Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Personalization in Recommender Systems through Explainable Machine Learning

Student: Jorge Paz Ruza
Advisors: María Amparo Alonso Betanzos
Berta Guijarro Berdiñas
Carlos Eiras Franco

A Coruña, June 2021.

To my family and friends.

Acknowledgements

First and foremost, I would like to thank all my family for unconditionally supporting me in all possible ways during my university degree studies. I also wish to express my gratitude to my advisors, Amparo, Berta and Carlos, for their collaboration during this project, as well as the research group LIDIA (Laboratorio de Investigación y Desarollo en Inteligencia Artificial) for offering me my first job. Lastly, I want to recognise the work of all teachers from the Facultade de Informatica de A Coruña, for their effort and dedication during these years.

Abstract

Recommender Systems have become ubiquitously utilized tools in multiple fields such as media streaming services, travelling and tourism business, e-commerce, and numerous others. However, in practice they show a tendency to be *black-box* systems, despite their increasing influence in people's daily lives. There is a lack of research on providing *personalised explanations* to the recommendations of a system, that is, integrating the idea of Explainable Artificial Intelligence into the field of Recommender Systems. Therefore, we do not seek to create a Recommender System, but instead devise a way to obtain this *explainability* or *personalisation* in such type of tool.

In this work, we propose a model able to provide said personalisation by generating explanations based on user-created content, namely text or photographs. In the context of the restaurant review platform *TripAdvisor*, we will predict, for any (user,restaurant) pair or existing recommendation, the text or image of the restaurant that is most adequate to present said recommendation to the user, that is, the one that best reflects their personal preferences. This model exploits the usage of Matrix Factorisation techniques combined with the feature-rich embeddings of pre-trained image classification and language models (Inception-ResNet-v2 and BERT), to develop a method capable of providing transparency to Recommender Systems.

Resumen

Los Sistemas de Recomendación se han convertido en herramientas usadas extensivamente en multitud de campos como *online streaming*, turismo, restauración, viajes y comercio electrónico, así como muchos otros. Sin embargo, en la práctica presentan una tendencia a ser sistemas de *caja negra*, pese a la cada vez mayor influencia que presentan sobre el día a día de nuestra sociedad. Hay una falta de investigación sobre la idea de aportar *explicaciones personalizadas* a las recomendaciones de un sistema, es decir, integrar el concepto de Inteligencia Artifical Explicable en el área de los Sistemas de Recomendación. Por lo tanto, no buscamos crear un Sistema de Recomendación *per se*, sino idear un modo de obtener esta capacidad de *explicabilidad* o *personalización* en dicho tipo de sistemas.

En este trabajo, proponemos un modelo capaz de proveer de esta personalización mediante la generación de explicaciones basadas en contenido generado por los usuarios, en particular texto e imágenes. En el contexto de la plataforma de reseñas de restaurantes *TripAdvisor*, buscaremos predecir, para cualquier par o posible recomendación (usuario, restaurante), la imagen o texto sobre dicho restaurante más adecuada para presentar esa recomendación al usuario, es decir, la imagen o texto que mejor refleja las preferencias personales del usuario. Este modelo explota el uso de técnicas de Factorización Matricial combinadas con modelos

de lenguaje y clasificación de imágenes (BERT e Inception-ResNet-v2), para desarrollar un método con capacidad de otorgar transparencia a Sistemas de Recomendación.

Keywords:

- Explainable Artificial Intelligence
- Text and Image-based personalised recommendation
- Matrix Factorisation
- Natural Language Processing
- Recommender Systems
- Image Classification
- BERT
- Inception-ResNet-v2
- Supervised Learning
- Dyadic data
- TripAdvisor

Palabras clave:

- Inteligencia Artificial Explicable
- Recomendación personalizada sobre texto e imagen
- Factorización Matricial
- Procesamiento de lenguaje natural
- Sistemas de Recomendación
- Clasificación de imágenes
- BERT
- Inception-ResNet-v2
- Aprendizaje Supervisado
- Datos diádicos
- TripAdvisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recommender systems (RS) are widespread information filtering tools used in all kinds of fields and applications, such as entertainment, multimedia, e-commerce, and multiple service business. Using data gathered from the users that interact with the platform, these systems are able to predict their preferences and offer certain products or services in the most optimal way, trying to benefit both the users and the business.

Recommender Systems provide a high degree of automation and performance in improving the user or customer satisfaction, which effectively makes them a powerful tool in the enterprise context. Logically, these benefits are directly proportional to the amount of user data available, making them highly relevant in the blooming field of Big Data.

Despite the flourishing of RS in all sorts of information filtering contexts, they still show a strong tendency to be *black-box* systems, as they may not be transparent when making recommendations. In this project, we will not focus on recommending items ourselves, but instead explore the idea of *Explainability* in RS, discussing on the benefits of *personalising the recommendations* based on each user's peculiarities. Specifically, the main focus of our research will be to assess the viability of using already existing user-uploaded content (text and images) to generate tailored, personalised explanations to other users.

1.1 Recommender Systems and Content Personalization

Recommender Systems have obtained very clear success as a tool that makes the difference when engaging with users and generating business value. The organizations that make the most profit from them, mainly *e-commerce* and digital media companies, often state so in their official dev blogs. Netflix, for instance, revealed in one of their periodic blog posts that *"(...) now 75% of what people watch is from some sort of recommendation. (...)"* [8], as early as in 2012. With their recommendation system as a keystone of the company, Netflix has been consistently increasing their business value since the very start of the bloom of online media

streaming services [9]. Another clear example is Spotify, a music streaming platform that obtains billions of weekly streams through their user-personalized *Discover Weekly* playlists [10], and actively recommends new artists to users in a personalised way by denoting they are similar to a different artist the user listens to, as it can be seen in Figure 1.1.

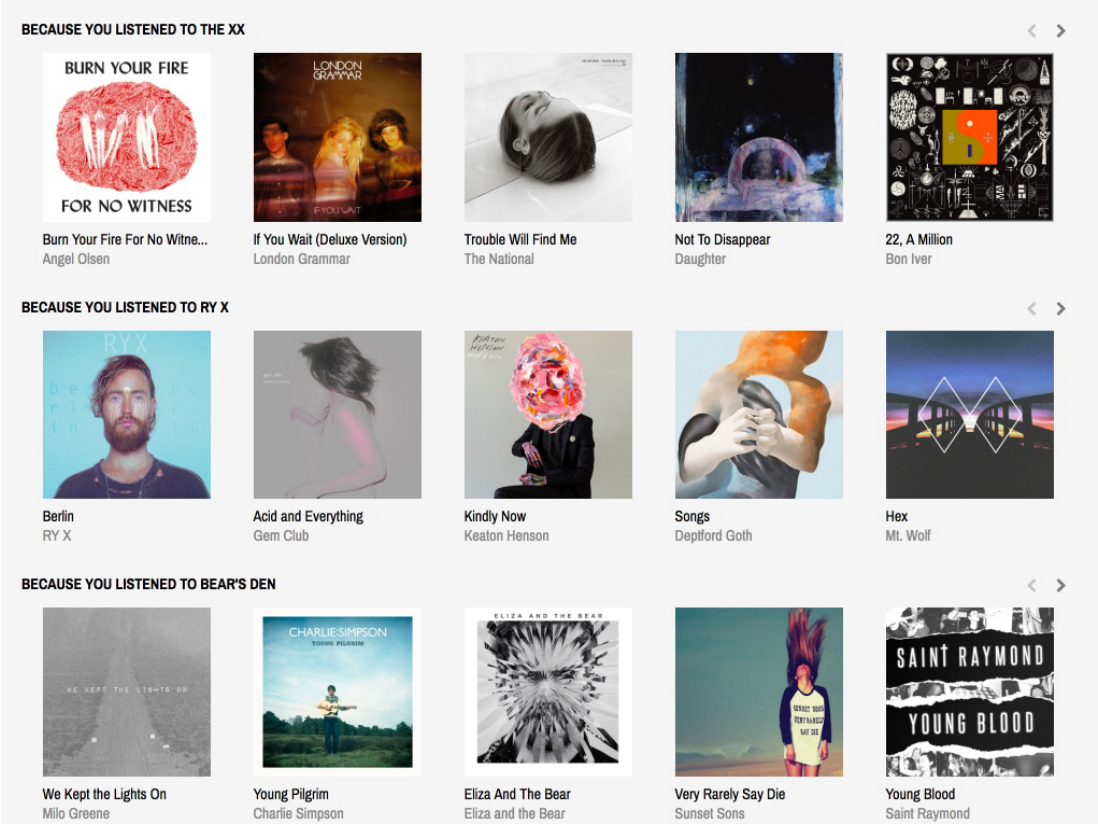

Figure 1.1: Spotify embeds personalisation into their *Artist Discovery* system: when recommending a new artist to any user, they will use as an explanation their similarity to an artist that the specific user usually listens to.

As a general rule, recommendations have gained large importance in most of the major online platforms, where a relatively high amount of the presented content is personalized considering each user's preferences.

In an era where the incoming amount of information to the user becomes at times overwhelming, content personalization begins to matter significantly to them. According to the consulting company Accenture, *"almost 70% of consumers want companies to personalize their communications"* [11]. This statement is reinforced with a recent study from personalization software company Monetate and WBR Research, which has found that 93% of businesses with advanced personalization strategies increased their revenue in 2018 [12].

While Recommender Systems have an established and proved success in their areas of

application, there is still not enough research in their ability to be integrated in Explainable Artificial Intelligence (XAI) models, that is, those in which the results of the solution can be understood by humans. XAI has become a concept of increasing in relevance during the past decade, and has become even more critical recently, as the General Data Protection Regulation (GDPR) of the European Union demands transparency in any data-treating system that takes decisions which affect people, which is the case of Recommender Systems: since they provide direct recommendations to users, explainability becomes an important part of the system.

Therefore, it would be ideal to have at our disposal a way to provide a sufficient explanation of any recommendation *(user,item)* provided by the system. Embedding *explainability* in our system will enable users to understand the recommendations they receive (it would answer to the question of *"Why am I being recommended this item"?*). Furthermore, since each user is different and has distinctive preferences, the explanation of an item is bound to not always be the same for every user: integrating explainability in Recommender Systems can be considered, by all means, a way of adding personalisation to it. This way, provided that we already are aware of ***what to recommend***, we want to incorporate the properties of XAI into our system, by researching a method to know how to present that recommendation. This is, we aim at knowing ***how to recommend***.

Being able to achieve this not only makes the decision-making of our AI model more *transparent*, but also highly benefits the performance of Recommender Systems as a whole: it enables the user to better understand any given recommendation, increasing their trust in the system, which means potentially enhancing its business value as well as the overall user experience.

Our proposal will explore the idea of obtaining personalised explanations through the usage of existing media content like images and text. In the context of the chosen platform, TripAdvisor, we will explore these possibilities separately, searching for the written reviews or images that best complement the recommendation of a restaurant to given a user. Thus, we will provide the review or image that better represents the chosen restaurants in terms of the user's preferences.

1.2 Related Work

In the context of Recommender Systems, the state-of-the-art shows a large amount of varied proposals in the basic problem of pairing users with new items, through various models of information filtering: collaborative, content-based, demographic or hybrid filtering approaches; all of them share the main idea of identifying features that are relevant to both users and items, and the optimal model for each problem heavily depends on the available data. Nevertheless, Recommender Systems are a helpful alternative to other information filtering algorithms,

which makes them great candidates to study content personalization on.

1.2.1 A review on different types of Recommender Systems

- **Collaborative filtering** focuses on making predictions about the reactions from a user to an item by taking into account the preferences from many users, which figuratively *collaborate* in the prediction. This model mimics user-to-user recommendations, as all it takes into account is other users' preferences, and completely neglects the explicit characteristics of the items or the users. Collaborative filtering provides good results regarding diversity (detecting how dissimilar recommended items are), serendipity (a measure of how surprising the successful or relevant recommendations are), and novelty (how unknown recommended items are to a user). However, the computational cost is high, and it greatly suffers from the so-called *cold start problem*, as it has trouble recommending new items without a large amount of interaction data to train a model. Notwithstanding, this type of Recommender System has been used extensively; the pioneer of this type of system was the e-commerce giant Amazon, that successfully exploited a *item-to-item* collaborative filtering system to recommend products to customers [13].

- **Content-based filtering** relies on known user preferences provided explicitly or implicitly, and data about item features. As it can be seen in Figure 1.2, in this case other users do not "collaborate" in the recommendation, but instead the recommendation procedure is based on looking for items that are similar to the ones the user has liked before. They do not require large amounts of interaction data like collaborative filtering systems, but rely heavily on the feedback of the user for their interactions with each item. It is a simple implementation of Recommender Systems, but tends to create "static" recommendations, as well as also suffering from the aforementioned *cold start problem*, as it cannot recommend items to a user when little to no information is available about their preferences.

- **Demographic filtering** places an emphasis in the demographic information of the users, which should allow to distinguish among different "communities" of users, through the usage of clustering algorithms.

- **Hybrid filtering** presents a combined approach of all the previously commented alternatives to integrate and exploit their advantages as much as possible.

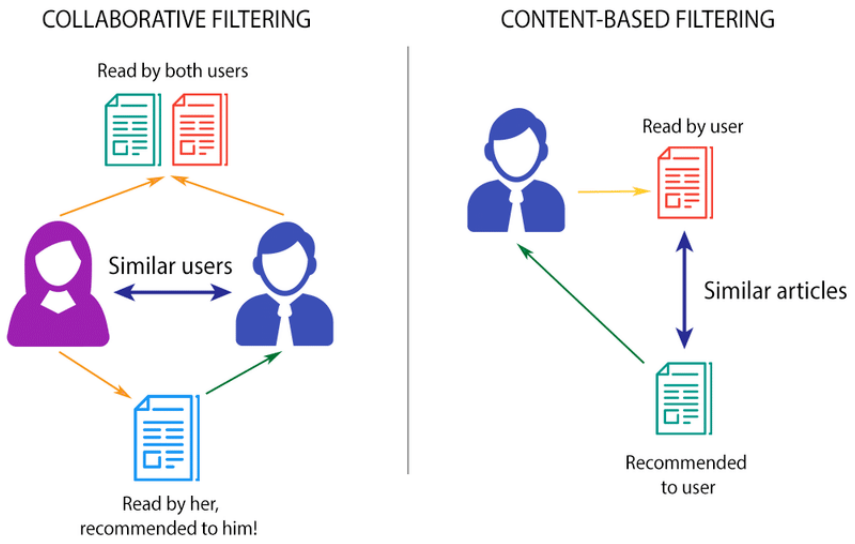

Figure 1.2: Visual comparison between the two most popular Recommender System approaches: while Collaborative Filtering takes advantage on users sharing common preferences, Content-based Filtering relies on extensive knowledge about item similarity: it provides recommendations to a user based only on their preferences [1].

1.2.2 Enhancing (user,item) recommendation pairs through user-based Personalisation

As it was previously commented, the main objective of our research is not recommending items given the context of a set of users (collaborative filtering) or a set of features (content-based filtering), but rather **personalising the way we present a recommendation of an item to a given user, using visual content such as images and text**. Since this is an additional layer we want to add to enhance the user satisfaction when using the system, **any recommender system can be employed in the underlying layer of our task**, in the context of content recommendation and personalisation. As far as we can know, this idea of content-driven personalisation of items has been already explored in one particular context, which is the usage of personalised thumbnails in the media streaming platform *Netflix*.

As Netflix heavily depends on visual content to attract potential subscribers, maximizing personalisation is key to their business model. After repeatedly hypothesising about it, Netflix conducted a study that revealed that *"(...) artwork was not only the biggest influencer for a user's decision about what to watch, it also constituted over 82 percent of their focus while browsing Netflix."* [14]. This meant that a compelling thumbnail could make a great difference when looking for engagement towards the titles available in the platform. To achieve this, each movie should ideally have a personalized thumbnail that maximizes the *click-through rate*, in hopes that this will relate to a higher monthly subscriber count and increased revenues. To

achieve this, Netflix tries to present to each user thumbnails that have worked better with users with similar preferences, and hopefully maximize the chances of drawing the attention of the user.

Examples of Netflix's thumbnail personalisation algorithm are shown in Figures 1.3, 1.4 and 1.5: the same Netflix series may not always be recommended using the same thumbnail for all users, and even for the same user. Depending on different user's overall preferences (such as in Figure 1.3) or a sole user's recent watching activity (like in Figure 1.4), the used thumbnail for a series or film may vary between users or during the time. However, it is of extreme importance to maintain this personalisation within the realm of the real features of an item: as it can be seen in Figure 1.5, Netflix has in some cases recommended some of their contents using thumbnails with minimal relevance to the actual movie, due to an excessive tailoring to suit user tastes or demographic traits, such as ethnicity.

Nevertheless, there is one crucial difference between Netflix's system and our proposed approach to achieving personalisation in the *TripAdvisor* platform: while Netflix uses thumbnails carefully crafted by expert graphic designers after automatically selecting the most *aesthetic* frames for the matter [2], we will rely in the content provided by the users themselves, which adds an additional layer of context and difficulty to take into account.

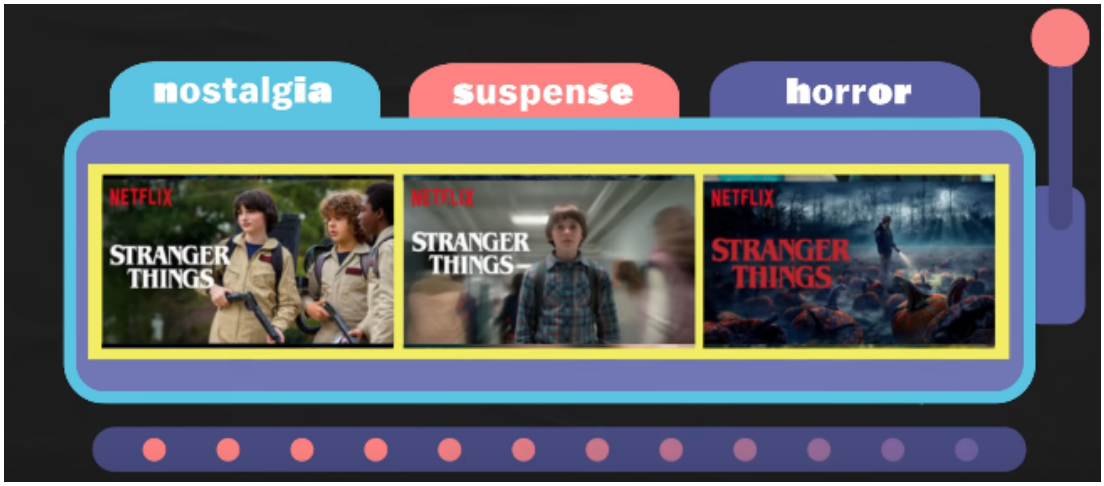

Figure 1.3: For different users, the same series (*Strange Things*) is recommended. However, the used thumbnails can be noticeably different: users that have different preferences regarding genres may be presented personalized images which depict those genres in the context of a certain TV series [2].

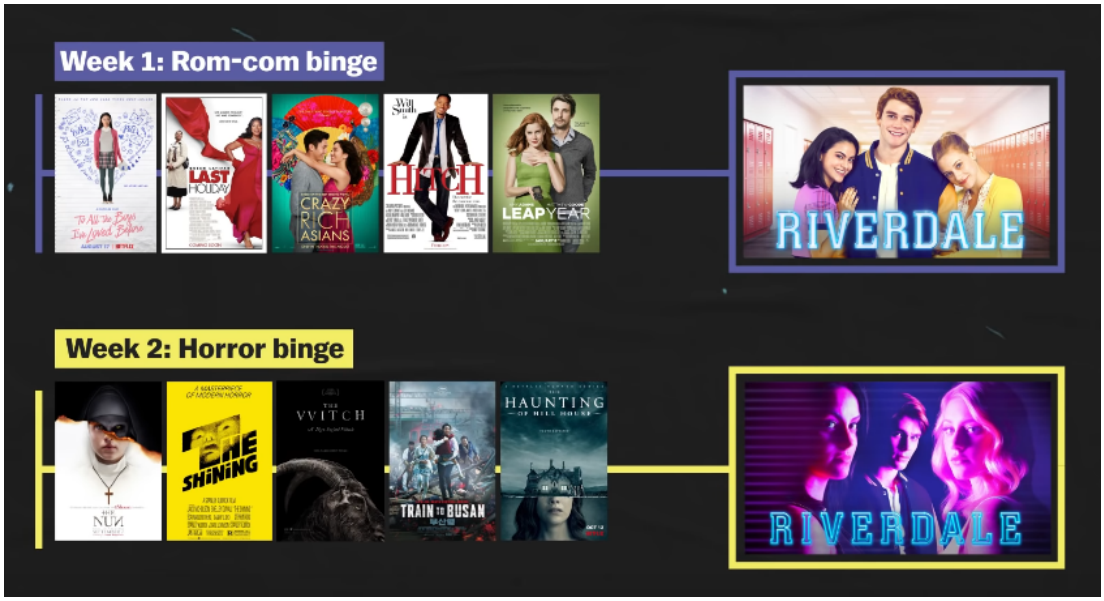

Figure 1.4: Even for the same user, a change in the tendencies of the type of content they watch (romance vs. horror) can suppose a change in the thumbnail used to recommend the same TV Series to that user. [2].

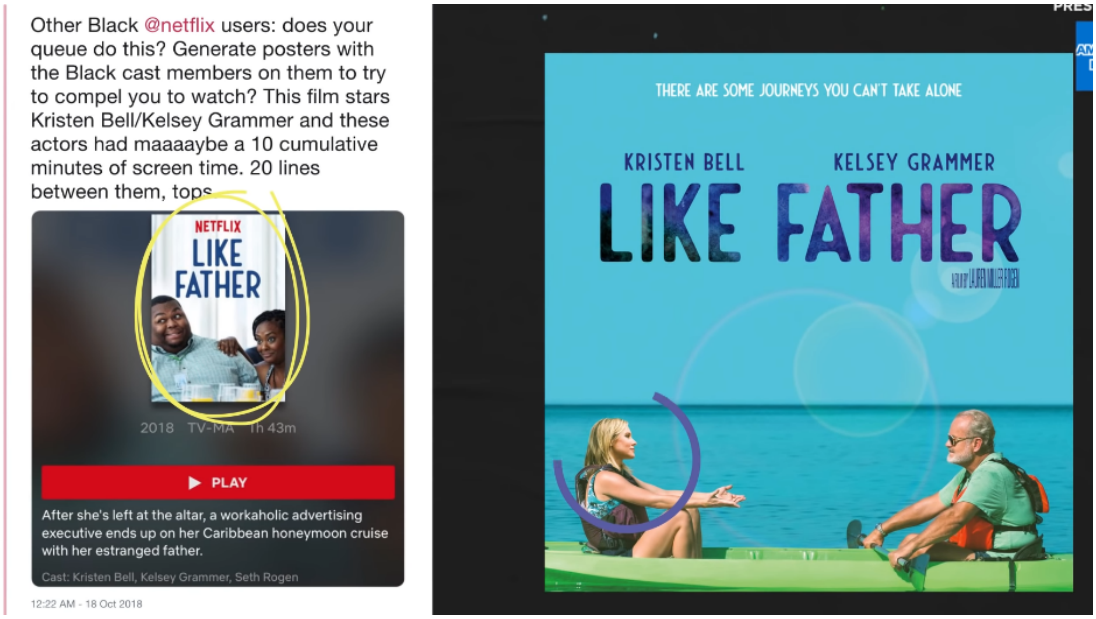

Figure 1.5: Even when trying to maximize content personalization, it's still crucial to stay relevant to the actual features of an item. For a lot users, Netflix presented a personalised thumbnail for the film *Like Father* based on their ethnicity, despite the characters appearing on said thumbnail not being relevant in the movie [2].

1.3 Main Goal of the Proposal

Given a certain city, its set of users, restaurants, and all of the interactions (reviews) between them, within the context of the *TripAdvisor* platform, our aim is to obtain personalised recommendations using both image and text-based approaches. For any recommendation pair *(user,item)*, we will incorporate into the recommendation the image/text review that best represents the users' preferences. Alternatively, this can be viewed as an attempt to recreate the review that the user would create if they had visited the recommended restaurant previously. We do not want to auto-generate images or texts nor recommend restaurants ourselves, but instead explore personalization as a mean to obtain explainability in the system's recommendation. Our line of work will also focus on the viability of Matrix Factorization (MF) techniques to obtain the desired explainability.

Chapter 2
Materials and methods

This chapter provides an insight over the gathered datasets, and describes the methods required to take on our proposal.

2.1 TripAdvisor datasets

In this section, we will describe the data available on both text and images datasets, analysing how the observed distributions of certain data features may affect the learning process.

2.1.1 Text Datasets

With the objective of training our model for personalisation through textual content, we collected large amounts of reviews from the *TripAdvisor* platform. We selected various cities with a focus on finding English-written reviews since BERT, the selected NLP model which we will discuss in Section 2.2.1, is trained exclusively over English texts. With this in mind, the selected cities were:

- A Coruña (Spain)
- Gijón (Spain)
- Barcelona (Spain)
- Madrid (Spain)
- Paris (France)
- London (United Kingdom)
- New York City (United States)
- Delhi (India)

The first two cities, *A Coruña* and *Gijón*, are small datasets selected with the objective of testing the ability of the model to train with few data, whereas the rest are large metropolises with huge amounts of reviews. While an extensive amount of information is known about each review, only the attached text is fed to the system, so most of the available information of each review is not present in the final, pre-processed datasets. [1]

Field Name	Field type	Description
parse_count	Integer	Auto-incremental index in order of extraction
user_id	String	Unique id of the *TripAdvisor* user
author	String	In-app, non-unique nickname of the *TripAdvisor* user
restaurant_name	String	Name of the restaurant the review corresponds to
rating_review	Integer	Score (no. of stars) given to the restaurant in the review [1-5]
sample	String	Whether a review is "Positive" [4-5] or "Negative" [1-3]
review_id	String	Unique id of the review
title_review	Text	Review title
review_preview	Text	Summarised review
review_full	Text	Complete review
url_review	Text	Url of review in the *TripAdvisor* platorm
date	Date	Date of the review submission (DD, MM, YYYY)
city	String	City where the restaurant of the review is located at
url_restaurant	Text	URL of the Restaurant in the *TripAdvisor* platorm

Table 2.1: Detail of the available information for each review (fields, field types and field descriptions)

Dataset	No. of Reviews (All languages)	Size	No. of Reviews (English)	Positive reviews (English)	Negative reviews (English)	Unique users	Unique restaurants
Delhi	148,541	162.5 MB	148,303	123,095	25,208	59,796	5,147
New York	517,904	559.1 MB	517,604	424,823	92,781	218,738	1,715
Madrid	216,565	198.9 MB	177,353	145,177	32,176	88,560	5,481
London	1,000,365	1.1 GB	998,939	833,453	165,485	441,821	1,827
Gijón	11,797	8.9 MB	1,537	1,246	291	902	294
A Coruña	10,089	8.4 MB	2,517	2,155	362	1563	397
Barcelona	426,785	428.3 MB	417,240	339,385	77,855	203,514	6,319
Paris	527,663	558.8 MB	510,084	401,589	108,495	219,340	11,004

Table 2.2: General information of each city's dataset regarding review language, positive/negative reviews and number of unique users and restaurants. The fields *Unique users* and *Unique restaurants* fields are computed only considering positive reviews written in English.

Table 2.1 provides a description of the existing information for each of the reviews in the datasets. On the other hand, Table 2.2 provides information for each of the cities' datasets, showing raw review count and storage space usage, as well as the resulting size after applying

[1] The raw review data for all cities is available at: https://udcgal-my.sharepoint.com/:f:/g/personal/inigo_lopezrioboo_botana_udc_es/EpS_7Dd9g8JGsyOyI-m-0UEB8sT3Oz03Q6fba2Zk4SF4NQ?e=cWphsj

each of the used filters: only positive reviews written in English. We also detail the final number of reviews of each dataset, along with the amount of unique users and restaurants, all after applying the aforementioned language and review positiveness filters (since we will only train and evaluate our model with texts that meet those conditions). By looking at the obtained general information of all separate datasets, we can make some insightful inferences about how users interact with the *TripAdvisor* platform:

- Considering exclusively English-written positive reviews, there is evidence that the *TripAdvisor* platform has a large amount of inactive users. As we can observe in Table 2.2, throughout all individual cities' datasets there are roughly *twice as many reviews as unique users*. We can say that, on average, we can find a new user in the platform for each pair of reviews. This distribution of user uniqueness is insightful: we are now aware that there will be a sizeable part of the *TripAdvisor* userbase which we have really little information about. Logically, since the quality of learning of the model relies on the amount of information available for each user, we will need to address this issue, which we can identify as a clear example of the *cold start problem* in the context of Recommender Systems. Figure 2.1 shows the amount of users that have written n reviews in the platform: as it can be observed, the majority of the data is concentrated within the first bins of the histogram, which corresponds to users with low review counts. That is, most of the users are rather *inactive* in the *TripAdvisor* platform. This tendency also seems to be a constant trend for all the selected cities, so we expect the sparsity of user interaction data so be problematic in all of our training contexts.
- On cities which are neither English-speaking nor international/touristic (that is, A Coruña and Gijón), filtering the data to include only reviews written in English greatly hinders the availability of training and testing samples. On larger, touristic and/or English-speaking cities, this filtering is not a problem.

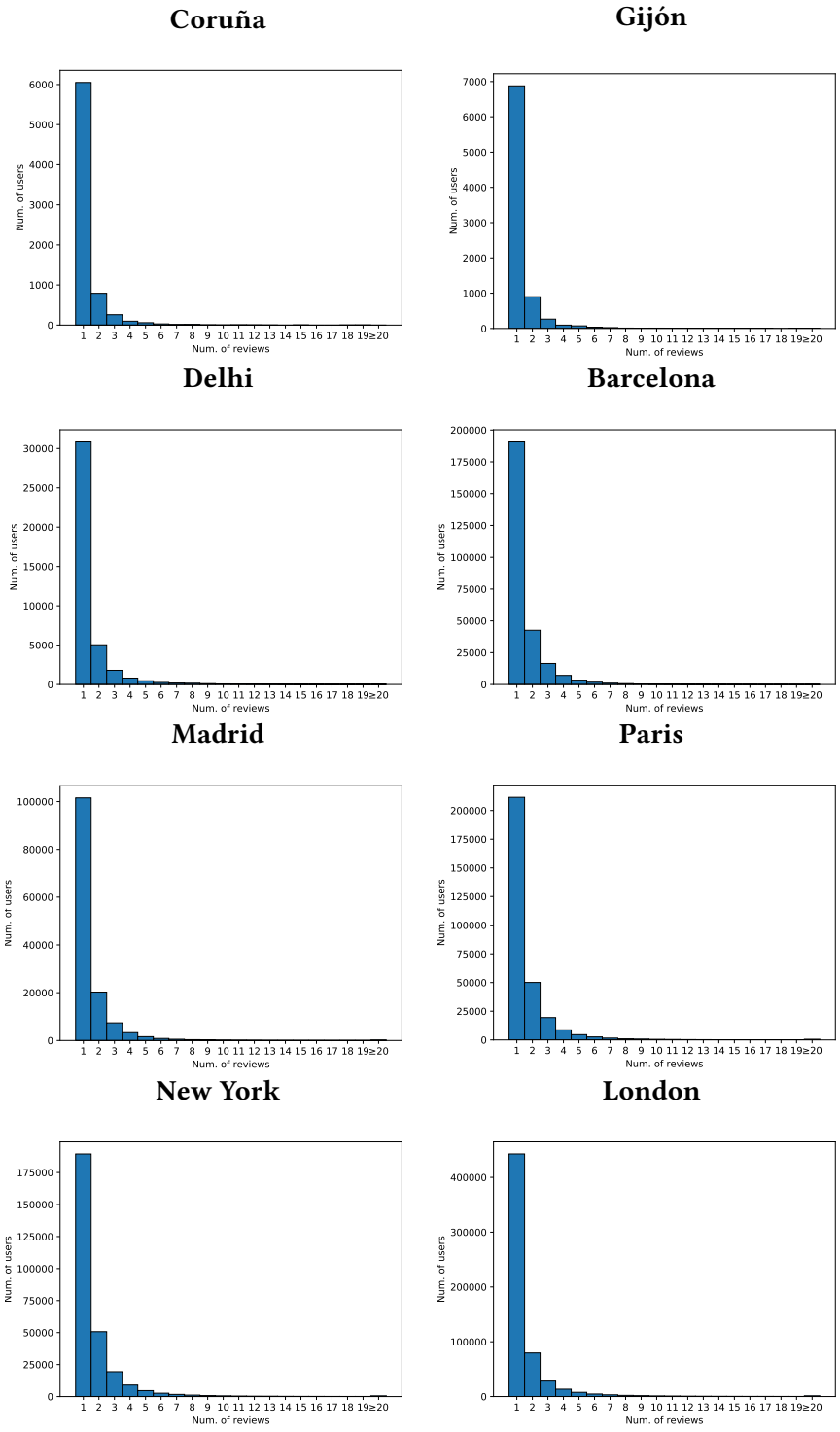

Figure 2.1: Distribution showing the number of reviews per user in all Text datasets.

2.1.2 Image Datasets

To provide personalisation by means of image recommendation, we will use the images attached by users to their reviews of a restaurant. The selected data had been previously gathered by a different research team from the *University of Oviedo*[2], and used for their own approach to personalisation in Recommender Systems [15].[3]

The aforementioned dataset includes image review information for the following cities:

- Gijón (Spain)
- Barcelona (Spain)
- Madrid (Spain)
- Paris (France)
- London (United Kingdom)
- New York City (United States)

As it was discussed in Section 2.1.1 when we examined the gathered written text data, the *Gijón* dataset is meant to be a toy-sized dataset to test the learning pipeline and the performance of the model when facing low amounts of training data. It is also worth mentioning that these datasets only contain data corresponding to *positive* reviews, as we can assume that the images attached to negative reviews are likely to be counterproductive if we use them for explaining a recommendation of that restaurant to a user.

Dataset	No. of Images	Size	Unique users	Unique restaurants
Gijón	18,679	110.1 MB	5,139	598
Barcelona	150,416	1.3 GB	33,537	5,881
Madrid	203,905	1.9 GB	43,628	6,810
New York City	231,141	3.6 GB	61,019	11,982
Paris	251.636	3.6 GB	61,391	11,982
London	479.798	7.3 GB	134,816	13,888

Table 2.3: General information of each city's dataset regarding its raw size and number of photos, unique users and restaurants.

[2] This TFG is a work of research conducted within the context of the project "xLearn: Aprendizaje Automático Escalable y Explicable" in coordination with a group from the *University of Oviedo*

[3] The pre-processed dataset is available for all the selected cities at https://www.kaggle.com/chusano/tripadvisor-image-restaurant/version/1

Table 2.2 provides information for each of the cities' datasets. A quick analysis of this data reveals some interesting properties about it. While text datasets had a *two-to-one* ratio regarding number of existing written reviews and unique users, images tend to have a *four-to-one* ratio. To be able to explain why this happens, we need to delve into the in-depth distribution of the data in these image datasets. On the one hand, Figure 2.2 shows that, while a majority of reviews include only one image, when choosing to attach more than one photograph a lot of users will decide to upload four or more of them. On the other hand, in Figure 2.3, which includes the distribution of reviews per user (a single review may include multiple photos), we can observe that the userbase inactivity levels are similar to those seen in the text datasets. What we can infer from this is that, even for inactive users, we will have on average more data to learn from. This supposition is consolidated if we analyse the distribution of available individual images per user, which can be seen in Figure 2.4: while in the case of text review data there was a majority of users who had written only one review, in the context of images this scarcity of data is lower; for most of the users we will have multiple images to learn from, which will likely make a difference when it comes to training our model.

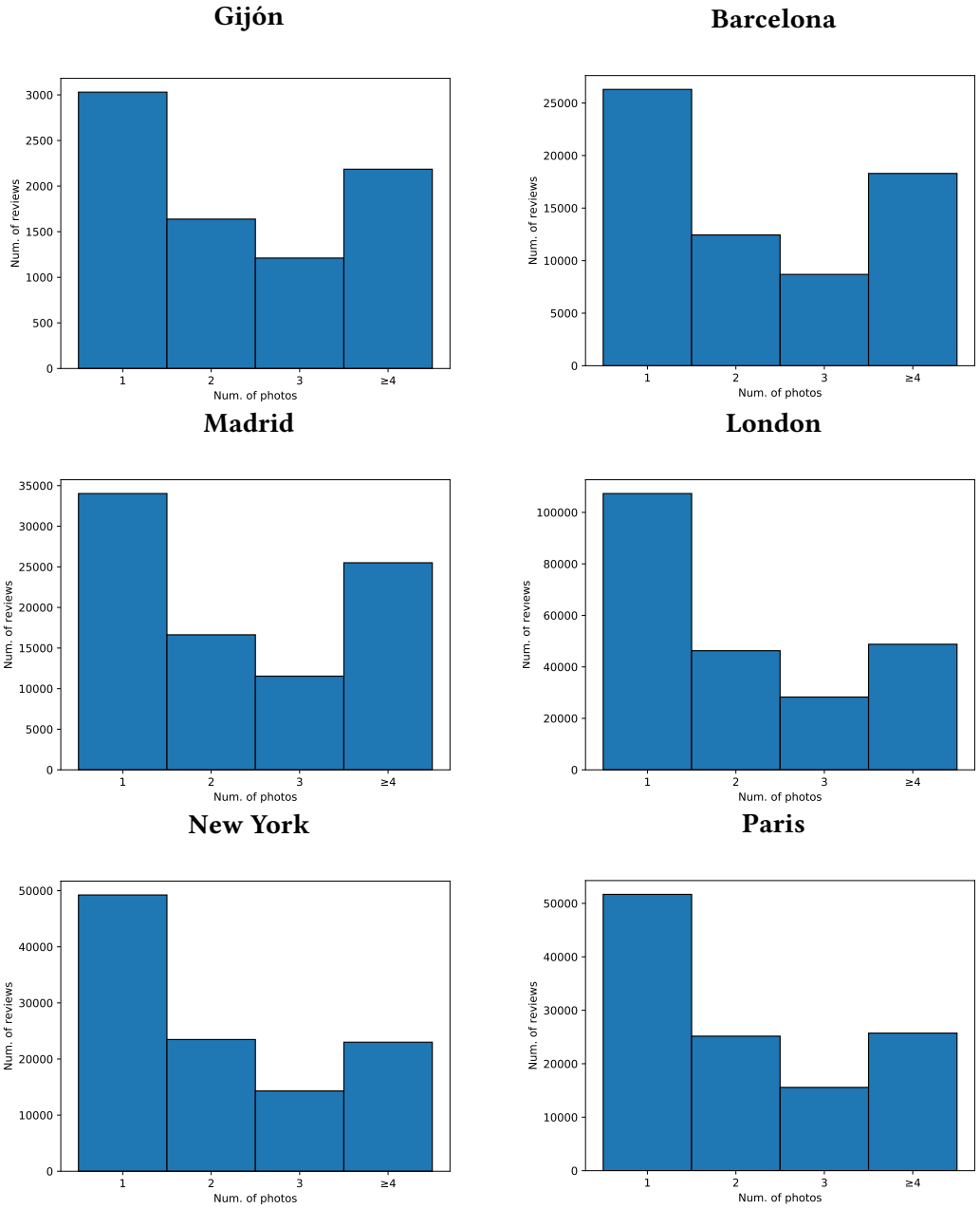

Figure 2.2: Distribution showing the no. of photos included per review in the processed Image datasets for all of the selected cities.

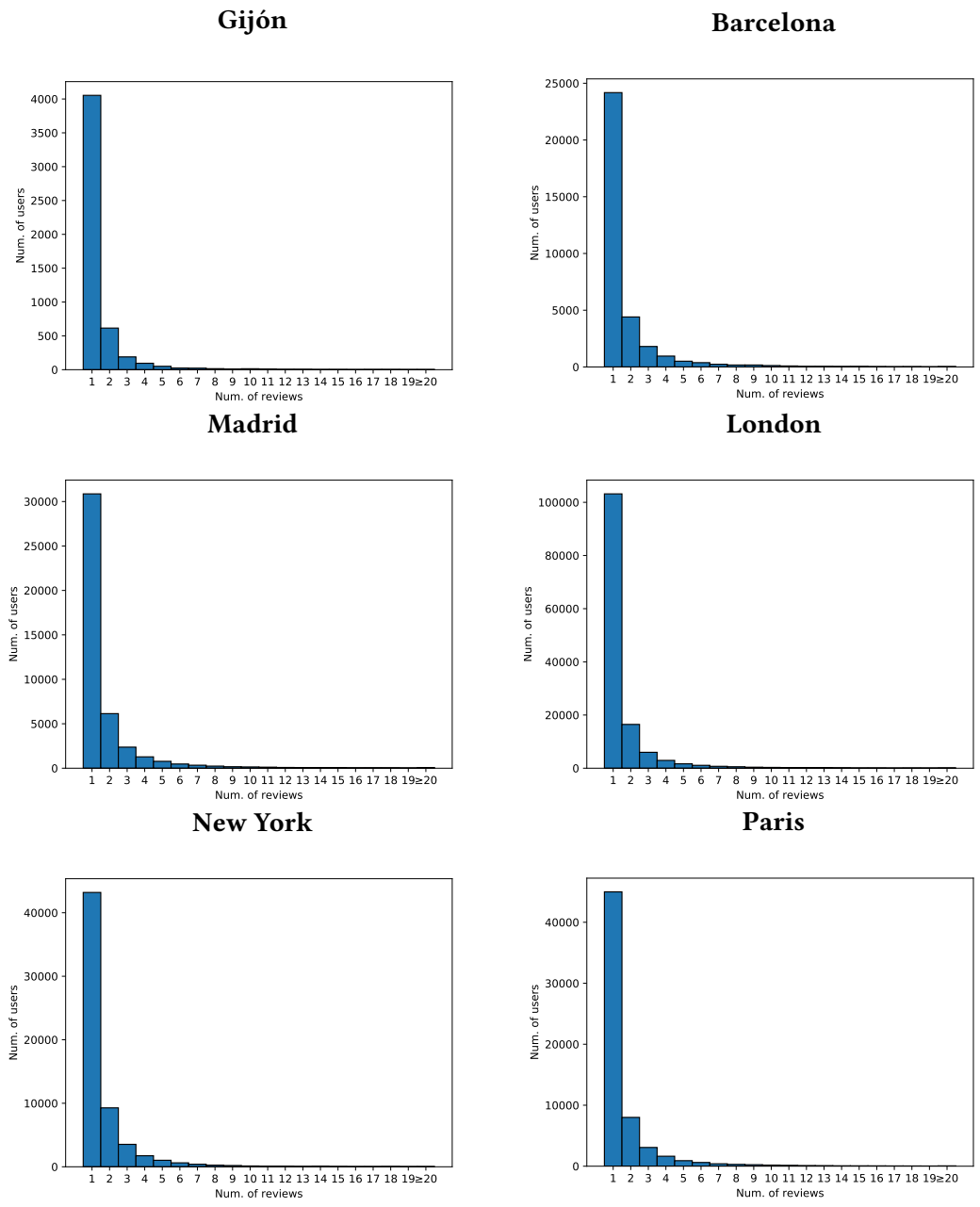

Figure 2.3: Distribution showing the no. of reviews per user in each processed images dataset.

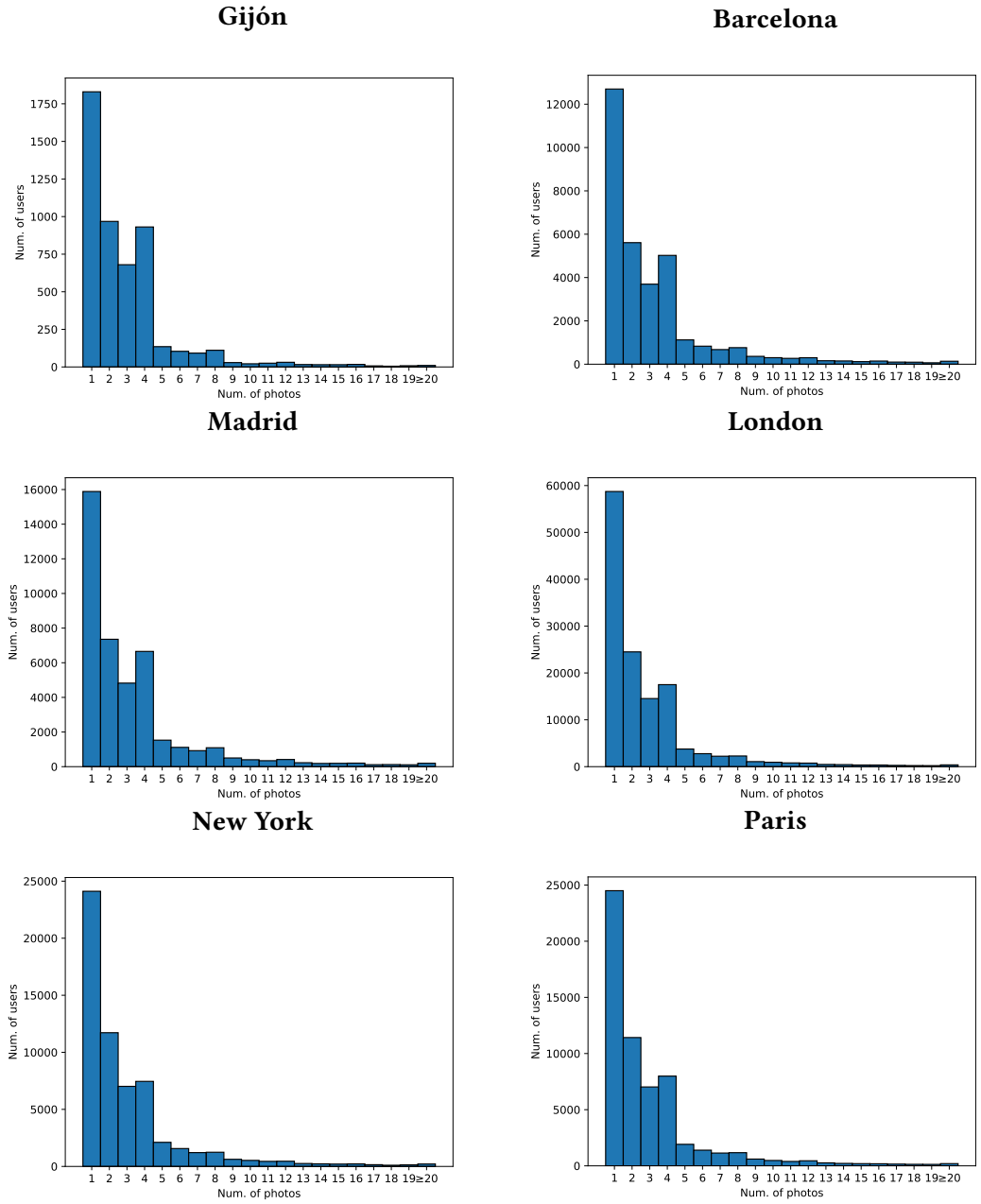

Figure 2.4: Distribution showing the no. of individual *photos* per user in each processed image dataset from the selected cities.

2.2 Methods

Since our intention is to work with the text and images attached to the reviews in order to use them as inputs to our proposed Machine Learning model, we're required to map or *embed* them to a numerical representation from a natural language or visual context, respectively.

In this section, we will discuss the basic aspects of Natural Language Processing and Image Classification that need to be addressed as to be able to create vector representations of the user's content.

2.2.1 Natural Language Processing for Document Embedding

We seek to create real-valued vector representations of each of the written reviews in our Text Datasets. This "translation" or encoding has the following objective: words, phrases or documents that are close to each other in their vector representation are expected to have similar meanings, styles or interpretations in the context of natural language.

Typically, a text consisting in multiple words will be divided into *tokens* (which may or may not correspond to the individual words) and the embedding model will map them to a vector of real values of a given, arbitrary dimension. This is, the basic functioning of Natural Language Processing (NLP) models consists in providing a complete embedding for each of the tokens. However, it is common to obtain instead a *pooled* representation of the text, which corresponds to generating a single final vector that summarises the meaning of the whole text, as depicted in Figure 2.5. This is advantageous as it allows us to work with representations of relatively low dimensionality, specially when compared to the dimensionality of sparse word-space representation. In that case, we would require a vector of size $(1 \times vocabulary\ size)$; conversely, text embeddings can be arbitrarily small.

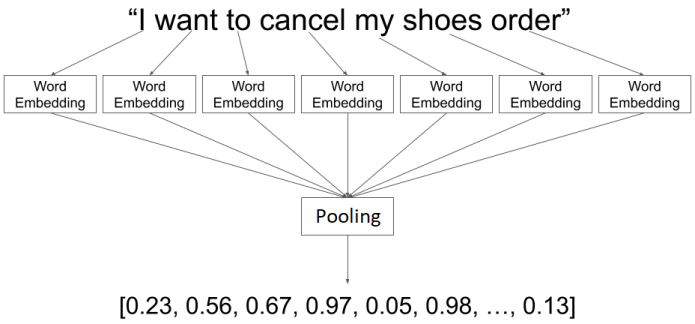

Figure 2.5: Example of a complete sentence or document embedding. After computing the individual word embeddings, these are *pooled* to generate a vector that summarises the meaning of the whole text [3].

2.2.1.1 Document pre-processing

Before computing the embeddings, several previous steps are usually considered in order to *pre-process* the text, in an attempt to enhance the performance of the embedding model and thus our Machine Learning proposal [16].

As a general statement, we should ensure the final pre-processed text maintains the original essence and meaning while trimming away all the irrelevant elements that may difficult the actual processing. Some of the tools to achieve this include:

- **Lowercasing:** Simply lowercase all of the text. Despite looking simple or naive, it is one of the most effective forms of text pre-processing. In the context of *TripAdvisor* reviews, we can expect this to be a useful step of pre-processing: users are likely to have incorrect and inconsistent capitalisation (some customers may even write their whole review in capitalised letters), so lowercasing will allows to by-pass that problem. Nevertheless, it is logical that this method will be counterproductive in other contexts where capitalisation may be crucial regarding word meaning or sentence structure.

- **Stemming:** This process consists in reducing inflections of the words to their root form. This root may or may not be the real etymological root: the *stemming* process does not really care about that. Instead, this root is a canonical form that represents the basic meaning of all the inflections of the word (i.e., "trouble", "troubling" and "troubled" will get stemmed to the canonical form "troubl"). Since this method uses a heuristic process to "cut off" parts of the words, these canonical forms sometimes might be inaccurate or completely incorrect.

- **Lemmatization:** The purpose of this process is akin to that of *stemming*, as it attempts to remove the inflections from words, effectively reducing them to their root form. However, it tries to do it "the proper way", instead of simply chopping off parts of the words. However, the overhead in computational complexity lemmatization adds (since it uses dictionaries and complex rule-based systems to map the inflected words), may me counterproductive to the system as a whole. Furthermore, some studies have found lemmatization to not provide a significant benefit, specially when compared to the less costly *stemming* [17].

- **Stopword Removal:** This method consists in removing the most commonly used words in a language (such as "a", "the", "is", "are", for English). The objective of undergoing such process is to remove the "less informational" words of a text, enabling the model to focus on the most important ones. Stopword removal usually also includes removal of *punctuation* (full stops, commas, semicolons, etc.) that may otherwise convolute the meaning of the text.

The chosen Natural Language Processing model, *BERT-base-uncased*, already includes some of these preliminary steps, but it is still important to mention that their usage or the aggressiveness with which to use them depends heavily on the field of application. This is, text preprocessing approaches are not directly transferable from task to task.

2.2.1.2 BERT

The word (document) generator used in this work is BERT (Bidirectional Encoder Representations from Transformers) [5], a Language Model developed by Google that has achieved state-of-the-art results in multiple NLP tasks by taking advantage of Transformers, deep learning models which exploit the usage of *attention mechanisms*.

BERT has a number of advantages over other known NLP models, which predisposed us to choose it to generate the text embeddings to feed our Machine Learning model:

- BERT solves the long-standing problem of uni-directional consideration of the context of each word. This is, until the proposal of BERT, NLP models only considered either the right or the left context of each of the words in a text, despite both being relevant to its meaning within the given sentence. BERT addresses this issue by considering both of the surrounding words (left and right) into the context of any given word.

- BERT is able to fit into multiple NLP processing tasks, without needing to re-train the complete model. The reason for this is the deliberate disentanglement of the training phase from the tuning phase; the latter may not be even necessary to achieve good results in a majority of generic tasks.

- The embeddings provided by the model consider not only individual embeddings for each token, but also pooled representations for each of the input sequences. This is crucial to our problem since we want to focus in obtaining numerical representations of the meaning of whole textual reviews.

Despite BERT's outstanding results in multitude of NLP tasks, the underlying architecture is relatively simple and intelligible. In the BERT model, each **input sequence** consists of the following elements:

- **[CLS]**: Special token that always occupies the first position of the input sequence, delimiting it. In the final hidden state it conforms a single pooled representation of the whole sequence, thus often used for classification tasks and Next Sentence Prediction (NSP), one of the goals used to pre-train the model.

- **[SEP]**: Used to separate pairs of sentences in the existing sequence (if applicable).

- $\mathbf{E}_n$: Numerical embedding of the token T_n; the sequence is split into these tokens by the *Tokenizer* prior to feeding it to the model.

On the other hand, for the *BERT-base* architecture in specific, which will be the one used in this proposal, the **output embeddings** are structured as follows:

- **[CLS]**: Output embedding that conforms a pooled representation of the input sequence. As we commented before, it is mainly used for tasks which require text classification.
- $\mathbf{O}_n$: Output embedding of each of the previously discussed input tokens. Each of these individual token output embeddings is a real-valued vector.

BERT's state-of-the-art performance is based on two novel tasks named Masked Language Model (MLM) and Next Sentence Prediction (NSP) which, as shown in Figure 2.6, are learned simultaneously during the pre-training process:

- **MLM** focuses on using the created tokens both as inputs and outputs for the pre-training process. A random subset of them is hidden or *masked* during the training (typically denoted as [MASK] tokens) and the model must predict these masked tokens as its output. This enables the model to achieve the bidirectional awareness of context we previously mentioned.
- **NSP** intends to allow BERT to understand relationship between consecutive sentences. To achieve this, a random pool of true consecutive pairs of sentences and randomly selected pairs are chosen, and the model will look to optimize its predictions.

2.2.1.2.1 Approaches to pre-trained BERT : As we mentioned before, BERT models are pre-trained over massive amounts of data. The user then has the choice of *fine-tuning* the model for the specific task, or using it *as is*:

- **Fine-tuning approach**: This method is relatively straightforward, as it consists in specializing the knowledge of the already pre-trained BERT model to perform better in the context of a given problem. To achieve this, the priorly pre-trained weights of the model are then fit to the problem at hand, as seen in Figure 2.7, using a low learning ratio. To fine-tune BERT to specific NLP tasks, no changes are required in the model topology or architecture. Instead, what changes is the selection of outputs of the model which will guide the supervised learning: depending on the specific task, both the last hidden state from the class label token ([CLS]) and/or the output embeddings of each individual token may be used [5]. Various examples of this output selection procedure are shown in Figure 2.8.

- **Feature-based approach**: In this alternative, the model is *frozen* and the pre-trained weights are not further fit to the problem at hand. Instead, the intermediate hidden states are used as an output of the model along the final hidden state. In the original work describing the BERT model [5], it is shown that when using a concatenation of the last 4 hidden states, the *feature-based* approach is surprisingly on par with *fine-tuning* the entire model. We will use this approach in the proposal described in Chapter 3, combining both the concatenation with the sequence pooling BERT provides. Since *BERT-base* outputs pooled sequence embeddings of 768 elements, we will work with real-valued vectors of $768 \times 4 = 3072$ elements as inputs to our proposed model.

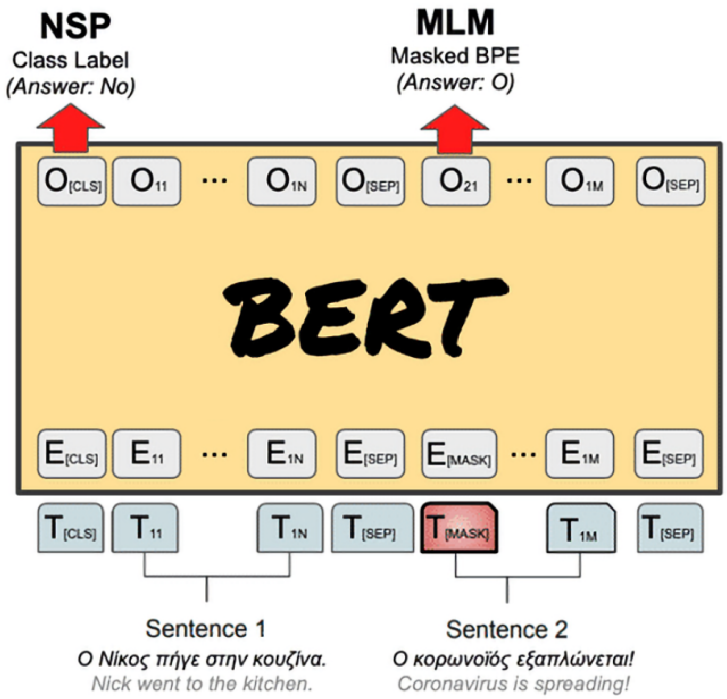

Figure 2.6: During the pre-training stage, BERT simultaneously learns two tasks: MLM and NSP. MLM predicts random tokens that have been masked from the original sentence (represented as a [MASK] token in the figure). At the same time, NSP predicts whether each pair of presented sentences are truly consecutive or not based on the output [CLS] classification token [4].

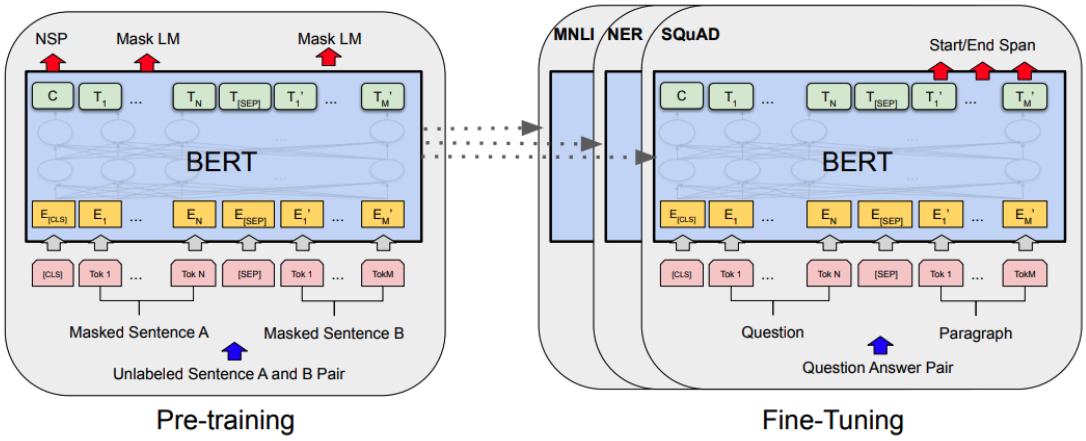

Figure 2.7: The left sub-figure depicts the *pre-training* process of the BERT model, governed by the separate tasks of MLM and NSP. The right sub-figure shows how BERT can be later fine-tuned to fit different NLP tasks, such as Question Answering (SQuAD), Named Entity Recognition (NER) or Text Classification: depending on the problem at hand, a different selection of outputs is employed to direct the *fine-tuning* of the model [5].

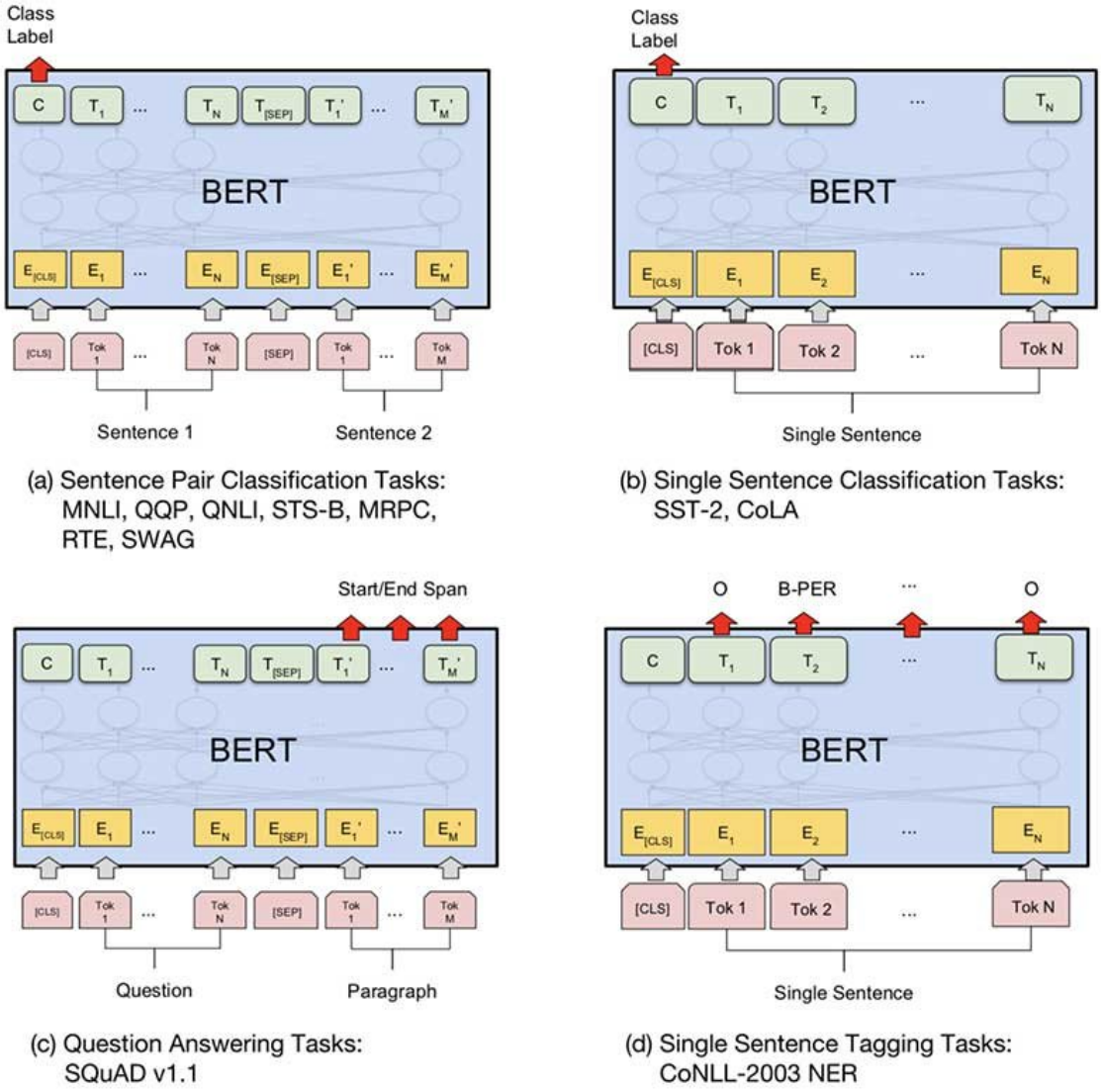

Figure 2.8: To specialize BERT to specific NLP tasks, a selection of outputs of the model is chosen to direct the *fine-tuning* process. For instance, in Question Answering tasks (c), the output embedding of the reference paragraph (which contains the necessary information to answer the question) is expected to be the desired answer, so it will be used to guide the supervised learning. Likewise, in Single Sentence Classification tasks (b), only the [CLS] classification label is used [5].

2.2.2 Image Classification

Unlike with written reviews, we have not tackled ourselves the task of obtaining embedded representations of the images that users attach to their reviews. These are already provided in the review image datasets gathered by the team belonging to the *Universidad de Oviedo*, which we will also use to test our proposals, as it has been previously discussed. Nevertheless, and for the sake of completion, we will briefly mention the method used for obtaining these low-dimensional image embeddings.

To obtain feature-rich representations of images, the used model was *Inception-ResNet-v2* [18], a Deep Convolutional Neural Network (DCNN), which takes 299×299 pixels RGB images as input, trained over one million images from the ImageNet database. The network is 164 layers deep, and provides as its output a vector of estimated class probabilities. We will not explore the functioning of this DCNN, but it can be mentioned that the extensive usage of residual connections, which are essentially "shortcuts" in the model, allows to mitigate the so-called "degradation problem" and accelerates the training, thus being able to train larger models. A review of this model's architecture can be viewed at Figure 2.9, extracted from Google's AI Blog [19].

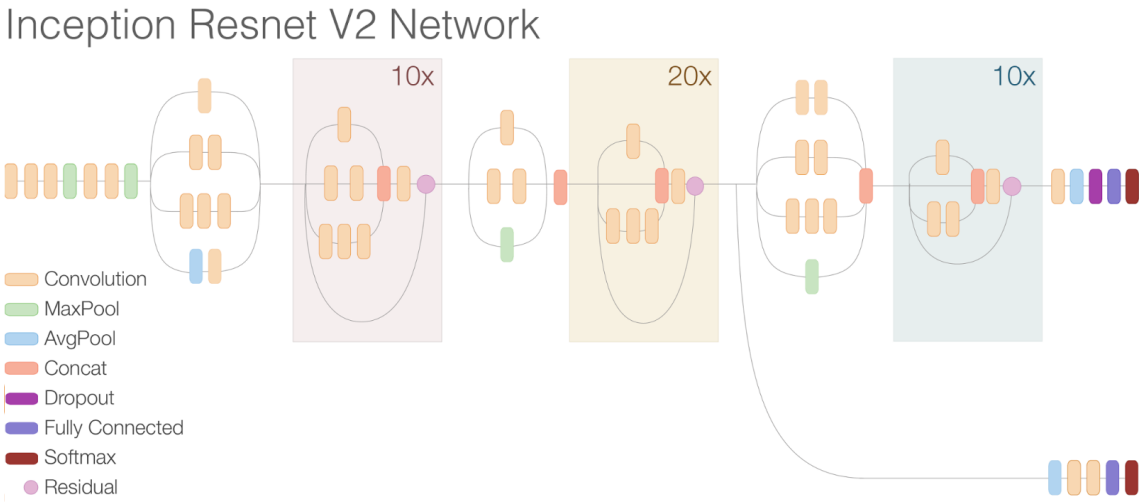

Figure 2.9: Schematic diagram of Inception-ResNet-v2, the network of choice used to create the image embeddings used in this project.

2.2.3 Matrix Factorisation in Recommender Systems

Matrix Factorisation (MF) is a widely used approach to tackling Recommender Systems, in contrast to more complex, black-box approaches like those that rely fully on Deep Learning. As our procedure to achieving personalisation is by itself another Recommender System, we will try to exploit Matrix Factorisation as a simple but powerful technique.

The key aspect of MF is its ability to map both users and items to a low-dimensional latent feature space. In an information system which tracks the data of n users and m items, the full matrix of user-item interactions has $n \times m$ entries. By factorising this matrix, we instead obtain two matrices U and V, which in total have $(n+m)d$ entries, where d is much smaller than m and n, and represents the size of the *latent space*. Alternatively, we can describe d as the amount of *latent features* we are using to describe our user's preferences (U) and the item's characteristics (V). As it can be observed in Figure 2.10, the lower-dimensional Item and User matrices may be arbitrarily small compared to the original full interaction matrix. In the Item Matrix, each column will represent the latent features of an item, while in the User Matrix, each row will represent one user's preferences in relation to said latent features.

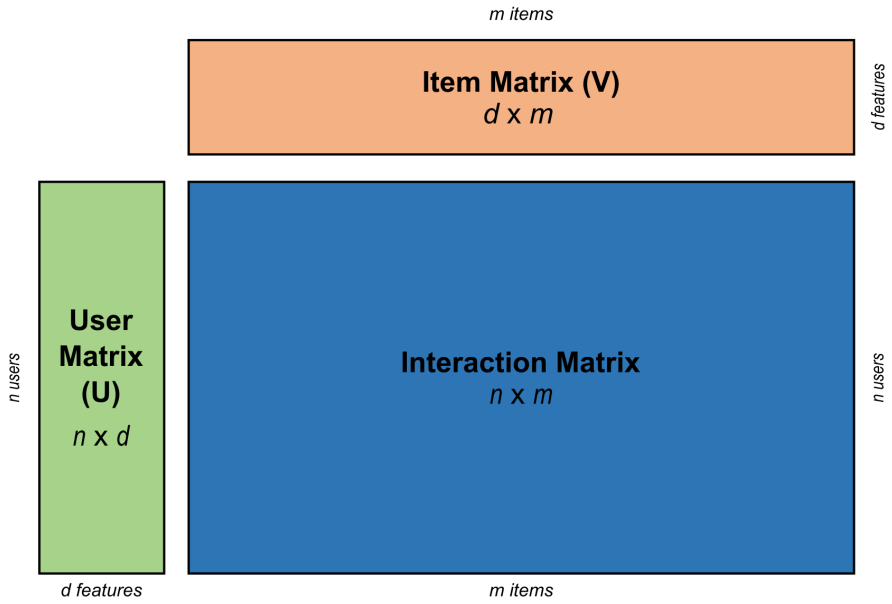

Figure 2.10: By using Matrix Factorisation techniques, the full (user-item) interactions matrix can be represented with two lower-dimensional matrices: the Item Matrix, which represents the latent features of each item, and the User Matrix, which represents the users' preferences in regard with those latent features [6].

Once we have obtained these two matrices U and V, predictions for $(user, item)$ ratings can be efficiently modeled as a simple dot product of vectors of size d, as depicted in figure 2.11. This enables Recommender Systems modeled with Matrix Factorisation techniques to

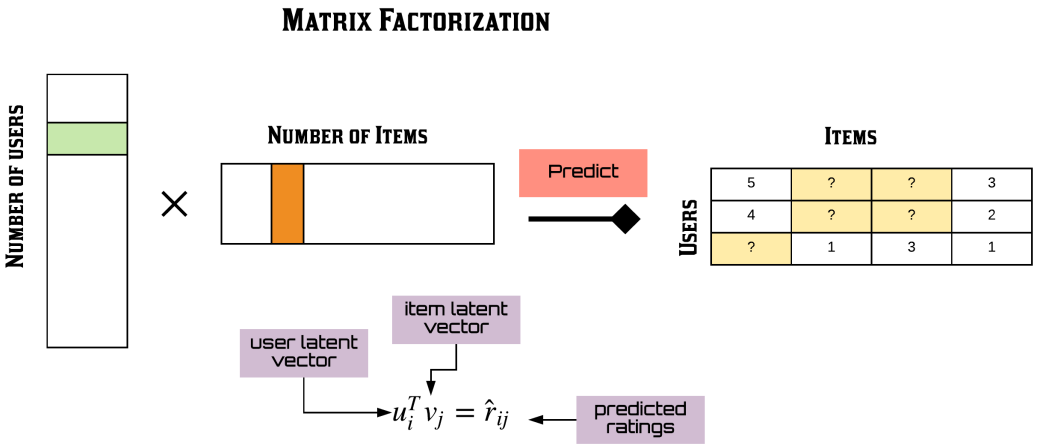

Figure 2.11: If we obtain the low-dimensionality matrices denoting the users' preferences (U) and the items' features (V), any (user,item) rating prediction can be obtained by computing a simple dot product between the row that represents the user in matrix U, and the column that represents the item in matrix V [7].

be computationally non-expensive compared to those that follow Deep Learning approaches.

With this base concept addressed, now the question relies on *how* this low-dimensionality, latent representation of the base (user-item) interaction matrix is obtained. Considering the available data from the *TripAdvisor* platform, this decision is governed by the differences between the types of Recommendation Systems explored in chapter 1.2.1:

- **Collaborative Filtering** is best used when we do not have any internal information about the items' characteristics or the user's preferences. In the context of Matrix Factorisation, collaborative filtering is achieved by decomposing the original matrix directly into the lower-dimensional item and user matrices. Usually, this factorisation is achieved through methods such as Singular Value Decomposition (SVD).

- As the gathered datasets contain relevant and semantic information about the items (in this case, written reviews and photos from the users), the scenario matches better that of **content-based filtering**. As we explored in sections 2.2.1 and 2.2.2, we have available low-dimensional, real-valued vector representations of our items. Moreover, these representations match the latent feature space of MF techniques. With this into consideration, we will follow a content-based approach, exploiting our image and text embeddings as direct latent-feature representations that together act as the factorised Item matrix; this will be discussed further in our method proposal during Chapter 3.

Chapter 3

Proposed Method

This chapter seeks to thoroughly define the Machine Learning model proposed to solve the recommendation personalisation problem described in Chapter 1.[1]

As it is typical in any recommender system, we will consider a set of users U and a set of items I. In the context of the TripAdvisor platform, U corresponds to the group of users that have written reviews or posted photos over the restaurants of a given city, whereas I corresponds to the set of those restaurants. However, it is important to remark that our intention is not to provide recommendations of the form $(\vec{u}, \vec{it})$, where $\vec{u} \in U$ and $\vec{it} \in I$. Instead, following a previous recommendation (or any other type of presentation of a restaurant to a user), our aim is to provide an explanation of it. To achieve this, we will use the existing photos and written reviews by other users of the same restaurant. Thus, we can define:

- $C(\vec{u})$: the set of texts/photos taken by user $\vec{u} \in U$
- $C(\vec{it})$: the set of texts/photos taken of the restaurant $\vec{it} \in I$

and immediately, by extension:

- $C(\vec{u}, \vec{it})$: the set of texts/photos taken by user $\vec{u} \in U$ at restaurant $\vec{it} \in I$

In this work, our aim is to provide users with text or images that reflect their personal preferences when presenting the restaurant to them. Our approach to achieve this will be to portray these preferences as a correlation to content *authorship*. This is, our concern will be to create a model able to identify which text/photographs of a given restaurant may have been uploaded or *authored* by a given user, as we understand that these are representative of that user's preferences regarding restaurants of the *TripAdvisor* platform.

[1] On the interest of reproducible research, the code for the proposed method is available at https://github.com/Kominaru/tfg-komi

Therefore, we will establish an Explainer for a RS where the set of users corresponds directly to U, the users of the platform, and the set of items corresponds instead to C, the group of text/photos (which we will name the *content*) of the restaurants of a given city, denoted as $C(\vec{it})$.[2] Every element of this matrix corresponds to a labeled pair

$$Pr(\vec{u}, \vec{c}) = \begin{cases} 0 & \vec{c} \notin C(\vec{u}) \\ 1 & \vec{c} \in C(\vec{u}) \end{cases} \tag{3.1}$$

where again $\vec{u} \in U$ represents a user, and $\vec{c} \in C$ is an individual piece of content attached by a user in a review. The label holds the meaning of the *authorship* of said content. Therefore, each element of the matrix $C \times U$ will correspond to 1 if the user is the *author* of that content, and 0 otherwise.

Logically, the predictions for the pairs that have not been labeled will be estimated through probabilities, like it is usual in any Recommender System. We can assume that, when a user reviews positively an item, the uploaded content intends to reflect the most important characteristics that gave rise to that positive review, which directly translates to the user's preferences. Consequently, when personalising a recommendation of an item $\vec{it}$ to a user $\vec{u}$, we can provide an explanation by showing to said user the content $\vec{c}*$ that best represents the user's preferences. To achieve this, we will predict which of the available pieces of content for that item the user is most likely to have authored.

Formally, we can define these predictions as

$$\vec{c}* = \underset{\vec{c} \in C(\vec{it})}{\arg\max}\, Pr(\vec{u}, \vec{c}) \tag{3.2}$$

with $Pr(\vec{u}, \vec{c})$ denoting the prediction about a user $\vec{u}$ being the author of content $\vec{c}$.

It is worth mentioning the following peculiarities:

- The proposed method for personalisation is bound to require **large amounts of data** to be able to provide good content recommendations. As in most models based on Recommender Systems, the scarcity of observed user interactions compared to the total size of the $U \times C$ matrix poses a challenge to successfully extracting latent features from users and items.

- Since we are modelling personalisation as a task of predicting *authorship*, we are effectively dealing with a **binary classification** problem. However, by definition, our dataset only contains *positive samples* (that is, the known data only considers that a certain user has authored a certain content). As the proposed model needs to train over

[2] Bear in mind that, even though we are referring to textual and photographic contents at the same time, both systems will be isolated from one another.

both positive samples and negative samples, we must also propose a method of fabricating negative samples, making various assumptions in the process. The quality of this *negative sampling* is bound to have a huge impact on the overall performance of the system.

- The proposed method intends to act as an additional personalisation layer beyond recommendation. In the previous step, any RS or even other types of systems (e.g. promotional, advertising or featured content systems) could be used without affecting the functioning of our model.

3.1 Dataset preparation

To prepare the information of each city's reviews to be fed to the system for training or testing, and prior to performing *test/train* split or *oversampling*, a series of pre-processing steps must be carried out over the raw *CSV* format datasets:[3]:

1. As the *CSV* file is being read, all entries with relevant null, lost or invalid values (such as entries with missing review text, malformed user id's, etc.) are filtered out.

2. The reviews are re-filtered to include only those that are *Positive* ($4 - 5\star$, as we can assume that only positive reviews are representative of the users' positive preferences), and are written in English language (as Bert-base-uncased, the chosen document embedding generator, is only trained over texts in English). Filtering them is trivial, as reviews written in other languages will have null values in their *review_preview* field. The reduction in size of the datasets after this step was discussed in Section 2.1, and can be appreciated in Figure 2.2.

3. To prevent unnecessary information bloat, the real user id's and restaurant names are replaced by new fabricated id's. Their correspondence to the real users and restaurants is kept in a separate structure not used during the training process. Additionally, other unuseful fields are dropped.

4. Due to the constraints of the BERT architecture, inputs are truncated in order to not exceed 512 words. This truncation policy does not hinder the overall availability or quality of textual review data, as reviews of such length are highly unusual in the *TripAdvisor* platform.

[3] The described pre-processing corresponds to our own work over the datasets including written reviews. As we mentioned in chapter 2, the datasets for photographic content were obtained in a pre-processed state. However, describing this process is equally relevant for said photograph datasets, as the ending result is equivalent with the purpose to maximize code re utilisation (and it is highly likely that their pre-processing was analogous to the one described in this section)

5. Regarding pre-processing of the text, we only do *punctuation removal* manually. We don't explicitly lowercase the text, but the BERT tokenizer does it internally [5].

6. All textual reviews are *embedded* using the Bert-base-uncased model, and saved in an auxiliary array loaded at the start of each training/test. The reviews get assigned a fabricated identifier, which corresponds to their index in said array. While it would be possible to use BERT as the lowest layer of our model, we have made the decision to create the document embeddings during pre-processing, as doing so **massively reduces training and testing times**. Needless to say, this comes at the exchange of an increased overhead in storage space usage (as it can be seen in Figure 3.1), and the datasets not being easily expandable.

City	Raw no. of Reviews	Raw CSV Size	Pre-processed dataset size
Coruña	10,089	8.00 MB	66.4 MB
Gijon	11,797	8.51 MB	70.7 MB
Delhi	148,541	68.9 MB	405 MB
Madrid	216,565	189 MB	1.07 GB
Barcelona	426,785	408 MB	2.45 GB
Paris	527,663	532 MB	2.75 GB
New York	517,904	533 MB	2.78 GB
London	1,000,365	1.02 GB	3.16 GB

Table 3.1: Comparison between the raw Text datasets in CSV format, extracted from the *TripAdvisor* platform, and the resulting pre-processed datasets after performing filtering, partitioning, negative sampling, oversampling and BERT embedding steps.

3.1.1 Partitioning

After the raw datasets have been pre-processed, a partitioning process is applied to split each of them into Training, Dev (Validation), and Test sets.

It is relevant noticing that, due to the context of the problem, a straightforward percent splitting of the dataset (e.g. 70% Train, 15% Dev, 15% Test) is not feasible. By the intrinsic rules of the *authorship* prediction method, and the *content-based* RS approaches, it is not possible for the system to know the preferences of users it has not "seen" before. As a majority of users only have uploaded one review to the platform (see Figures 2.1 and 2.3), testing over users the system has not trained with before would be bound to happen.

To solve this, we will apply a customised partitioning method that ensures that all tested users have been previously "seen" in the training phase, even if with minimal information. This method matches the one applied in [15], but we have replicated and recreated it with our own implementation.

Despite avoiding encountering strict *cold start* situations, the chosen partitioning method is relatively simple:

1. For all the users that have uploaded at least two reviews, one of them (which includes either one text or up to 4 photos) is randomly chosen and added to the Test partition. The remaining reviews are reserved for the Train+Dev partition. This way, the Test partition will not include any user which does not appear in the Train+Dev partition, avoiding potential *cold start* situations. Conversely, conducting this step implies that all the users available in the dataset will appear at least once in the Train+Dev partition and, in the case they have 2 or more uploaded contents, also in the Test partition.

2. Using the same procedure, the Train+Dev partition is split between Train and Dev partitions. This ensures that we will be able to accurately measure the training and validation progress without needing to worry about anomalies in the validation loss or accuracy caused by the *cold start* problem. Similarly to the previous step, all available users in the Train+Dev set for any given city will thus appear at least once in the Train partition and, if they have 2 or more uploaded contents, also in the Dev partition.

This process is applied to the datasets available for each city. The resulting Train, Dev and Test (additionally, Train+Dev) partitions are stored into separate files for subsequent statistic purposes. However, it is worth noticing that at this point we still only have *positive samples* in the partitioned dataset, that is, samples $(\vec{u}, \vec{c})$ such that $Pr(\vec{u}, \vec{c}) = 1$.

3.1.2 Negative sampling and Oversampling

Since we must fabricate all the negative samples needed to feed the proposed model for its training, it will be necessary to make some assumptions over the given set of user interactions. The most reasonable one, considering the context of *authorship* prediction, is that any given user will not have authored content that they have not uploaded themselves. Based on this, we propose the following sampling method:

- In **training** sets, for each positive sample $(\vec{u}, \vec{c})$, which denotes that user $\vec{u}$ uploaded content $\vec{c}$ of a restaurant $\vec{it}$, we will:
  - Add 10 negative samples $(\vec{u}, \vec{c}')$, where $\vec{c}'$ is a content about the same restaurant $\vec{it}$ but uploaded by a different user $\vec{u}'$.
  - Add 10 negative samples $(\vec{u}, \vec{c}'')$, where $\vec{c}''$ is a content about a different restaurant $\vec{it}'$ also uploaded by a different user $\vec{u}'$.
  - Add 19 copies of the positive sample $(\vec{u}, \vec{c})$, in compensation for the 20 new negative samples we added in the two previous steps.

- In **testing** sets, for each positive sample $(\vec{u}, \vec{c})$, we will add negative samples $(\vec{u}, \vec{c}')$ with all the contents of the same restaurant $\vec{it}$ that appear in the training set.

Figure 3.1, extracted from [15], summarises the construction of the final datasets (partitioning, negative sampling and oversampling). This procedure is done completely *offline*. That is, both partitioning and oversampling steps are undertaken statically, and the resulting Train, Dev and Test sets are always stored in the file system instead of doing it dynamically when training and testing the model. The purpose of this, as it was mentioned at the start of this section, is reducing training and testing times to maximize time efficiency, which also enables us to directly compare these times with those of the previous existing approach in [15]. This comes at a cost of a noticeable increase in storage space usage, which can be observed in Table 3.1.

After preemptively creating content embeddings and partitioning, negative sampling and oversampling the datasets, these would be now ready to be fed to the model for training and testing purposes. Tables 3.2 (image datasets) and 3.3 (text datasets) show the resulting sample distribution of each dataset after applying the pre-preprocessing steps discussed in this section.

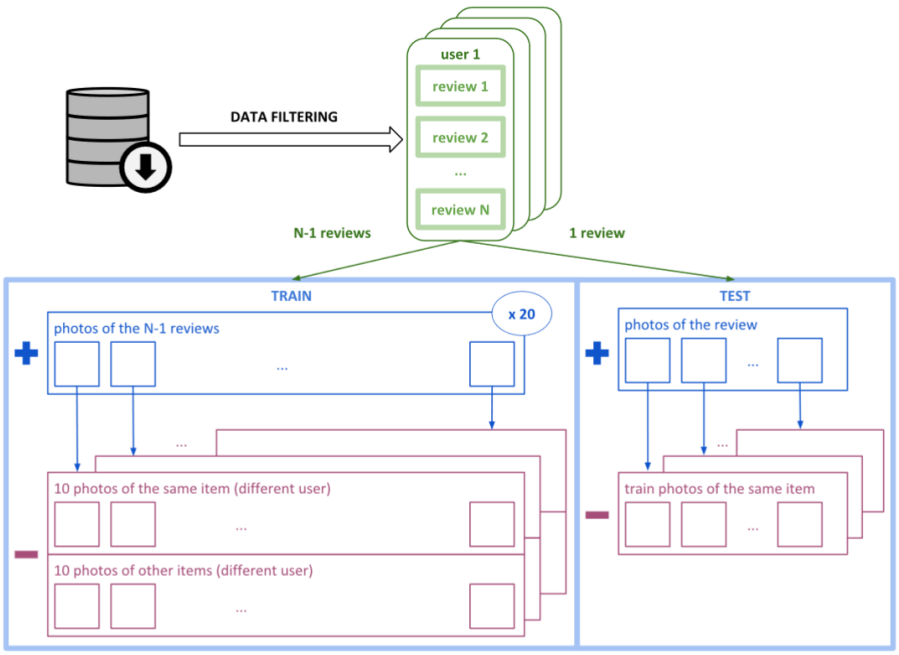

Figure 3.1: Construction process of the final datasets for each of the available cities, including review filtering, dataset partitioning, negative sampling and oversampling procedures.

(a)

	All data		
	Reviews	Unique Users	Unique Restaurants
Gijón	18,679	5,139	598
Madrid	203,905	43,628	6,810
Barcelona	150,416	33,537	5,881
Paris	251,363	61,391	11,982
New York	231,141	61,019	7,588
London	479,798	134,816	13,888

(b)

	Training Set		
	Reviews	Unique Users	Unique Restaurants
Gijón	16,302	5,139	598
Madrid	176,763	43,628	6,810
Barcelona	130,674	33,537	5,881
Paris	219,588	61,391	11,982
New York	196,315	61,019	7,588
London	416,356	134,816	13,888

(c)

	Testing Set		
	Reviews	Unique Users	Unique Restaurants
Gijón	2,377	1,023	346
Madrid	27,742	11,874	3,643
Barcelona	19,742	8,697	3,211
Paris	34,826	15,242	6,345
New York	32,048	16,842	4,135
London	63,442	30,393	8,097

(d)

	Oversampled Training Set		
	Reviews	Positive Reviews	Negative Reviews
Gijón	632,598	316,299	316,299
Madrid	6,842,232	3,421,116	3,421,116
Barcelona	5,027,898	2,513,949	2,513,949
Paris	8,382,630	4,191,315	4,191,315
New York	7,589,420	3,794,710	3,794,710
London	16,226,214	8,113,107	8,113,107

(e)

	Oversampled Test Set		
	Reviews	Positive Reviews	Negative Reviews
Gijón	201,629	2,377	199,252
Madrid	3,527,368	27,142	3,500,226
Barcelona	1,136,728	19,742	1,116,986
Paris	1,601,790	32,048	1,569,742
New York	6,015,682	34,826	5,980,856
London	9,571,605	63,442	9,508,163

Table 3.2: Statistics of the constructed Training (Train+Dev) and Test sets, for all Image datasets. (a) represents the original datasets after preprocessing. Below, (b) and (c) represent the state of the Train+Dev after performing partitioning, whereas (d) and (e) depict said partitions after performing negative sampling and oversampling procedures. While the process is not depicted here, since it is equivalent to the above shown, the Training set is partitioned into the individual Train and Dev sets and then sampled using the same policy as the one depicted in this table.

(a)

	All data		
	Reviews	Unique Users	Unique Restaurants
A Coruña	10,088	7,435	876
Gijón	11,796	8,332	722
Delhi	66,622	40,073	3,097
Madrid	216,474	136,795	9,058
Barcelona	426,643	265,582	8,137
Paris	527,437	303,045	13,861
New York	517,683	282,294	1,828
London	1,000,000	441,821	1,621

(b)

	Training Set		
	Reviews	Unique Users	Unique Restaurants
A Coruña	8,715	7,435	845
Gijón	10,233	8,332	703
Delhi	56,959	40,073	3,097
Madrid	180,502	136,795	8,981
Barcelona	350,691	265,582	8,092
Paris	434,761	303,045	13,788
New York	423,670	282,294	1,826
London	912,968	441,821	1,674

(c)

	Testing Set		
	Reviews	Unique Users	Unique Restaurants
A Coruña	1,373	1,373	394
Gijón	1,563	1,563	346
Delhi	9,663	9,663	3,097
Madrid	35,972	35,972	3,735
Barcelona	75,952	75,952	3,554
Paris	92,676	92,676	6,404
New York	94,013	94,013	1,176
London	87,032	87,032	897

(d)

	Oversampled Training Set		
	Reviews	Positive Reviews	Negative Reviews
A Coruña	346,570	174,300	172,270
Gijón	407,440	204,660	202,780
Delhi	2,266,380	1,139,180	1,127,200
Madrid	7,202,370	3,610,040	3,592,330
Barcelona	14,013,240	7,013,820	6,999,420
Paris	17,367,510	8,695,220	8,672,290
New York	16,943,040	8,473,400	8,469,640
London	18,259,350	9,129,680	9,129,670

(e)

	Oversampled Test Set		
	Reviews	Positive Reviews	Negative Reviews
A Coruña	46,350	1,373	45,012
Gijón	83,171	1,563	81,634
Delhi	915,212	9,663	905,649
Madrid	2,650,088	35,972	2,614,200
Barcelona	6,974,326	75,952	6,898,434
Paris	7,670,261	92,676	7,577,670
New York	9,439,419	94,013	9,345,408
London	7,402,472	87,032	7,328,956

Table 3.3: Statistics of the constructed Training (Train+Dev) and Test sets, for all Text datasets. (a) represents the original datasets after preprocessing. Below, (b) and (c) represent the state of the Train+Dev after performing partitioning, whereas (d) and (e) depict said partitions after performing negative sampling and oversampling procedures. While the process is not depicted here, since it is equivalent to the above shown, the Training set is partitioned into the individual Train and Dev sets and then sampled using the same policy as the one depicted in this table.

3.2 Proposed Model

To achieve the purpose of personalising recommendations, we present an XAI model to predict the *authorship* of uploaded content (photos or written reviews) by a given user:

- In the case of predicting authorship for **photographs**, the approach is based on Inception-ResNet-v2, as we briefly discussed in Section 2.2.2. The already available image embeddings consist in real-valued vectors of size 1536 which we will feed into the model for training and testing.
- When predicting the authorship of **text**, we will follow the *feature-based* approach on a pre-trained BERT-base-uncased implementation, which was discussed in Sections 2.2.1 and 3.1. The embeddings of all texts are computed and stored preemptively, then fed directly to the model, in order to reduce training and testing times. Effectively, this means all of the weights of BERT are frozen, and we will use the last four hidden states to build the contextual embeddings. As we already mentioned, this results in real-valued vectors of size $768 \times 4 = 3072$ for each of the written reviews.

Taking this into account, each of the created samples includes information regarding the context of the review (user id, restaurant id and review id), as well as the target output value, which is the *authorship* of that review in the context (whether that user actually uploaded that photo or text). This sample structure is shown in Table 3.4.

Until now, we have discussed the structure of the samples and input data our model receives, exploring how these are obtained through a disentanglement of the contextual embedding process from the architecture of the network itself. We will now describe the layers that conform this architecture for *authorship* prediction, summarised in Figure 3.2, which can be grouped in two separate blocks:

- The **Mapping** block has the purpose of codifying the input data (user identifier and content embedding) to the subspace of size d where we will exploit the usage of Matrix Factorization techniques:
  - Each user $\vec{u}$ is first codified with a *one-hot* codification, and then mapped into a d-dimensional embedding by means of a dense Embedding layer.
  - Each content $\vec{c}$, which is represented by a real-valued vector (of size 3072 in the case of text, and 1536 for images), is transformed by a Fully Connected (FC) layer to obtain a vector of reduced dimensionality with d elements.

  The results of this mapping block are two vectors of size d which, in terms of Matrix Factorisation and Recommender Systems, will represent the latent vectors of **user preferences** and **item features**, respectively.

Field Name	Type	Description
Common to all samples		
user_id	Integer	Unequivocal, fabricated identifier of the user
restaurant_id	Integer	Unequivocal, fabricated identifier of the restaurant. Not fed directly to the model (used only for evaluation and dataset statistics purposes).
review_id	Integer	Identifier of the contextual embedding of a piece of content. Size 1536 for photographs, and size 3072 for written reviews. All embeddings are stored in separate files to avoid excessive storage usage.
take or *is_dev*	Integer	Indicates whether the user is the real author of the review: 1 if they did author it, 0 otherwise. Named *take* in Training samples, and *is_dev* in the case of Testing samples.
Only in Testing samples		
id_test	Integer	Fabricated identifier that represents the testing case the sample corresponds to. This is necessary to evaluate the method with the metrics proposed in Section 3.3.

Table 3.4: Structure of the samples fed to the model. Fields *user_id* and the embedding corresponding to *review_id* are the direct inputs of the model, whereas *take* and *is_dev* (in Training and Testing sets, respectively) denote the expected output to carry out the supervised learning process and its evaluation.

- The **Matrix Factorisation** block applies the basic concepts of obtaining content-based recommendations with MF discussed in Section 2.2.3. As the output of the aforementioned mapping block contains the desired latent item features and user preferences in a low-dimensionality subspace of size d, it is now possible to apply a simple dot product between both vectors.

  The result of this dot product operation is a single Float value that effectively represents the similarity or *affinity* between the user and the content, and therefore, a joint prediction of the user's *authorship* of said content. Finally, a Sigmoid activation function is used to produce a probability output in the $[0, 1]$ range. This probability denotes what we defined as $Pr(\vec{u}, \vec{c})$ at the beginning of this chapter, viz. the probability that user $\vec{u}$ authored content $\vec{c}$.

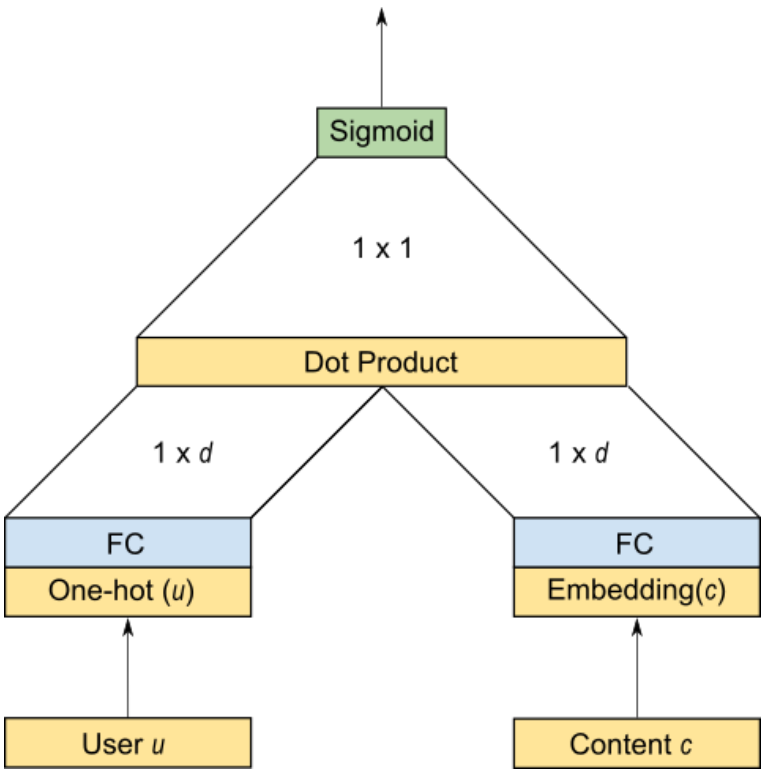

Figure 3.2: Overview of the architecture of the proposed model for *authorship* prediction. d denotes the size of the subspace where the dot product operates (or alternatively, the no. of latent features extracted from contents). In the case of Image authorship prediction, $d = 1024$, while for Texts, $d = 256$. The Embedding block represents the text or image embeddings we obtain with BERT and Inception-ResNet-v2, respectively.

In order to tune the model, a *grid search* was conducted with the intention of finding the best performing combination of model structure, training conditions and hyperparameters. In our model, these parameters are:

- The size **d** of the subspace the dot product operates over. Alternatively, d can be interpreted as the number of latent features extracted from the uploaded contents.
- The **batch size**
- **Learning Rate** (with Adam optimiser)
- Number of **epochs** used to train the model

Some references were considered during this search:

- Regarding the model to predict authorship of **images**, we took into account the tuning carried out by the group of the *University of Oviedo* over their proposed model for the

same problem [15]. However, it is important to take into account the more complex network topology of their model, ELVis, compared to our proposal.

- Regarding predictions over **text**, although we identified the existence of recommended hyperparameters in the original BERT publication paper [5], it is also mentioned that *"All of the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a down-stream task."*. We are instead using a *feature-based* approach for our NLP task, and regarding the results of this approach in said paper, the original authors only mention that *"Hyperparameters were selected using the Dev set."*.

Therefore, we used the aforementioned reference hyperparameters from [15] and the recommended for the fine-tuning approach in [5] as a guide to fit our model, employing the performance in the validation dataset during the training epochs as a metric to assess each combination. Additionally, as the loss function used to fit the model to the data and validate it, we selected the Binary Cross-Entropy (BCE) function. Its functioning consists in comparing two probability distributions (corresponding to the target and predicted labels, respectively), to give an insight into how well the model performs when making prediction over binary labels [20]. This makes it suitable for guiding the Adam optimizer to fit a model that seeks to solve binary classification tasks like our *authorship* prediction.

The final selected hyperparameters, training conditions and model structure parameters are shown in Tables 3.5 (when predicting authorship of texts) and 3.6 (when predicting authorship of photos). The large batch sizes indicated in these tables seek to accelerate training times, although we are aware that reducing them could mean a finer fitting of the model and thus better results. Other parameters like learning rate, epochs and d were adjusted strictly through the results obtained on the *grid search* process and the analysis of their performance in the validation datasets.

Parameter	Description	Dataset	Value
Batch size	Number of samples for each update step within the execution of one epoch.	A Coruña	256
		Gijón	256
		Barcelona	32768
		London	32768
		Madrid	32768
		New Delhi	32768
		New York	32768
		Paris	32768
Epochs	Amount of iterations used to train the model.	All cities	25
Learning rate (Adam)	Aggressiveness of the learning regarding minimization of loss function and fitting of the model to the data.	All cities	4e-5
Maximum review length	Maximum length for the written review to be embedded by BERT.	All cities	512
d	Size of the subspace the dot product is computed over. Corresponds to the amount of latent features extracted from texts and users.	All cities	256

Table 3.5: Selected hyper-parameters, training conditions and model structure parameters for the proposed text authorship prediction model.

Parameter	Description	Dataset	Value
Batch size	Number of samples for each update step within the execution of one epoch.	Gijón	256
		Barcelona	32768
		London	32768
		Madrid	32768
		New York	32768
		Paris	32768
Epochs	Amount of iterations used to train the model.	All cities	15
Learning rate (Adam)	Aggressiveness of the learning regarding minimization of loss function and fitting of the model to the data.	All cities	4e-5
d	Size of the subspace the dot product is computed over. Corresponds to the amount of latent features extracted from texts and users.	All cities	1024

Table 3.6: Selected hyper-parameters, training conditions and model structure parameters for the proposed image authorship prediction model.

In all cases and for both text and photo models, all configurations were trained with up to 100 epochs to ensure a thorough search for the best fit to the data was found. As neither the embeddings provided by BERT-base-uncased or Inception-ResNet-v2 are *fine-tuned* to the task (that is, we are not training the image and text classification models ourselves), we considered the possibility of needing a high amount of epochs to achieve a good fit. To complement this process, we considered the usage of **L2 Regularisation** and **monitored early stopping** [21] [22].

Regularisation is a method to avoid over-fitting in the data, by pushing the model to be less complex, or *parsimonious*. To achieve this, regularisation penalises large values of all the weights of the model in an attempt to keep them as closer to 0 as possible, lessening the impact of high-valued weights and bringing to model to a more uniform state, which should allow it to generalise better. However, an overly large regularisation value (*lambda)* will stop the model from being able to fit to the data, causing under-fitting. On the other hand, a too low *lambda* value will not prevent the model from over-fitting to the data.

In our case, we opted to attempt using **L2** regularisation, which adds the *square magnitude* of each weight as a penalty to the loss function. After intensive tweaking of the *lambda* parameter, we observed that the best found regularisation value did not cause any significant improvement to the validation loss function metric, and therefore decided to discard its usage to maximize simplicity of the model.

Early stopping is another technique extensively used to ensure that the model does not over-fit the data. This strategy consists in monitoring a metric that reveals when it is likely the best moment to stop learning. Typically, the comparison between the training loss and validation loss metrics is used to stop the training: the moment when the validation set consistently starts worsening its loss function, the learning process is halted. To carry out our grid search, we performed early stopping with the following parameters:

- **Monitored metric:** As it is common in this technique, a metric related to the validation set is used to decide when to stop the learning, as the purpose of this partition is to tune hyper-parameters and maximize generalisation while minimizing over-fitting. For this purpose we selected the **validation loss** function which, as we have mentioned beforehand, is computed through the Binary Cross Entropy (BCE) function.

- **Delta variation**: Smallest decreasing change in the aforementioned monitored metric that is interpreted as an improvement. This is, a minimal improvement in said metric (in our case, $\delta < 0.005$), will not qualify as such.

- **Patience:** Maximum number of epochs the monitored metric can last without improving until training is stopped. As validation loss tends to be erratic, specially during the

initial epochs, this prevents the training from ending prematurely, thus improving the flexibility of the learning process. This was set to **7 epochs**.

As it has been mentioned before, we considered for all datasets up to 100 epochs of training time. However, the early stopping strategy showed a strong tendency to halt the training process at approximately 15 epochs in the case of image models, and 25 epochs in the case of text models. We are aware that the previous research suggested noticeably different epoch amounts: 3,4 epochs are recommended in the original BERT paper [5], and the previous image-based personalisation research used 100 epochs for all cities [15]. However, it is worth noting that the usage of a *feature-based* approach in BERT is likely to imply a longer training of the model to fit the particular NLP task, while the usage of a simpler architecture is likely to reduce the amount of epochs needed compared to the complex topology used in [15].

After a consideration of the results, the consistency of the early stopping led us to decide using a fixed amount of epochs to simplify the logic of the training process: **25 epochs for text datasets**, and **15 epochs for image datasets**.

3.3 Evaluation Methods

In this section we will propose the methods to evaluate the results obtained by our model. Due to the inherent characteristics of the problem at hand, this has proven to be one of the toughest parts of the proposal.

Let us recall the problem we intend to solve: for any pair $\vec{u}, \vec{it}$ (where $\vec{u}$ is a user of the *TripAdvisor* platform, and $\vec{it}$ is a restaurant in said site), we aim to provide the piece of content $\vec{c}$ that is more useful to explain the recommendation of restaurant $\vec{it}$ to user $\vec{u}$. To achieve this, we proposed showing the piece of content with the highest $Pr(\vec{u}, \vec{c})$, which corresponds to the probability that the user actually *authored* that content.

Generalising this idea, we proposed *ranking* the photos/written reviews of the restaurant, according to what this predicted *authorship* is for each of them. The contents on top of the ranking would be more likely to have been written or taken by the user. This directly translates to the idea that, if a piece of content has been ranked high, it is likely to match the user's *preferences*, and therefore will be a good explanation about that restaurant. Logically, the main point of our proposal is that, for different users, this ranking may be completely different: not always will the same content be the best to recommend a restaurant to *any* user. Effectively, we will be able to *personalise* recommendations and their explanations.

Now delving into the evaluation methods themselves, the most evident way of knowing the users' satisfaction about the obtained rankings would be to involve the users in some sort of surveys or feedback system which could evaluate how satisfactory these predictions were.

This is commonly referred to as *explicit* evaluation, where users directly give feedback over the predictions of the model [23]. However, this approach was rather unfeasible at this stage of the project: the time and logistic constraints we have would have been problematic if trying to carry out an explicit evaluation, as this kind of surveys require meticulous planning, and need to be performed through a direct interaction with users in the platform providing the service (in our case, *TripAdvisor*). Therefore, while *explicit* evaluation is sure to be needed in the long term, we concluded that a different approach should be taken instead.

To tackle this issue, we conceived the idea of performing an *implicit* evaluation method, that is, one that obtains the experimental results by relying on the observed user behaviour and analysing it [23].

In Section 3.1 we discussed the steps undertaken to process all datasets, where we explored partitioning, negative sampling and oversampling. For the testing sets, we included unseen positive samples, and added all the contents of that restaurant that did not actually belong to the user as negative samples. Therefore, for each positive sample in the training set, we have a prediction of whether its author may have uploaded any of the other contents in said restaurant: this means that we can directly obtain the desired prediction ranking of that $(user, restaurant)$ pair.

Our evaluation procedure will make a reasonable assumption: if the model is able to place the user's own content on top of the ranking when mixed with all the other texts or photos, this means it was able to correctly predict its authorship, and moreover, **capture the user's preferences**. Consequently, for any $(user, restaurant)$ pair, the top ranked contents will mostly be representative of these preferences, thus being comparable to an *explicit* user satisfaction measure.

We are still to find a concrete metric to *implicitly* measure the quality of the rankings, based on how the user's own content is ranked on it. To achieve this, we will employ metrics extracted from the field of **Information Retrieval (IR)**, which is prominently known for dealing with methods to evaluate different types of rankings.

One possibility could be to measure how often our model ranks the user's real uploaded content, namely $\vec{c}$, among the top k positions of the ranking, as an indicator of the ranking's quality. As we have discussed, $\vec{c}$ is the only content of the ranking to have been authored by the user, what is usually named a *Relevant Document* in IR. Coincidentally, the measure we are proposing trivially matches the concept of calculating **Recall at** k. Formally, this metric is computed as

$$Recall@k = \frac{|\{Relevant\ Documents\}| \cap |\{Top\ k\ Retrieved\ Documents\}|}{|\{Relevant\ Documents\}|} \tag{3.3}$$

We can suppose this ubiquitous IR metric will be able to judge the overall quality of our

rankings. This supposition is backed by the literature. The work described at [23] acknowledges *Recall at k* to be adequate for the tasks of the type defined as *Find Good Items*, which matches ours.

Despite the viability of this metric, it is important to take into account that *Recall at k* will have a clear bias due to the intrinsic characteristics of our context. This is because not all restaurants have the same amount of available uploaded content. If any restaurant has *k* or less texts or photographs, the ranking provided to a user will always return an evaluation of $Recall@k = 1$, even if the user's own uploaded content is placed dead last. Conversely, for a restaurant with $|\{Reviews\}| \gg k$, the ranking will be prone to produce an evaluation of $Recall@k = 0$ that may not reflect its real quality.

Therefore, while we will maintain *Recall at k* as a basic but adequate measure to assess the quality of the produced rankings, our main tool of analysis of the results will be a second metric that does take into account the size of each of the rankings. This metric will be computed as

$$percentile(\vec{c}, \vec{R}) = \frac{pos(\vec{c}, \vec{R}) - 1}{|\vec{R}|} \times 100 \tag{3.4}$$

where $\vec{c}$ is the content uploaded by user $\vec{u}$ at restaurant $\vec{it}$, $\vec{R}$ is the ranking created by the model for the pair $(\vec{u}, \vec{it})$, and $pos(\vec{c}, \vec{R})$ is the position of $\vec{c}$ in said ranking. As the ranking is sorted in descendant $Pr(\vec{u}, \vec{c})$ order, the lower this *percentile* metric is, the better the ranking is considered.

Intuitively, this metric does allow us to take into account the variable size of $\vec{R}$. Take for example, that we rank the user's own content in 7th position of the ranking $\vec{R}$:

- If the restaurant has 10 photos, ranking 7th out of 10 yields a percentile value of 60%.
- If the restaurant has 1000 photos, ranking 7th out of 10 this yields a percentile value of 0.6%

As it can be observed, the metric captures the reasonable assumption that ranking the user's content 7th of 10 is worse than 7th of 1000, thus solving the issue we identified in the *Recall at k* metric.

Additionally, we will compute this metric multiple times, varying an *active users* threshold, that is, the minimal quantity of contents a user must have uploaded to be considered as "active". Therefore, for all n in the range $[1, 100]$, the median *percentile* metric will be calculated for all users with $|\{Contents\ of\ user\ \vec{u}\} > n$. This will allow us to have an idea of how our model performs when personalising both the explanations of inactive users and really active ones.

Although the complete experimental results will be discussed and evaluated in Chapter 4, we can showcase the functioning of our model through a clear example, illustrating the ideas behind the assumptions undertaken to devise the evaluation methods, in the context of predicting authorship for *photographs*:

Figure 3.3 shows the predicted ranking for a user in a specific restaurant. As it can be appreciated, the photos uploaded by them (top row) mainly show exterior areas of the restaurant. After computing the predicted probability $Pr(\vec{u}, \vec{c})$ for all of the photographs in the restaurant, these were ordered: the ones at the topmost of the ranking, which we would expect to be the ones that better match the user's preferences, do indeed show the characteristic exterior of the building, while the ones that focus on the food or the *dècor* get relegated to the lower positions. The user's real photo, marked with a DEV label, was placed near the top of the ranking. On the other hand, Figure 3.4 depicts the opposite situation: the user has a tendency to only take photographs of the food (top two rows), so the model places dead last in the ranking those that are not food-related. With this, we can conclude that our model was able to grasp the peculiar preferences of the user, and more relevantly, with the topic of *model evaluation* at hand, that our *percentile* metric can quantitatively measure the quality of these rankings.

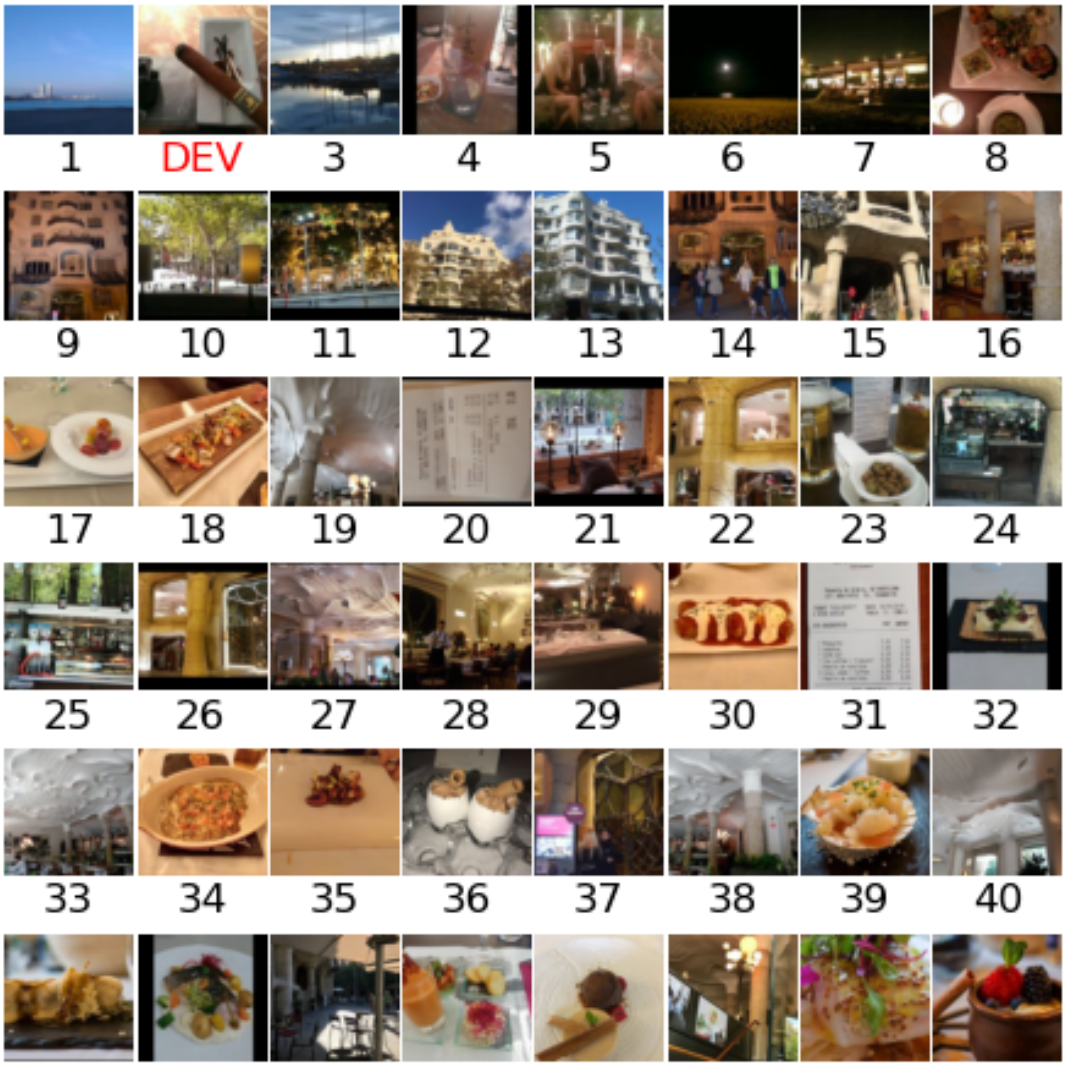

Figure 3.3: Ranking of the predicted authorship of a user for the restaurant in the famous Casa Milà, in Barcelona. The first row shows the eight photographs in the training set uploaded by the user, while the rest of the rows showcase the ranking of the photos ordered by the predicted *authorship* probability $Pr(\vec{u}, \vec{c})$ computed by our model. Qualitatively, we may observe that the user tends to take photos of the *exterior* of restaurants, not the interior dècor or food, and the model places photos of the exterior of the building in the first positions. The user's own picture, marked with a DEV indicator, is placed second in the ranking.

Figure 3.4: Ranking of the predicted authorship of a user for a different restaurant in Barcelona. The first two rows show the photographs in the training set uploaded by the user, while the rest of the rows showcase the ranking of the photos ordered by the predicted *authorship* probability $Pr(\vec{u}, \vec{c})$ computed by our model. In this case, it can be appreciated that the user likes taking photographs of the food, and never of the dècor or the exterior of the restaurant. Consequently, our model is able to place the food-related pictures on the higher positions of the ranking, while all of those that do not depict food are relegated to the last positions. The model is being able to understand the user's preferences from their training photographs.

Additionally, for comparative purposes, we will consider three methods to contrast the quality of our model:

- We will define two *baseline* algorithms to have an idea of how our method differentiates from trivial approaches:
  - The first baseline simply defines the ranking with a *Random* (RND) algorithm. This is, it will compute all probability predictions $Pr(\vec{u}, \vec{c})$ using a random uniform distribution in the range $[0, 1]$. To prevent biases, this probability is calculated 10 times for each corresponding $(\vec{u}, \vec{c})$ pair, and then averaged across those 10 executions.
  - The second baseline method bases its computations on the *Centroid* (CNT) of all of the user's uploaded contents across all restaurants. This way, we obtain a simple method that directly uses the embeddings created for written reviews and photographs. Each predicted *authorship* probability $Pr(\vec{u}, \vec{c})$ is computed as the inverse of the euclidean distance between $\vec{c}$ and the set of contents uploaded by the user. Therefore, the top-ranked (and therefore most representative) content in $\vec{R}$ will be the one closest to the centroid of the user's uploaded contents.
- Regarding evaluation of our model when predicting authorship of photos, we will also compare our results to those obtained by *ELVis*, the agent developed by the *University of Oviedo* and presented in [15] which proposes a similar approach (with a noticeably more complex model) that tackles the same problem of personalising with user's content on the *TripAdvisor* platform datasets. As the results of our approach and theirs can be contrasted directly, we will compare not only the model's performance, but also the differences in each model's training times. Therefore, we will be able to qualitatively evaluate the *performance vs. efficiency* trade-off in our models.

Chapter 4

Experimental Results

This chapter presents the insights into the learning process of the model, along with the training and validation metrics employed to assess it, as well as the experimental results in accordance with the methods established in Section 3.3, to evaluate the performance of our proposed model.

We will briefly comment the conducted learning process, along with the utilized metrics to evaluate it, in Section 4.1. During Section 4.2 we will evaluate and discuss the performance of our model according to the metrics we defined in Section 3.3, also comparing it to prior work on the field, in the case of images, and undertaking additional experiments derived from the initial results' discussion, in the case of texts.

4.1 Learning process evaluation

In this section, we detail the experiments used to ensure and measure the quality of the learning process, as well as to confirm that the model is able to generalise to previously unseen testing data. For all the experiments in this section, it is important to remind that we employed the best hyper-parameters configuration found during the *grid search* process (shown in Tables 3.6 and 3.5).

To carry out the experiments, we can classify the available datasets in two groups, for both image and text datasets[1]:

- As we observed in Section 2.1, datasets corresponding to small cities like A Coruña or Gijón are meant to work as a toy-sized testground to validate our learning process. In the following sections, we will use them to test how this scarcity of data affects the quality of the learning process and the experimental results.
- The rest of the cities, more international and with larger amounts of available data, will

[1] A Coruña and Gijón are present in the text datasets, but were not available in the selected Image datasets

be the focus in this chapter, as they are the most adequate to evaluate the experimental results of our model. The selected cities were London, Madrid, Barcelona, Paris, Delhi and New York.

We detailed the statistics and structure of all datasets in Tables 2.3 and 2.2, and explored the pre-processing procedures in Section 3.1.

4.1.1 Training and Validation metrics

In this subsection we describe the metrics used to assess the performance the learning process, considering the training and validation partitions, and subsequently the testing partitions to confirm that the usage of a *validation* partition was effective.

Our focus to conduct this fitting of the model is the BCE loss function, which is computed for both training and validation partitions, and calculated as

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(p(y_i)) + (1 - y_i) * \log(1 - p(y_i)) \tag{4.1}$$

where in our context, y_i is the *real authorship* of a given content, whereas $p(y_i)$ corresponds to the *predicted authorship* by the model.

Additionally, we also pondered the usage of metrics ubiquitously employed in the evaluation of classification tasks, such as *Precision*, *Sensitivity* and *Specificity*, or employing *confusion matrices*. However, we deemed that this set of metrics, even if considered a standard for evaluating Machine Learning models, was not useful to assess the performance of our proposal. Although our model is theoretically solving a binary classification task, our intention is to use the model's predictions, which will be in the range $[0, 1]$, to create a *ranking* of the contents most representative of the user's preferences, and employ it to create our personalised explanations. This is, the actual predicted binary label (*authored* or *non-authored*) is not relevant to us, as all we need to care about is the predicted probabilities and the ranking they create when sorted in descendant order. Due to this, we decided to discard these classic Machine Learning metrics and focus on evaluating the BCE loss function, with the main purpose of knowing when to stop the learning process of our model, and understand how the scarcity of data affects the evolution of said loss function curves.

Figure 4.1 shows the evolution of the Train and Validation BCE loss function over epochs for the text datasets, whereas Figure 4.2 displays the corresponding curves for the image datasets. As we would expect, the availability of data governs how good the fitting to the data of the model is: the aforementioned small-scale datasets of A Coruña and Gijon achieve a poor fitting to the curve in the validation dataset, whereas the cities with larger amounts of available data show a more clear fitting to the data.

It is worth mentioning that the photographic data seems to achieve a better fit to the models consistently throughout all cities. This is probably related to data distribution differences between image and text datasets, which we revealed during Section 2.1: on average, we have more information available from the photographs of a user than from their written reviews alone, as users may embed multiple images on one single review. This translates to a higher fitting to the data from the model in said Image datasets, whilst fitting the model to the Text ones is a harder task, as an overwhelming majority of users only have one written review available for training. This matches with our assumptions made in Chapter 3 about the need of large amounts of data to achieve competitive results in our model.

Likewise, Table 4.1 shows how satisfactorily the fitting of our model to the training and validation sets transfers to the real testground of the Testing datasets. The exhibited data represents the mean and standard variation of the BCE loss function values at epoch where the learning process is halted, averaged over 5 experiments and accompanied by its standard deviation. As it can be observed, the usage of a validation set to carry out the *grid search* in seek for the best hyper-parameters has been successful, as the fitting achieved for said set generalises satisfactorily to the isolated Testing datasets. As we discussed previously, these sets also match the perceived trend: the model is only able to obtain good fits to the data when large amounts of images or texts are available, and the higher sparsity present in the text datasets produces a less pronounced fitting.

City	BCE Loss Validation	BCE Loss Test
Coruña	0.469 ± 0.007	0.472 ± 0.005
Gijon	0.276 ± 0.010	0.287 ± 0.004
Delhi	0.505 ± 0.002	0.485 ± 0.002
Madrid	0.402 ± 0.004	0.393 ± 0.002
Barcelona	0.342 ± 0.001	0.292 ± 0.003
Paris	0.312 ± 0.002	0.296 ± 0.003
New York	0.292 ± 0.003	0.300 ± 0.001
London	0.302 ± 0.001	0.286 ± 0.001

City	BCE Loss Validation	BCE Loss Test
Gijon	0.216 ± 0.005	0.203 ± 0.002
Madrid	0.105 ± 0.004	0.156 ± 0.005
Barcelona	0.182 ± 0.002	0.174 ± 0.001
Paris	0.213 ± 0.003	0.210 ± 0.001
New York	0.118 ± 0.002	0.096 ± 0.004
London	0.135 ± 0.002	0.090 ± 0.001

Table 4.1: For all Text (left table) and Image (right table) datasets, comparison between the Validation BCE loss and the Test BCE loss in the epoch before stopping the learning process, averaged over 5 executions and accompanied by the standard deviation.

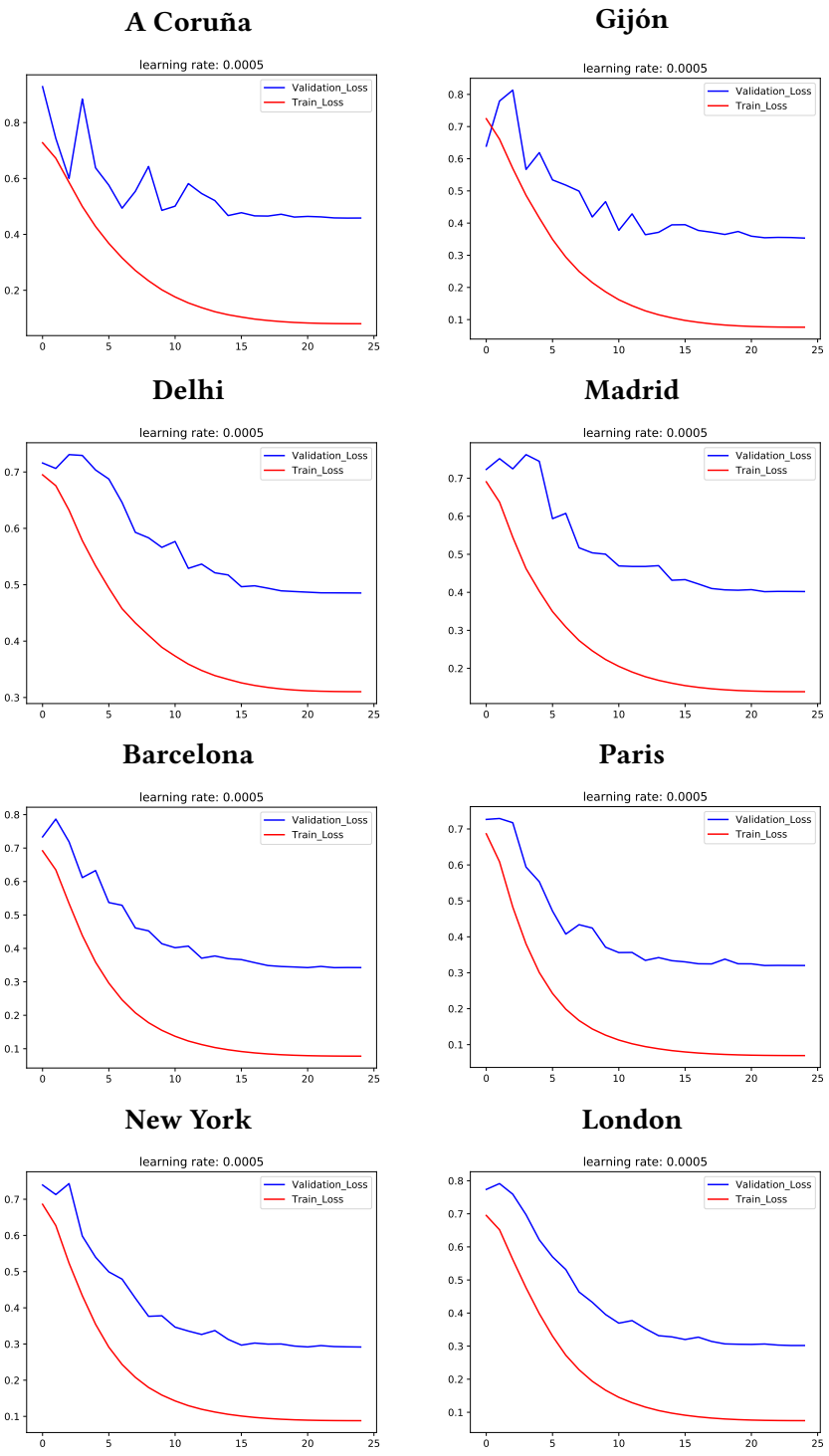

Figure 4.1: Evolution of the BCE loss function in the Train and Validation partitions for all text datasets, where the X and Y axes represent the current epoch and BCE loss, respectively. Each graphic corresponds to one individual execution of the learning process for that city.

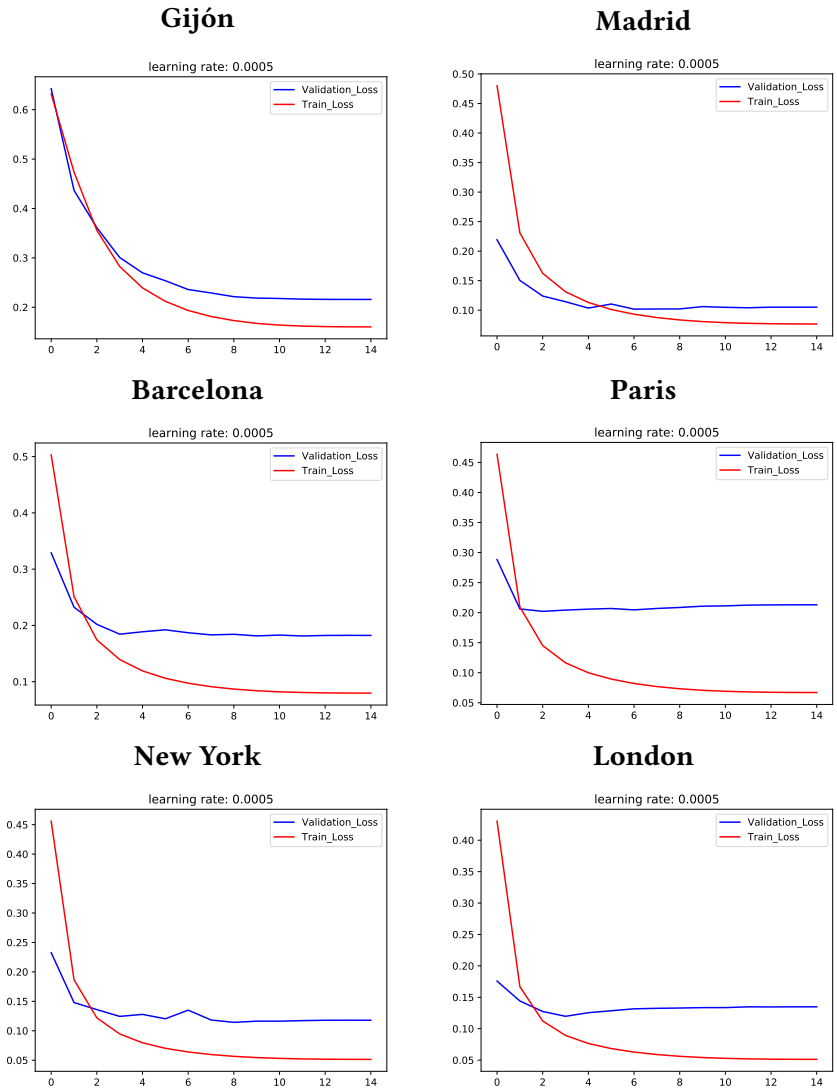

Figure 4.2: Evolution of the BCE loss function in the Train and Validation partitions for all image datasets, where the X and Y axes represent the current epoch and BCE loss, respectively. Each graphic corresponds to one individual execution of the learning process for that city.

4.2 Evaluation Results

In order to evaluate the model, we will employ the methods devised and described in Section 3.3: we will focus on *ranking-based* evaluation metrics, comparing it with baseline algorithms and the existing ELVis agent proposal. Additionally, we will also assess how the simplicity of our model compares with ELVis' more complex network topology, regarding model training times on large datasets.

We conceived our *ranking-based* evaluation metrics as a result of the similarities between our *authorship* prediction problem and the usual tasks Information Retrieval evaluates through ranking-based metrics. We made the assumption that placing the user's own content (which the model had not "seen" beforehand) on top of the ranking would mean that we are able grasp which contents best represent the user's preferences, placing them on the higher positions of the ranking and using them as personalised explanations to any RS.

To evaluate the quality of these rankings, we defined in Section 3.3 the two metrics to be used:

- ***Recall at k***: denotes whether the user's own content has been retrieved in the first k positions of the ranking. This metric will be shown in the results as a relative frequency calculation (which percentage of the Test dataset photos have been retrieved in the first k positions), for $k \in [1, 10]$.
- **Percentile**: computed as the relative position of the user's photo or text with respect to the ranking size. We will compute the average for all users on each dataset.

To analyze the effect of inactive users in these results, we will compute the *median percentile* measurement with 100 different "active user" thresholds for each city, computing the metric only with those samples where the user has more than n photos in the training set, for $n \in [0, 100]$. In the case of the *Recall at k* metric, considering this amount of possible thresholds is not feasible, but we will nevertheless compute it both for all users and for the ones with more than 10 text/photos in the training set, which we deem is an arbitrarily sufficient knowledge of the user to learn from (and thus able to reflect the difference in performance of the model between "inactive" and "active" users).

4.2.1 Photograph prediction evaluation

Using the metrics discussed at the start of this section, we will now present and discuss the obtained results for prediction of *authorship* over photographs.

Figures 4.3 and 4.4 show the obtained percentile metrics over the Test partition from each of available datasets, where the X axis represents the threshold employed to filter out users based on the amount of photos used for training (thus the amount of information known about the user), and comparing it to the Centroid (CNT) and Random (RND) *baseline* methods described in Section 3.3.

As expected, our model easily outperforms the two *baseline* methods, CNT and RND. For these comparative algorithms, the amount of training information is not really relevant, as their results are relatively consistent. *Random* is generally stable around the 50% percentile as expected, while *Centroid* is always the worst on all of the selected cities. Our model exhibits good results, and consolidates one of our prime expectations about the problem: its performance increases when applied to users from who we have more information. This is reflected in the percentile figures: the percentile value decreases (therefore a better result, since the users' real photos are closer to the top of the ranking) when we test the model with users with more photos in the training set.

We must however point out that, as we filter out "inactive" users by moving the threshold (X axis in Figures 4.3 and 4.4), the available number of test cases decreases significantly due to the scarcity of truly active users. To reflect this, these figures also include the available number of test cases for each value of the threshold, represented by the right vertical axis and its green-colored curve. This reduction in the number of test cases causes the erratic movement of the curves in graphics corresponding to the *Gijón* dataset, where there are little to no users with high amounts of uploaded photographs. In this particular city, the behaviour of the model is not representative when taking into account only users with more than 30-35 photographs.

Additionally, it is also worth noticing that, even if for our model the median of the percentile does increase when including inactive users in the testing (left side of each graphic), it is still much better than the two *baseline* methods. This means that, even near a *cold start* situation, we are able to produce decently accurate rankings.

Likewise, Table 4.2 shows for all six cities the percentage of test cases where the user's real photo is ranked in the Top-k positions (*Recall at k*), both taking into account *all* test cases (upper table), and considering only test cases about users with more than 10 photos in the training set (lower table), which we have observed is a sufficient amount of training information as to be possible for the model to learn about the user's peculiarities. These tables also contain the scores obtained by *ELVis* [15]. In all cases, we have filtered out the data from restaurants which have less than 10 photographs and included at most 100 of the available

ones, to mitigate the "optimistic" and "pessimistic" biases this metric is expected to have, which were discussed in Section 3.3.

As we can observe, for every individual method, all cities show relatively similar scores in this *Recall* metric. What is noticeably different, though, is the performance of the three methods. The baseline method CNT is the worst in every case; this is a trend we also observed when analysing the results derived from the *percentile* metric. We can derive interesting conclusions from this: while the image embeddings provided by the DCNN are good vector representations of the photos, these do not really distill the semantic rules that define each user's preferences; we must learn those through an active learning process.

Regarding our model's performance (MDL), it again exhibits a noticeably good performance when compared to the *baseline* methods. When placing the user's photo among the Top 10 positions of the ranking, our proposed model is around 15 percentage points better than *Random* in all cities when taking into account "active" users, and about 10 percentage points better if considering all users. As observed previously, the performance in Gijón is rather erratic due to the small size of the dataset. Conversely, the best scores were obtained in those datasets with larger photograph counts.

If we briefly compare our proposed model to ELVis, we can observe that our model's performance is competitively similar, even if ELVis comes marginally on top in most of the computed measurements. However, it's important to take into account the differences in model complexity and efficiency (our model is architecturally simpler, as well as less costly to train). We will address in depth this comparison between ELVis and our proposal in Section 4.2.1.1.

TOP	Gijón (2,005)				Barcelona (15,342)				Madrid (22,384)			
	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL
1	3.0%	1.5%	5.2%	**6.8%**	3.1%	1.5%	**8.3%**	7.1%	2.8%	1.4%	**7.6%**	6.6%
2	5.9%	3.5%	9.6%	**10.7%**	6.3%	3.3%	**14.4%**	12.3%	5.6%	3.1%	**13.4%**	11.5%
3	8.7%	5.0%	13.6%	**14.3%**	9.4%	5.4%	**19.7%**	16.6%	8.4%	5.1%	**18.8%**	15.5%
4	11.4%	7.7%	**18.3%**	17.6%	12.5%	7.9%	**24.6%**	20.9%	11.2%	7.3%	**23.5%**	19.2%
5	14.3%	10.1%	**22.2%**	21.9%	15.7%	10.2%	**29.4%**	24.9%	14.0%	9.8%	**27.7%**	22.7%
6	17.1%	12.7%	**25.9%**	25.4%	18.8%	13.0%	**33.5%**	28.3%	16.7%	12.3%	**31.7%**	26.1%
7	20.0%	15.2%	**29.8%**	28.1%	22.0%	15.9%	**37.5%**	31.7%	19.5%	15.0%	**35.1%**	29.2%
8	23.1%	17.6%	**32.6%**	31.0%	25.2%	18.9%	**41.0%**	34.9%	22.3%	17.8%	**38.3%**	32.1%
9	26.0%	21.0%	**35.2%**	34.2%	28.3%	22.3%	**44.4%**	38.3%	25.1%	20.9%	**41.3%**	34.9%
10	29.0%	24.9%	**37.8%**	36.7%	31.4%	26.1%	**47.1%**	41.5%	27.9%	24.3%	**44.1%**	37.5%
TOP	New York (28,531)				Paris (23,450)				London (53,901)			
	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL
1	2.6%	1.3%	**7.1%**	6.1%	3.8%	1.9%	**10.0%**	8.3%	2.5%	1.3%	5.9%	**6.0%**
2	5.1%	2.8%	**12.6%**	10.5%	7.5%	3.9%	**17.2%**	14.2%	4.9%	2.8%	**10.3%**	**10.3%**
3	7.5%	4.4%	**17.4%**	14.3%	11.3%	6.3%	**23.4%**	19.4%	7.4%	4.4%	**14.7%**	14.3%
4	10.1%	6.3%	**21.7%**	17.8%	15.0%	9.3%	**29.1%**	24.1%	9.9%	6.3%	**19.9%**	17.8%
5	12.6%	8.4%	**25.8%**	21.0%	18.7%	12.3%	**34.7%**	28.6%	12.4%	8.3%	**25.8%**	21.0%
6	15.2%	10.7%	**29.5%**	23.9%	22.4%	15.8%	**39.6%**	32.7%	14.8%	10.7%	**28.0%**	24.3%
7	17.7%	13.2%	**33.2%**	26.6%	26.2%	19.7%	**43.8%**	36.7%	17.3%	13.2%	**31.0%**	27.4%
8	20.2%	15.8%	**36.2%**	29.2%	29.9%	23.4%	**47.8%**	40.0%	19.8%	15.7%	**34.5%**	30.4%
9	22.7%	18.5%	**38.9%**	31.8%	33.5%	27.7%	**51.5%**	43.5%	22.3%	18.4%	**37.2%**	33.1%
10	35.3%	21.5%	**41.4%**	34.2%	37.3%	32.4%	**54.8%**	46.8%	24.8%	21.4%	**41.1%**	35.5%

TOP	Gijón (338)				Barcelona (3,023)				Madrid (4,578)			
	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL
1	4.3%	2.7%	8.9%	**11.2%**	4.0%	1.6%	**11.8%**	10.6%	3.6%	1.6%	**11.9%**	10.0%
2	8.3%	5.9%	**16.3%**	16.0%	7.9%	4.0%	**22.2%**	17.9%	7.5%	3.6%	**20.3%**	17.3%
3	11.8%	7.7%	20.1%	**21.6%**	11.8%	6.1%	**29.3%**	23.4%	11.3%	5.9%	**27.9%**	23.3%
4	15.7%	11.2%	26.6%	**27.2%**	15.8%	9.2%	**34.9%**	30.2%	14.9%	8.8%	**33.5%**	28.6%
5	19.6%	15.7%	29.9%	**33.1%**	19.9%	12.2%	**39.8%**	36.1%	18.7%	11.9%	**38.8%**	33.4%
6	23.7%	18.9%	35.2%	**36.4%**	23.7%	15.9%	**44.9%**	40.3%	22.4%	15.1%	**43.2%**	38.1%
7	27.9%	21.6%	40.2%	**41.1%**	27.7%	20.1%	**49.0%**	43.9%	26.1%	18.5%	**47.0%**	42.4%
8	32.5%	24.0%	42.9%	**43.8%**	31.8%	23.6%	**53.0%**	47.9%	29.9%	22.4%	**50.6%**	45.7%
9	36.9%	27.2%	46.7%	**50.0%**	35.9%	27.9%	**56.3%**	52.0%	33.6%	26.3%	**53.8%**	49.3%
10	40.9%	35.5%	52.1%	**54.7%**	39.9%	32.8%	**59.7%**	55.7%	37.3%	31.1%	**57.2%**	52.8%
TOP	New York (4,230)				Paris (4,625)				London (9,176)			
	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL	RND	CNT	ELVis	MDL
1	3.8%	1.6%	**11.6%**	10.3%	4.6%	1.9%	**13.9%**	12.7%	3.4%	1.7%	**11.5%**	9.6%
2	7.4%	3.7%	**20.1%**	18.0%	9.3%	4.3%	**22.5%**	20.6%	6.9%	3.5%	**19.3%**	17.6%
3	11.2%	5.6%	**26.9%**	23.9%	13.8%	6.9%	**29.7%**	27.6%	10.3%	5.4%	**25.5%**	23.6%
4	14.9%	8.0%	**32.4%**	28.9%	18.3%	10.3%	**35.8%**	33.6%	13.7%	7.8%	**30.9%**	28.7%
5	18.6%	11.3%	**36.8%**	33.4%	22.8%	13.5%	**42.0%**	38.4%	17.1%	10.6%	**35.7%**	33.2%
6	22.3%	14.2%	**41.4%**	37.7%	27.3%	17.4%	**47.6%**	43.6%	20.5%	13.9%	**39.9%**	38.2%
7	26.0%	18.3%	**45.3%**	41.4%	31.9%	22.6%	**52.3%**	48.2%	24.0%	17.2%	**43.8%**	42.5%
8	29.7%	22.3%	**49.2%**	45.0%	36.4%	27.4%	**56.5%**	52.5%	27.5%	20.5%	**47.4%**	46.4%
9	33.5%	26.2%	**52.4%**	48.7%	40.8%	33.1%	**60.4%**	56.1%	30.9%	24.3%	**50.1%**	49.8%
10	37.3%	30.1%	**55.3%**	51.6%	45.2%	39.3%	**64.4%**	60.0%	34.2%	28.2%	**53.1%**	**53.1%**

Table 4.2: For each Image dataset, comparison of the *Recall@k* or *Top-k* metric between RND, CNT, and our proposed model (MDL); larger values denote more test cases in the Top-k positions of the ranking, thus better results. The values in parenthesis are the number of test cases in each city. The upper table considers all test cases, while lower table considers only test cases about users with more than 10 reviews in the Training set. It is also important to mention that, to alleviate the known biases of the metric, we only include restaurants with at least 10 reviews, and use at most 100 of them.

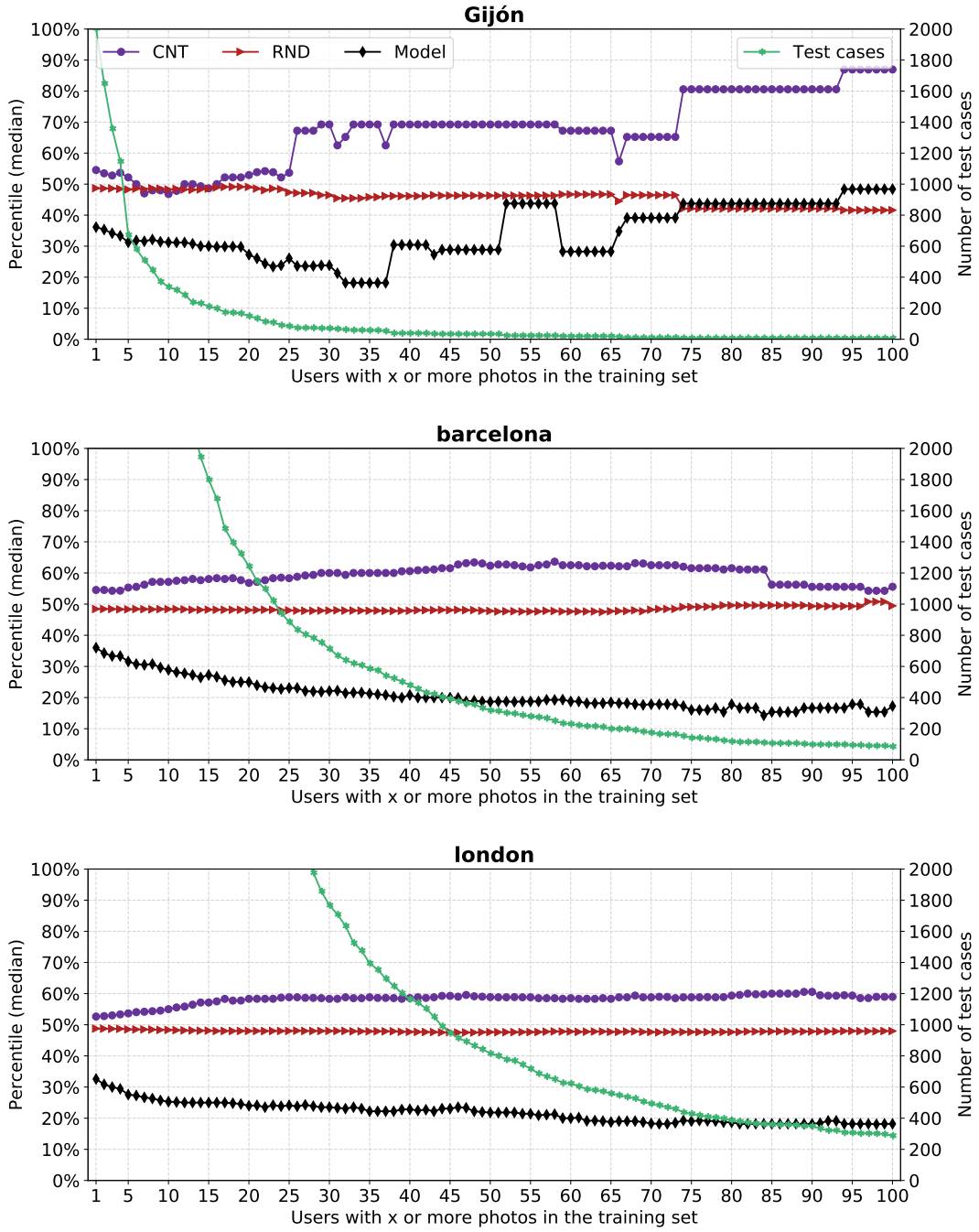

Figure 4.3: Median percentiles of the user's real photographs in the predicted rankings, for the Image datasets of Gijón, Barcelona and London. The X axis represent the minimum amount of photos by the user in the training test to be included in the computation. The green curve represents the number of test cases (total no. of rankings) available with each threshold value. It can be appreciated that our predictions are more accurate when we test the model on user's for which we have learned a lot from, which meets our expected behaviour for the model. It is important to notice that the lack of test cases in Gijón leads to an erratic behaviour of the model as soon as the threshold surpasses 30-35 minimum photos in the training set for the user.

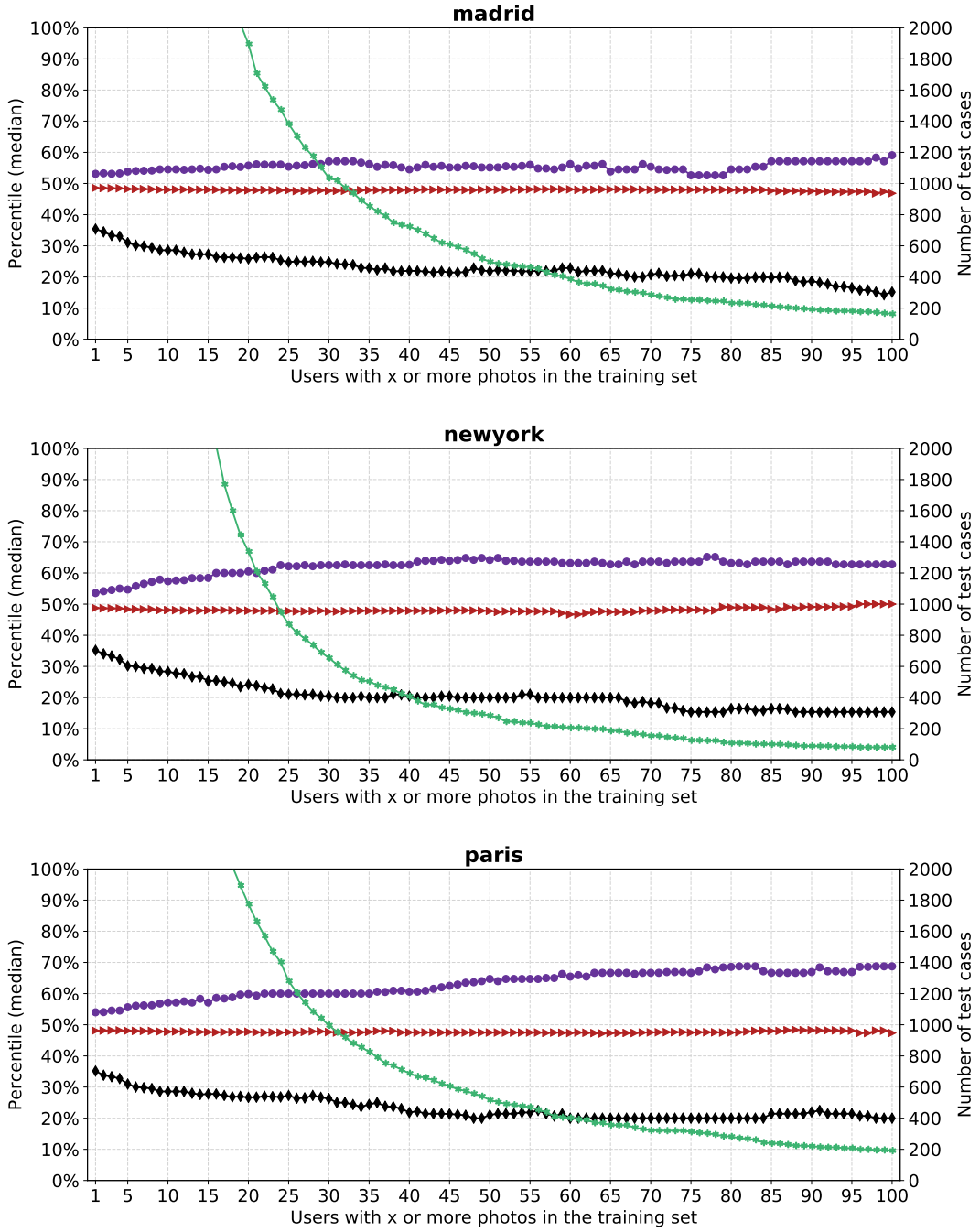

Figure 4.4: Median percentiles of the user's real photographs in the predicted rankings, for the Image datasets of Madrid, New York and Paris. The X axis represent the minimum amount of photos by the user in the training test to be included in the computation. The green curve represents the number of test cases (total no. of rankings) available with each threshold value. It can be appreciated that our predictions are more accurate when we test the model on user's for which we have learned a lot from, which meets our expected behaviour for the model. It is important to notice that the lack of test cases in Gijón leads to an erratic behaviour of the model as soon as the threshold surpasses 30-35 minimum photos in the training set for the user.

4.2.1.1 Comparison to the related Work

In this subsection, we will compare the performance and time consumption of the learning process between our proposed model and the reference agent ELVis by the *University of Oviedo* [15].

In the same task, ELVis utilizes a model architecture with a higher focus in deep-learning, whereas our model puts an emphasis in exploiting the simplicity of Matrix Factorisation techniques to still achieve ambitious results. Table 4.2 includes a direct comparison between ELVis' and our model's performance when scored through the *Recall@k* metrics. As it can be appreciated, our results are notably competitive, specially when considering the percentage of photos placed in the very first positions of the ranking, with ELVis being about 3 percentage points better at ranking the user's real photograph inside the Top 10 of the ranking. This trend is maintained if we compare the *percentile* metrics between both models, and also allows for an interesting insight: even though we are able to produce satisfactory recommendations for users with few information (less than 10 photographs), ELVis has a bigger advantage when we consider them in the calculations (around 7 percentage points, on average). We suspect that the higher complexity of their model allows for a better fit to the data of those users with little to no information.

However, as we have mentioned beforehand, an expected advantage of basing our model in Matrix Factorisation techniques was to have an edge regarding the computational cost of the learning process. ELVis' complex architecture implies in the long run a need to train the model for longer (in their case, 100 epochs). On the other hand, the architectural simplicity of our proposal allowed to obtain our best results with a training length of 15 epochs in the case of image datasets, as the only layers that require training are the ones that embed the users and images to the 1024-dimensional subspace we compute a dot product over.

Table 4.3 contains the comparison of training times between ELVis and our proposed model, averaged between 5 executions and including the standard deviation. As it shows, we are able to consistently obtain noticeably lower training times, across all cities, which reflects our expectations about the advantages of using a simpler model. This time efficiency is specially remarkable if we take into account the necessary re-training the model needs to undergo if we want to include new users into the system (due to the usage of internal one-hot encoding to codify the user, among other limitations): having a simpler model with noticeably lower training times mitigates the issue of having to train the model from scratch.

	Training Times (s)	
	ELVis	Proposed Model
Gijón	240 ±4.35	53.90 ± 1.80
Madrid	2465 ± 16.1	696 ± 5.02
Barcelona	1530 ± 10.4	436 ± 5.49
New York	2865 ± 19.5	746 ± 7.31
Paris	2940 ± 14.5	786 ± 6.48
London	5197 ± 48.9	1578 ± 17.4

Table 4.3: Comparison of training times between ELVis and our proposed model, run in the same environment. Times are averaged over 5 executions and standard deviation is included, to account for possible punctual performance issues.

4.2.2 Text prediction Evaluation

In this subsection, we will put on display and discuss the results obtained for the prediction of *authorship* over written text reviews.

For this matter, Figures 4.5 and 4.6 depict the results gathered for the percentile metrics in each of the cities we collected data from. As we mentioned when we discussed the results for image prediction, the X axis represent the minimum written reviews a user must have in the training set to be taken into account in that computation; this is, we are obtaining 100 different values per city depending on the minimum information known about the user. We compare our model with Centroid (CNT) and Random (RND), two *baseline* methods which we conceived for comparative purposes in Section *3.3*.

Again, the proposed model clearly outperforms the aforementioned baseline methods. Just like we had observed in our evaluation for image prediction, *Random* tends to stabilize in the 50% percentile, while the performance for *Centroid* varies depending on the city (most likely due to the scarcity of data in the the smaller datasets). In this case we can also make the assumption that the low performance of these algorithms makes the amount of training information for a given user irrelevant for predicting.

Our model exhibits noticeably good results, which allows us to once more conclude that our main assumption holds: the *quantity* of information known about a given user directly affects the *quality* of the model's learning about them. As it can be appreciated, the value of the calculated percentile metric decreases (thus a better performance of the model) when we require users to have more training text reviews in order to be taken into account in the calculation (right side of each figure). We also observe that the overall performance of the model improves with the increase of size in each dataset, which is congruent with our assumptions of needing large amounts of data to achieve good results due to the sparsity of interactions in all datasets.

In Section 4.2.1, when we evaluated photograph authorship prediction, we took notice of how the number of test cases rapidly decreases as we gradually consider only more "active"

users when computing the percentile metric (represented by the green curve and secondary vertical axis in Figures 4.5 and 4.6). In the context of text datasets, this scarcity of test cases for really active users is way more drastic, as we can observe if we compare the faster decrease in the aforementioned figures compared to Figures 4.3 and 4.4. This matches our conclusions drawn in Section 2.1, in which we inferred that this data scarcity would be a higher problem in Text datasets, where even if the user inactivity is equivalent, users can only upload one text per review. As an overwhelming majority of users had only one review uploaded to *TripAdvisor*, this leads to an overall shortage of test cases (and slightly erratic behaviours when taking into account only users with 50+ reviews in the smaller datasets).

On a different matter, Table 4.4 presents the results obtained regarding the *Recall@k* metric, both considering all test cases (upper table) and only test cases about users with more than 10 reviews in the Training set (lower table). It is relevant to recall that for all calculations we are filtering out test cases where the restaurant has less than 10 available text reviews (and included at most 100 of them), in order to avoid the results being notably altered by the expected biases of the metric, which we explored in Section 3.3.

In the case of the *Recall@k* performance, we see the results are relatively consistent across all cities. The **baseline** methods, *Random* (RND) and *Centroid* (CNT) exhibit roughly similar results. Regarding our own model's performance, it is consistently better than the aforementioned *baseline* methods. Overall, we are approximately 42 percentage points better at placing the user's photo among the Top 10 positions of the ranking compared to *Random* and *Centroid* if considering only "active" users, and about 20 percentage points better if considering all test cases. As we have observed previously, the low amounts of data in the smaller datasets (A Coruña and Gijón) leads to non-representative results, but allows to show how large amounts of data are needed to obtain reasonably good predictions due to the intrinsic data sparsity of the problem.

Overall, we can appreciate that, for all metrics, CNT obtains better results when predicting over textual data compared with predictions over images. We can theorize that the bi-directional contextual embeddings provided by BERT, which we explored in Section 2.2.1, are more akin to the semantic representations of users' tastes than those provided by the image embedding process undertaken by Inception-ResNet-v2, which we briefly discussed in Section 2.2.2. This is also a reasonable motive as to why we obtain overall better scores in Text prediction compared to Image prediction in all the devised evaluation methods.

TOP	A Coruña (1,373)			Gijón (1,563)			Delhi (9,663)			Barcelona (75,952)		
	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL
1	3.8%	2.5%	**8.8%**	2.8%	1.8%	**7.3%**	1.3%	1.6%	**2.5%**	1.3%	1.1%	**6.7%**
2	7.8%	4.9%	**15.2%**	5.2%	4.2%	**12.9%**	2.4%	3.6%	**5.2%**	2.5%	2.3%	**11.4%**
3	11.6%	8.5%	**20.9%**	7.8%	6.8%	**18.2%**	3.6%	6.3%	**7.9%**	3.8%	3.5%	**15.5%**
4	15.3%	11.0%	**25.6%**	10.3%	9.3%	**22.0%**	6.1%	9.8%	**9.9%**	5.0%	4.7%	**19.1%**
5	18.7%	14.0%	**30.3%**	12.9%	11.2%	**25.9%**	7.5%	11.8%	**12.1%**	6.3%	5.9%	**22.3%**
6	22.5%	17.3%	**34.8%**	15.2%	13.7%	**29.9%**	8.9%	13.6%	**14.4%**	7.5%	7.2%	**25.1%**
7	25.9%	20.3%	**39.3%**	17.6%	17.2%	**32.5%**	11.0%	16.3%	**16.7%**	8.8%	8.4%	**27.8%**
8	29.5%	24.6%	**43.4%**	20.2%	19.6%	**35.7%**	11.8%	18.9%	**19.8%**	10.0%	9.6%	**30.2%**
9	32.9%	28.1%	**47.1%**	22.6%	22.0%	**38.7%**	12.9%	21.7%	**21.9%**	11.3%	10.9%	**32.6%**
10	36.5%	32.0%	**50.9%**	25.2%	24.5%	**41.6%**	13.3%	25.3%	**24.3%**	12.5%	12.1%	**34.8%**
TOP	Madrid (35,972)			Paris (92,676)			New York (94,013)			London (87,032)		
	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL
1	1.9%	1.6%	**6.7%**	1.6%	1.3%	**8.7%**	1.0%	1.0%	**6.3%**	1.0%	1.0%	**6.3%**
2	3.9%	3.3%	**12.0%**	3.2%	2.8%	**14.5%**	2.0%	2.1%	**11.0%**	2.0%	1.9%	**10.7%**
3	5.8%	4.9%	**16.5%**	4.7%	4.2%	**19.1%**	3.0%	3.2%	**14.8%**	3.0%	3.0%	**14.3%**
4	7.7%	6.7%	**20.3%**	6.3%	5.7%	**23.2%**	4.0%	4.2%	**18.2%**	4.0%	3.9%	**17.4%**
5	9.6%	8.6%	**23.8%**	7.9%	7.2%	**26.8%**	5.0%	5.2%	**21.2%**	5.0%	4.9%	**20.2%**
6	11.6%	10.8%	**26.9%**	9.4%	8.7%	**30.1%**	6.0%	6.3%	**24.2%**	6.0%	5.9%	**22.8%**
7	13.5%	12.5%	**30.1%**	11.0%	10.2%	**33.1%**	7.0%	7.4%	**26.6%**	7.0%	6.9%	**25.2%**
8	15.4%	14.5%	**33.0%**	12.6%	11.9%	**35.8%**	8.0%	8.4%	**29.0%**	8.0%	7.9%	**27.5%**
9	17.4%	16.3%	**35.6%**	14.2%	13.5%	**38.3%**	9.1%	9.5%	**31.3%**	8.9%	8.9%	**29.5%**
10	19.3%	18.2%	**38.0%**	15.7%	15.2%	**40.7%**	10.1%	10.5%	**33.4%**	9.9%	9.8%	**31.4%**

TOP	A Coruña (18)			Gijón (29)			Delhi (360)			Barcelona (1,188)		
	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL
1	4.4%	5.6%	**27.8%**	1.7%	0.0%	**20.7%**	1.4%	0.6%	**11.9%**	1.5%	1.4%	**20.0%**
2	8.3%	5.6%	**44.4%**	5.5%	0.0%	**31.0%**	2.2%	0.6%	**19.2%**	2.8%	2.4%	**28.5%**
3	11.1%	5.6%	**50.0%**	5.9%	0.0%	**41.4%**	3.3%	1.1%	**26.1%**	4.1%	3.4%	**36.4%**
4	14.4%	5.6%	**50.0%**	9.7%	0.0%	**41.4%**	4.5%	1.9%	**29.7%**	5.3%	4.5%	**43.3%**
5	15.0%	5.6%	**55.6%**	12.1%	0.0%	**44.8%**	5.4%	2.2%	**33.9%**	6.7%	5.2%	**47.1%**
6	16.7%	5.6%	**66.7%**	14.5%	0.0%	**48.3%**	6.8%	2.8%	**38.6%**	8.1%	5.9%	**50.8%**
7	19.4%	5.6%	**66.7%**	15.9%	3.4%	**51.7%**	7.6%	3.6%	**41.1%**	9.3%	6.6%	**53.7%**
8	22.8%	5.6%	**72.2%**	18.6%	3.4%	**51.7%**	8.6%	4.4%	**43.9%**	10.5%	8.4%	**56.1%**
9	25.6%	16.7%	**72.2%**	21.4%	3.4%	**51.7%**	9.5%	5.6%	**45.3%**	11.8%	9.8%	**58.2%**
10	30.0%	22.2%	**72.2%**	23.8%	3.4%	**55.2%**	10.4%	7.2%	**48.3%**	13.1%	10.4%	**60.4%**
TOP	Madrid (681)			Paris (2,388)			New York (2595)			London (1467)		
	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL	RND	CNT	MDL
1	2.4%	1.6%	**19.2%**	1.7%	1.4%	**25.4%**	0.9%	0.8%	**17.8%**	0.9%	1.1%	**19.4%**
2	4.6%	3.8%	**30.8%**	3.5%	2.7%	**36.9%**	1.9%	1.6%	**27.8%**	1.9%	1.8%	**29.9%**
3	6.6%	5.7%	**39.9%**	5.2%	4.0%	**44.9%**	2.9%	2.7%	**35.3%**	3.0%	2.7%	**36.7%**
4	9.3%	7.3%	**45.4%**	6.9%	5.4%	**50.8%**	3.9%	3.4%	**41.0%**	4.0%	3.5%	**42.0%**
5	11.6%	9.4%	**50.1%**	8.6%	7.6%	**55.1%**	4.9%	4.5%	**45.0%**	4.9%	4.6%	**45.9%**
6	13.9%	12.5%	**54.9%**	10.3%	9.1%	**59.3%**	6.0%	5.4%	**49.2%**	5.8%	5.5%	**50.3%**
7	16.3%	14.2%	**59.0%**	12.0%	10.1%	**62.5%**	7.1%	6.6%	**52.4%**	6.9%	6.4%	**53.9%**
8	18.9%	16.3%	**62.1%**	13.7%	11.7%	**64.6%**	8.1%	7.4%	**55.3%**	7.9%	7.2%	**56.9%**
9	20.7%	19.2%	**64.0%**	15.4%	12.9%	**67.2%**	9.1%	8.9%	**58.0%**	8.8%	7.9%	**59.3%**
10	23.2%	21.4%	**65.9%**	17.1%	14.8%	**69.6%**	10.2%	9.9%	**60.2%**	9.7%	8.9%	**61.3%**

Table 4.4: For each text dataset, comparison of the *Recall@k* or *Top-k* metric between RND, CNT, and our proposed model (MDL); larger values denote more test cases in the Top-k positions of the ranking, thus better results. The values in parenthesis are the number of test cases in each city. The upper table considers all test cases, while lower table considers only test cases about users with more than 10 reviews in the Training set. It is also important to mention that, to alleviate the known biases of the metric, we only include restaurants with at least 10 reviews, and use at most 100 of them.

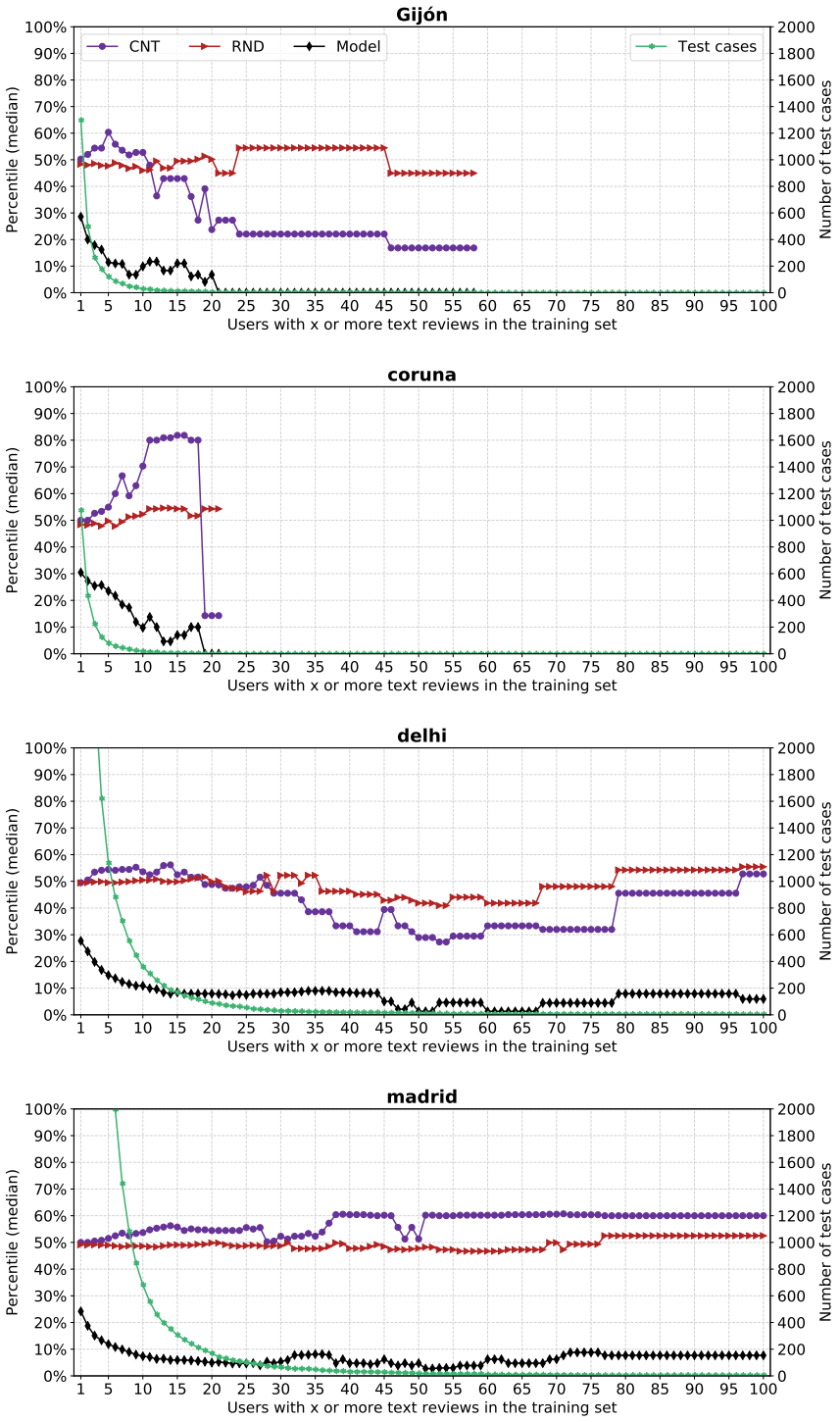

Figure 4.5: Median percentiles of the user's real photographs in the predicted rankings, for the Text datasets of A Coruña, Gijón, Delhi and Madrid. The X axis represents the minimum amount of photos by the user in the training test to be included in the computation. The green curve represents the number of test cases (total no. of rankings) available with each threshold value. The scarcity of data in A Coruña and Gijón precludes us from computing this metric when considering only highly active users.

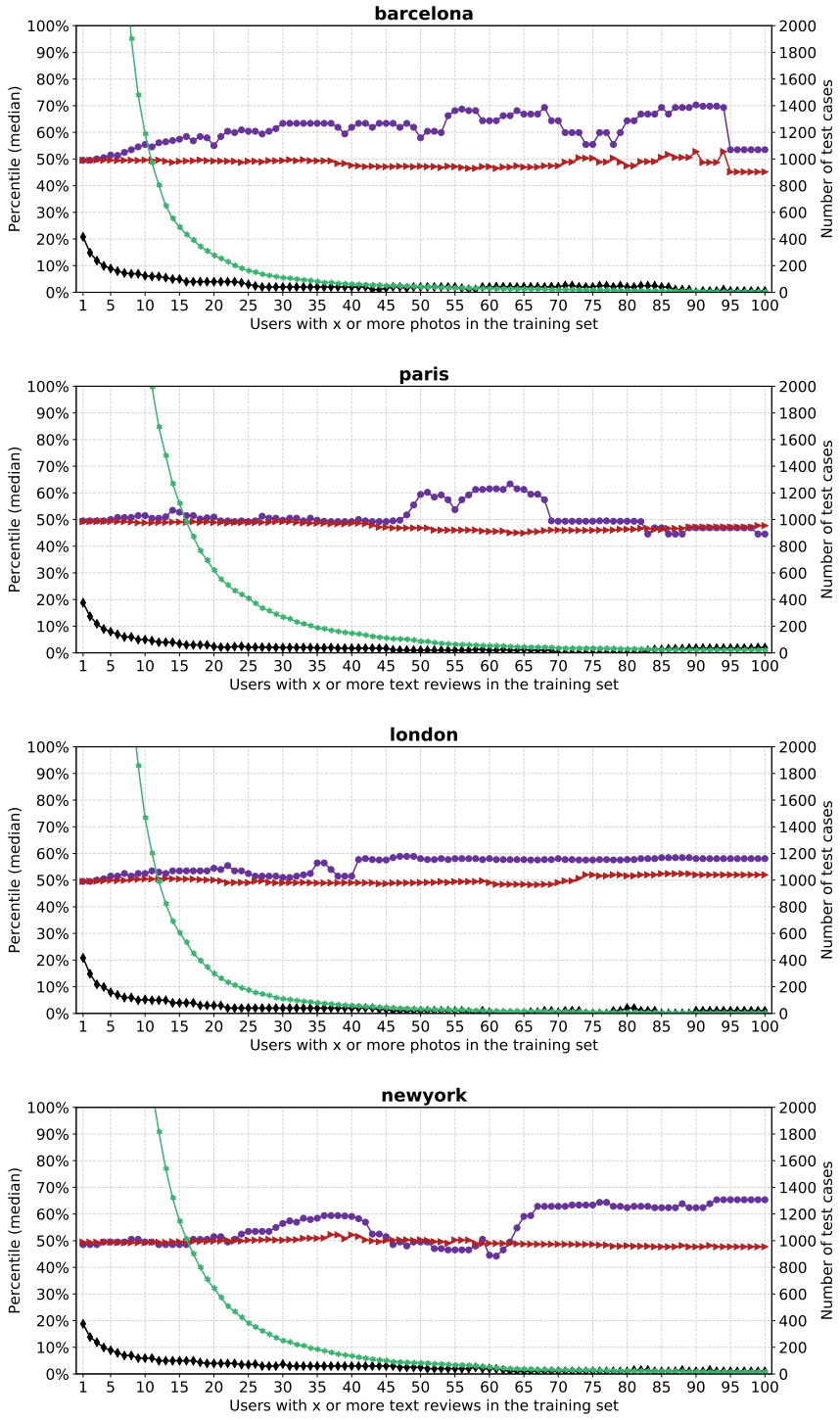

Figure 4.6: Median percentiles of the user's real photographs in the predicted rankings, for the Text datasets of Barcelona, Paris, New York and London. The X axis represents the minimum amount of photos by the user in the training test to be included in the computation. The green curve represents the number of test cases (total no. of rankings) available with each threshold value.

4.2.2.1 Addressing stylistic biases in text embeddings

As we discussed in Section 2.2.1, when we commented the functioning of BERT, this Natural Language Processing model focuses on capturing the semantic information of sentences or documents, by analysing bidirectionally the context of each of the tokens produced by the text.

However, it is relevant to recall that the chosen evaluation methods are *implicit* methods. This means we are making assumptions which, albeit reasonable, could mask up possible biases in our predictions, as our experimental evaluation heavily relies in measuring how good the model is placing the user's own picture on top of the created ranking. The predicament this implicit evaluation could carry is that, even if the intention is to predict authorship based on a *semantic* basis, these evaluation methods could carry instead an *stylistic* bias: the model may be learning the "meaning" of the texts through the user's writing technique, formality or characteristic words or catchphrases, instead of doing it by grasping the real traits of the reviewed restaurant (and therefore either inadvertently improving or worsening the obtained results).

To account for this theorized bias, we devised a method to attempt to get rid of the most prominent stylistic nuances of each review, so that BERT can focus on capturing the semantic concepts of the texts. To achieve this, we will:

1. Translate all reviews to a different language unrelated to English, so that the stylistic of each text is effectively *lost in translation* (for this matter, we chose Japanese, but any language not from the Indo-European family should have sufficed).

2. Translate all reviews back to English, in order to achieve a more uniformly styled English that can help BERT embed the texts on a strictly semantic basis.

An example of how this process can help get rid of stylistic nuances is shown at Figure 4.7: as we can see, the usage of a language where most words do not have a one-to-one relation to English, together with the translation patterns of the chosen tool (the official Google Translate API), tends to a convergence of the original texts towards a more uniform sentence structure and vocabulary choice, effectively mitigating the prominence of user-specific stylistic traits.

In order to analyse whether this procedure made a significant impact on the performance of the algorithm (whether that would be for the worse or the better), we re-ran our experiments for two arbitrarily selected cities: A Coruña and Barcelona. As we have mentioned beforehand, the results in A Coruña have had a tendency to be not too representative of our model, but this erratic behaviour could also easily reveal changes caused by the translation procedure described in this section.

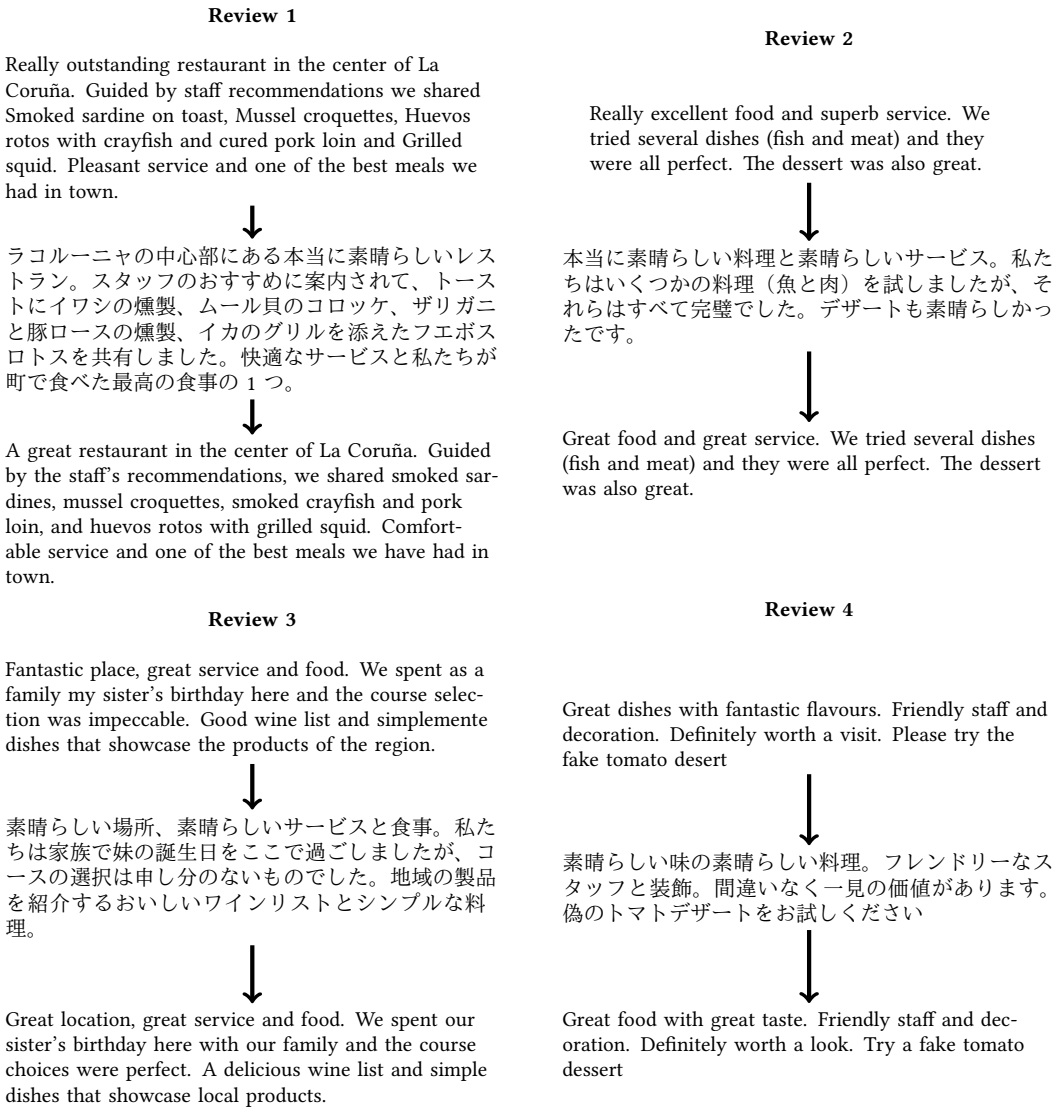

Figure 4.7: Four examples of the bidirectional translation procedure we designed to mitigate the stylistic bias the users are prone to have. Every text was translated to an unrelated language (in this case, Japanese) and translated back to English.

Figure 4.8 shows the comparison on our *percentile* metric between the original dataset (left-side figures) and the resulting datasets after undergoing the bidirectional translation (right-side figures). As we can observe, there exists a minimal variation between the performance on the unaltered and modified datasets. However, we do not believe this is caused by a clear weakening of a stylistic bias in the existing reviews, but rather the intrinsic non-deterministic nature of training this kind of Machine Learning model with data that has suffered small variations.

All things considered, despite being aware that the procedure chosen to distill the semantic information of the reviews may not be 100% effective, with the data at hand we can conclude that there is no evidence of a clear stylistic bias that may be guiding the overall performance of the algorithm. This reinforces the viability of BERT as a powerful tool to extract contextual, semantic-focused embeddings of the gathered text reviews.

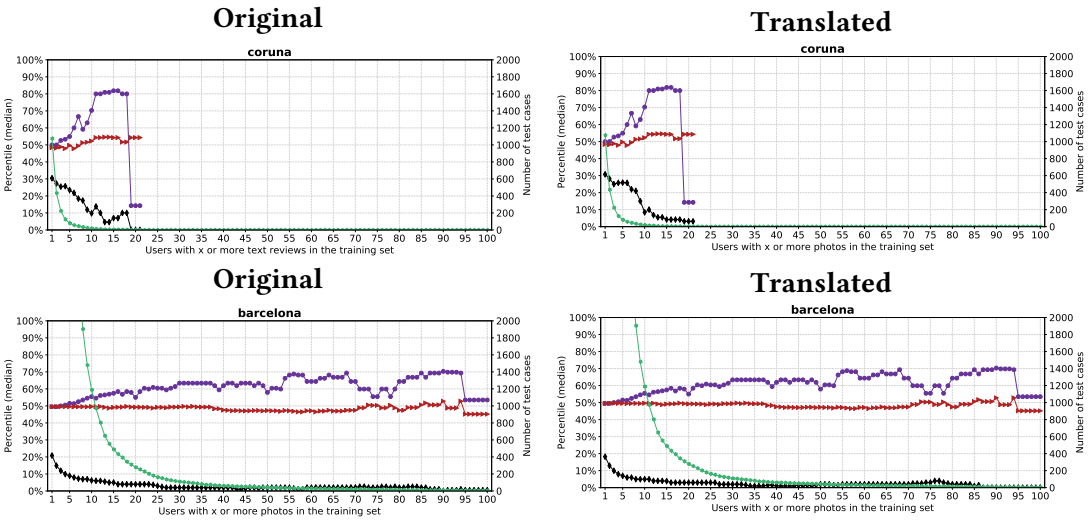

Figure 4.8: Comparison of the performance of the model between the original datasets of A Coruña and Barcelona, and after having bidirectionally translated all reviews to account for a possible stylistic bias.

‘

Chapter 5

Conclusions and Future Work

To recapitulate, in this work we have presented an Explainable Machine Learning system which acts as a tool to *personalise* recommendations by means of user-created content, such as photographs and text. Our project has been conducted within the context of the *TripAdvisor* platform, where we have gathered the images and texts that conform users' reviews to exploit them as utensils to achieve the desired explainability. Our main purpose was to acquire knowledge on each user's preferences, based on their interactions and existing reviews. Learning from the photos or texts a user has *authored*, we will obtain a latent representation of their tastes which will allow us to provide, for all (user,restaurant) pairs, images or texts that tally with the user's preferences. That is, for any of these pairs, we will present the existing text or photograph (about the restaurant) that best resembles the user's tastes in the platform. We have conducted our proposal on 8 different cities varying in size and cultural foundation, such as A Coruña, Madrid, London, New York, Delhi, etc.

With this project, we have attempted to contribute to the current state-of-the-art of Explainable Artificial Intelligence (XAI) and Recommender Systems (RS), by devising a model that is able to provide a concise and personalised explanation to each user, tailored to reflect their personal preferences. To achieve this, we have made usage of Matrix Factorisation techniques, as well as deep learning NLP and image classification models, tools which have become of uttermost significance in the field of Machine Learning in recent years.

Our proposed model is simple, straightforward, and fully disentangled from the previous recommendation step. Any RS may be used prior to generating an explanation, or none at all. In outline, given an arbitrary (user,item) pair, our model will personalise the presentation of that item to the user, by showing the most adequate text or photograph in terms of the user's preferences, from the set of existing text or photographs from said item. This way of integrating *personalisation* to a RS can be considered as novel within the XAI and RS research areas, and compared to ELVis [15], which also attempted image-based explanations, we achieve competitive results with a noticeable reduction in model complexity and computational cost.

5.1 Conclusions

Throughout the conception and development of this research work, we have made several interesting insights into the studied topics, context, methods, tools and experimental results:

- Regarding the underlying field of Recommender Systems, we have explored and discovered the manifold approaches to recommend items to users, the eminent relevance of RS in multiple sectors of modern life, but also the lack of personalised recommendations, specially those that exploit user-created content, which has been our focus of research.

- In relation to the methods used to construct our proposal, we have made an insight into the usefulness of using pre-trained Natural Language Processing and Image Classification models (BERT and Inception-ResNet-v2, respectively) to efficiently create our proposal, by saving significant time and computational costs with *transfer learning* approaches. Particularly, we have understood the ability of BERT to fit into numerous NLP tasks without requiring any additional tuning, and its contextual-intensive understanding of human language.

- With respect to Matrix Factorisation techniques, we have ascertained its viability in achieving data and computational efficiency in the field of XAI and RS, while at the same time exploiting our image and text embeddings as representations of latent item features and user preferences.

- Concerning the gathered data and its surrounding context, we have witnessed the challenges that a real-world, relatively unexplored context poses in relation to conducting Machine Learning-focused research. The non-existence of pre-labeled datasets, or any sort of explicit interaction between the real users and most of the extracted samples, proved to be a demanding obstacle towards being able to train and evaluate our model. Moreover, the intrinsic scarcity of data, the sparsity of interactions in the context of the *TripAdvisor* platform, added to the absence of direct negative samples, was specially troubling. Nevertheless, by means of reasonable ground truth assumptions, as well as *implicit*, context-specific evaluation methods, we were able to conduct a satisfactory experimentation with our model.

5.2 Future Work

As far as future work in this field of research is concerned, we have identified the following objectives:

- It is of our interest to explore **more reliable methods to evaluate our results**. In the proposal detailed in this work we took advantage of *implicit* metrics and evaluation procedures; it is true that these are based upon reasonable assumptions over the user's preferences regarding *authorship*, but it would be better to have at our disposal evaluation methods that are fully faithful to the final desired usage of the model: presenting the content that is most akin to the users' tastes, rather than limiting itself to placing the user's real content on top of the ranking. The only conceivable way to achieve this, as we discussed during Chapter 3.3, would be to use *explicit* evaluation methods, likely in the form of surveys or other forms of feedback through the actual *TripAdvisor* platform or some sort of mock system that allows to evaluate our model with guarantees.

- An immediate target is to **obtain further training data and generate more datasets of other well-known cities and contexts**. While there is an acceptable variety in our existing datasets, it is also true we aim to broaden the functioning of our model to a more global perspective, including data from other areas of the world with distinctive cultures, where the user behaviour may be different from the observed until know.

- In the context of our model's functioning with textual reviews, we plan to address the possibility of expanding the system to support **multi-lingual personalised recommendations**. Logically, it would be desirable to be able to exploit the existing data in all possible languages to both increase the usability of our system and also its ability to learn thanks to the increase in data samples. One possibility would be to use a multi-lingual BERT approach, for which there exist pre-trained models. Another possible method to achieve this multi-lingual system sparks from the experiment carried out in Section 4.2.2.1: we observed that applying translation does not seem to meaningfully alter the *semantic* concepts of each text. We could take this to our advantage and, while maintaining the current monolingual BERT, use Machine Translation techniques to efficiently have all existing reviews at our disposal in translated English language, and then conveniently translate the selected text to the user's language when presenting the restaurant to them.

- We forsee the exploration of **Few-shot learning techniques** to incorporate into our proposal. The currently presented algorithm performs satisfactorily, but we had expected and can appreciate a decline in the performance when little to no information is known about users (which is a major occurrence in *TripAdvisor* and similar platforms), or in cities with low available review data. In order to maximize the usability of our approach, it would be beneficial to consider the usage of models which are able to address our *authorship* prediction problem with few samples.

List of Acronyms

BCE Binary Cross Entropy. v, x, 40, 52–55

BERT Bidirectional Encoder Representations from Transformers. iv, v, ix, 9, 20–24, 31, 32, 37, 39–41, 43, 68, 70, 72, 73

CNT Centroid. x, xi, 49, 57, 59, 63–65

DCNN Deep Convolutional Neural Network. 25

FC Fully Connected. 37

GDPR General Data Protection Regulation. 3

IR Information Retrieval. 44

MF Matrix Factorisation. 8, 26, 27, 38

MLM Masked Language Model. iv, 21–23

NLP Natural Language Processing. iv, 18, 20, 21, 23, 24, 43, 71, 72

NSP Next Sentence Prediction. iv, 20–23

RND Random. x, xi, 49, 57, 59, 63–65

RS Recommender Systems. 1, 30, 32, 71, 72

SVD Singular Value Decomposition. 27

XAI Explainable Artificial Intelligence. 3, 37, 71, 72

Bibliography

[1] L. Ngoupeyou Tondji, "Web recommender system for job seeking and recruiting," Ph.D. dissertation, 02 2018.

[2] Vox, 2018. [Online]. Available: https://www.youtube.com/watch?v=axCBA3VD5dQ

[3] D. Ferreira, "What are sentence embeddings and why are they useful?" Mar 2020. [Online]. Available: https://engineering.talkdesk.com/what-are-sentence-embeddings-and-why-are-they-useful-53ed370b3f35

[4] J. Koutsikakis, I. Chalkidis, P. Malakasiotis, and I. Androutsopoulos, "Greek-bert: The greeks visiting sesame street," 08 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[6] S. Venkateswaran, S. Senthilkumar, and S. P. Thandapani, "Recommendation systems: Collaborative filtering using matrix factorization — simplified," 2019. [Online]. Available: https://medium.com/sfu-cspmp/recommendation-systems-collaborative-filtering-using-matrix-factorization-simplified-2118f4ef2cd

[7] J. Le, "Recommendation system series part 4: The 7 variants of mf for collaborative filtering," Jun 2020. [Online]. Available: https://towardsdatascience.com/recsys-series-part-4-the-7-variants-of-matrix-factorization-for-collaborative-filtering-368754e4fab

[8] N. T. Blog, "Netflix recommendations: Beyond the 5 stars (part 1)," 2012. [Online]. Available: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

[9] B. Dean, "Netflix subscriber and growth statistics: How many people watch netflix in 2021?" 2012. [Online]. Available: https://backlinko.com/netflix-users

[10] K. Jacobson, E. N. Vidhya Murali, B. Whitman, and R. Yon, "Music personalization at spotify," 2016.

[11] Accenture, "Younique: Accenture personalized. marketing index," 2017. [Online]. Available: https://www.accenture.com/t20170922T071216Z__w__/cz-en/_acnmedia/PDF-60/Accenture-Interactive-Personalized-Marketing-Index-Infographic.pdf

[12] "Monetate Study Finds Brands Experiencing Highest Return on Personalization are Twice as Likely to Cite Customer Lifetime Value as Primary Goal," 2019. [Online]. Available: https://www.businesswire.com/news/home/20190313005156/en/Monetate-Study-Finds-Brands-Experiencing-Highest-Return-on-Personalization-are-Twice-as-Likely-to-Cit

[13] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," 2003.

[14] Y. Allen, "How Netflix Uses AI and Machine Learning." [Online]. Available: https://becominghuman.ai/how-netflix-uses-ai-and-machine-learning-a087614630fe

[15] J. Díez, P. Pérez-Núñez, O. Luaces, B. Remeseiro, and A. Bahamonde, "Towards explainable personalized recommendations by learning from users' photos," *Inf. Sci.*, vol. 520, pp. 416–430, 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2020.02.018

[16] V. Gurusamy and S. Kannan, "Preprocessing techniques for text mining," 10 2014.

[17] J. Camacho-Collados and M. T. Pilehvar, "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis," 2018.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 4278–4284.

[19] A. Alemi, "Improving inception and image classification in tensorflow," 2016. [Online]. Available: https://ai.googleblog.com/2016/08/improving-inception-and-image.html

[20] J. Brownlee, "A Gentle Introduction to Cross-Entropy for Machine Learning," Oct. 2019. [Online]. Available: https://machinelearningmastery.com/cross-entropy-for-machine-learning/

[21] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser.

ICML '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 78. [Online]. Available: https://doi.org/10.1145/1015330.1015435

[22] L. Prechelt, "Early stopping - but when?" *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, p. 55 - 69, 1998.

[23] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, p. 5 - 53, Jan. 2004. [Online]. Available: https://doi.org/10.1145/963770.963772