, and Antonio Bahamonde. Towards explainable personalized recommendations by learning from users' photos. *Information Sciences*, 520:416–430, 2020b. ISSN 0020-0255.

Bruce Ferwerda and Marko Tkalcic. Predicting users' personality from instagram pictures: Using visual and/or content features? In *26th Conference on User Modeling, Adaptation and Personalization*, pages 157–161, 2018.

Mickael Figueredo, Jose Ribeiro, Nelio Cacho, Antonio Thome, Andrea Cacho, Frederico Lopes, and Valeria Araujo. From photos to travel itinerary: A tourism recommender system for smart tourism destination. In *IEEE 4th International Conference on Big Data Computing Service and Applications*, pages 85–92, 2018.

Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

Achraf Gazdar and Lotfi Hidri. A new similarity measure for collaborative filtering based recommender systems. *Knowledge-Based Systems*, 188:105058, 2020.

Simona Giglio, Eleonora Pantano, Eleonora Bilotta, and TC Melewar. Branding luxury hotels: Evidence from the analysis of consumers' big visual data on TripAdvisor. *Journal of Business Research*, 119:495–501, 2020.

Stephanie Gil, Jesús Bobadilla, Fernando Ortega, and Bo Zhu. VisualRS: Java framework for visualization of recommender systems information. *Knowledge-Based Systems*, 155: 66–70, 2018.

Carlos A Gomez-Uribe and Neil Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 6(4):13, 2016.

Google. Google lens. `http://lens.google`. Access date: 15-06-2022.

Google I/O'22. Google keynote (google i/o '22). URL `https://www.youtube.com/watch?v=nP-nMZpLM1A&t=1191s`.

Guibing Guo, Yuan Meng, Yongfeng Zhang, Chunyan Han, and Yanjie Li. Visual semantic image recommendation. *IEEE Access*, 7:33424–33433, 2019.

Robert M Haralick, Karthikeyan Shanmugam, and Its' Hak Dinstein. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6): 610–621, 1973.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Ruining He and Julian McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *30th AAAI Conference on Artificial Intelligence*, pages 144–150, 2016.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, jan 2004. ISSN 10468188.

Antonio Hernando, Jesús Bobadilla, Fernando Ortega, and Abraham Gutiérrez. Trees for explaining recommendations made through collaborative filtering. *Information Sciences*, 239:1–17, 2013.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Yunzhe Jia, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie, and Xingjun Ma. Exploiting patterns to explain individual predictions. *Knowledge and Information Systems*, 62(3):927–950, 2020.

Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. Visually-aware fashion recommendation and design with generative image models. In *IEEE International Conference on Data Mining*, pages 207–216, 2017.

Khanabhorn Kawattikul. Product Recommendation using Image and Text Processing. In *International Conference on Information Technology*, pages 1–4, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, pages 1–15, 2014.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42:30–37, 2009.

Frank P Kuhl and Charles R Giardina. Elliptic fourier features of a closed contour. *Computer graphics and image processing*, 18(3):236–258, 1982.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. Leveraging Graph to Improve Abstractive Multi-Document Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243, 2020.

Shang Liu, Zhenzhong Chen, and Xiaolei Li. Time-semantic-aware poisson tensor factorization approach for scalable hotel recommendation. *Information Sciences*, 504: 422–434, 2019.

Wu Liu, Tao Mei, Yongdong Zhang, Cherry Che, and Jiebo Luo. Multi-task deep visual-semantic embedding for video thumbnail selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3707–3715, 2015.

Elena Lloret, Laura Plaza, and Ahmet Aker. The challenging task of summary evaluation: an overview. *Language Resources and Evaluation*, 52(1):101–148, 2018.

Abadi Martín, Agarwal Ashish, Barham Paul, Brevdo Eugene, Chen Zhifeng, Citro Craig, Greg S Corrado Andy Davis, Dean Jeffrey, Devin Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, pages 1–19, 2016.

Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough. In *CHI '06 extended abstracts on Human factors in computing systems - CHI EA '06*, pages 1097–1101, New York, New York, USA, 2006. ACM Press. ISBN 1595932984.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Don Monroe. Ai, explain yourself. *Communications of the ACM*, 61(11):11–13, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *27th International Conference on Machine Learning*, pages 807–814, 2010.

Loris Nanni, Stefano Ghidoni, and Sheryl Brahnam. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition*, 71:158–172, 2017.

Ani Nenkova and Kathleen McKeown. *Automatic summarization*. Now Publishers Inc, 2011.

Netflix. Selecting the best artwork for videos through A/B testing. Netflix Technology Blog: https://medium.com/netflix-techblog/selecting-the-best-artwork-for-videos-through-a-b-testing-f6155c4595f6, 2016.

James Neve and Ryan McConville. ImRec: Learning reciprocal preferences using images. In *Fourteenth ACM Conference on Recommender Systems*, pages 170–179, 2020.

Mehrbakhsh Nilashi, Othman Ibrahim, Elaheh Yadegaridehkordi, Sarminah Samad, Elnaz Akbari, and Azar Alizadeh. Travelers decision making using online review in social network sites: A case on TripAdvisor. *Journal of Computational Science*, 28:168–179, 2018.

Fernando Ortega, Antonio Hernando, Jesus Bobadilla, and Jeon Hyung Kang. Recommending items to group of users using matrix factorization based collaborative filtering. *Information Sciences*, 345:313–324, 2016.

Bhavik Pathak, Robert Garfinkel, Ram D Gopal, Rajkumar Venkatesan, and Fang Yin. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, 27(2):159–188, 2010.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Modeling and User-Adapted Interaction*, 22(4-5):317–355, 2012.

Pablo Pérez-Núñez, Oscar Luaces, Jorge Díez, Beatriz Remeseiro, and Antonio Bahamonde. Tripadvisor restaurant reviews, December 2021. URL `https://doi.org/10.5281/zenodo.5644892`.

Xueming Qian, Mingdi Li, Yayun Ren, and Shuhui Jiang. Social media based event summarization by user–text–image co-clustering. *Knowledge-Based Systems*, 164:107–121, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002.

Helmut Schmid. Part-of-speech tagging with neural networks, 1994.

Mete Sertkan, Julia Neidhardt, and Hannes Werthner. Eliciting touristic profiles: A user study on picture collections. In *28th ACM Conference on User Modeling, Adaptation and Personalization*, pages 230–238, 2020.

Yijun Shao, Stephanie Taylor, Nell Marshall, Craig Morioka, and Qing Zeng-Treitler. Clinical text classification with word embedding features vs. bag-of-words features. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2874–2878. IEEE, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications*, 37:100879, 2019.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 4278–4284, 2017.

Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. Adversarial training towards robust multimedia recommender system. *IEEE Transactions on Knowledge and Data Engineering*, (Early access):1–13, 2019.

TensorFlow. Tensorflow lite. `www.tensorflow.org/lite`. Access date: 28-04-2022.

Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pages 801–810. IEEE, 2007.

Nava Tintarev and Judith Masthoff. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):399–439, 2012.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide.* Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 3319579584, 9783319579580.

Chunlei Yang, Jialie Shen, Jinye Peng, and Jianping Fan. Image collection summarization via dictionary learning for sparse representation. *Pattern Recognition*, 46(3):948–961, 2013a.

Liu Yang, Steve Hanneke, and Jaime Carbonell. A theory of transfer learning with applications to active learning. *Machine learning*, 90(2):161–189, 2013b.

Longqi Yang, Yin Cui, Fan Zhang, John P Pollak, Serge Belongie, and Deborah Estrin. Plateclick: Bootstrapping food preferences through an adaptive visual interface. In *24th ACM International on Conference on Information and Knowledge Management*, pages 183–192, 2015.

Chenbin Zhang, Hongyu Zhang, and Jianqiang Wang. Personalized restaurant recommendation method combining group correlations and customer preferences. *Information Sciences*, 454:128–143, 2018.

Hong-yu Zhang, Pu Ji, Jian-qiang Wang, and Xiao-hong Chen. A novel decision support model for satisfactory restaurants utilizing social information: A case study of TripAdvisor.com. *Tourism Management*, 59:281–297, 2017.

Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*, 2018.

Rui Zhao and Kezhi Mao. Fuzzy bag-of-words model for document representation. *IEEE transactions on fuzzy systems*, 26(2):794–804, 2017.

Xiaolin Zheng, Menghan Wang, Chaochao Chen, Yan Wang, and Zhehao Cheng. Explore: Explainable item-tag co-recommendation. *Information Sciences*, 474:170–186, 2019.