POLITECNICO DI TORINO

Master's Degree in COMPUTER ENGINEERING

Master's Degree Thesis

Heterogeneous data driven recommendation systems for books in libraries

Supervisors

Dr. Luca VASSIO

Dr. Greta VALLERO

Prof. Marco MELLIA

Candidate

Alessandro SPECIALE

DECEMBER 2022

Summary

In an era where digital progress runs fast, it is imperative to find ways to innovate public infrastructures through technology in order to help them improving the quality of their services while maintaining the characteristics that make them good. This thesis focuses on finding a way to extract information from data supplied by "Biblioteche Civiche Torinesi (BCT)" to implement a recommender system capable of suggesting a book tailored to the preference of the reader. Moreover, these data are used in conjunction with a VR environment designed by "VR Polito Team" to create a digital space where readers could taste the feeling of a library even when they are at home. This is done in order to make libraries more appealing to the general public that has grown used to the appeal of the digital world and to invigorate the readers' community by offering them services that can improve the reading experience.

For the thesis, I proceeded following three main phases.
The first was the data characterization: my main objective was to define, quantify and preprocess BCT data in order to make it suitable to being used as input for a recommender system. This phase was also heavily based on the use of Anobii (a book-based social network) to augment both the quantity of data and of information per book at our disposal. Using Anobii data was crucial in finding reading patterns associated with different readers due to the richer quantity of information related to books with respect to BCT. In particular, we analyzed a dataset of approximately 100000 books although, with the necessary preprocessing filters (italian language, removal of periodicals and a threshold for books with a minimum number of ratings) that number dropped to 3000-8000 depending on the value of the threshold used. The size of each book entity was quite variable because it depended much on the length of the Anobii book's description: to maintain meaningful infos we added another filter to have descriptions no shorter than 50 characters.
The second phase focused on finding ways to further increase the quantity of information for each books by inferring them from already possessed info. This was done in order to make the VR environment more interactive and playful by allowing the user to input characteristics he would like to see in a book to see the result but also to try to use them as inputs for the recommender models.

Following this thread, I used Anobii's informations regarding the books' descriptions to extract a mood (anger, sadness, joy or fear) and a sentiment (positive or negative) related to the books using natural language processing approaches. To do so, I passed the descriptions of each book that, by using a neural network trained on social network posts, gave an output. The set of extracted keywords for each book could be used as a new feature for the recommender model.
Then, I went on by implementing an algorithm to extract, for each book genre we had at disposal in our dataset, a set of limited and as meaningful as possible keywords representing the genre. This was done in order to make the VR environment more immersive by allowing the users to use keywords as a filter to find new related books.
Finally, the third phase was the implementation and evaluation of the recommendation system. During this phase, we applied various recommender models and fine-tuned them to have a good performance for the problem at hand. One of the major problems was how to use the explicit ratings of Anobii with the implicit book-reader interaction of BCT. The answer was to build implicit feedback models. Content based and collaborative filtering were used in the thesis.
The main idea of content based models is that users will read books that are similar to books they have already read: as a consequence, the suggestions tend to not be serendipitous but they are also more explainable, allowing the user to discover why a certain book was suggested to him in a punctual manner.
Collaborative based models are recommenders that suggest item to a user by considering preferences of similar users assuming that 'similar people like similar things'. This lead to define groups of 'user stereotypes' on which a user tends to fall and cause the non-explainability of suggestions but also their serendipity and overall quality.
In particular, the implemented model were:

- Item-item similarity - content based model where the most similar book to one the user has already read is suggested.

- Bayesian Personalized Ranking (BPR) model - Collaborative model based on the BPR algorithm and useful in our use case due to its native adaptability to implicit feedbacks and good performances

- Alternating Least Squares (ALS) model - Collaborative model based on the ALS algorithm. BPR is often considered better suited to implicit feedback environments and this model is thus used to assess the quality of the BPR.

- Tensorflow-recommender retrieval model - collaborative implicit model offered by the tensorflow-recommender library, it has been considered due to its efficiency with large datasets and the possibility to extend it with more layers.

The models were trained with the 64% of the dataset created during the first two phases (while 16% was used for validation and 20% for test) and the main metrics used to evaluate the models were:

- Average hit users - ratio between hit users (users that received a good recommendation) and the total number of users. It is representative of how many users are surely satisfied by the recommender.
- Average hit recommendations - ratio between hit recommendations (good recommendations) and the total number of users. It represents how many good recommendations were given for each user.
- Average rank: ratio between the sum of minimum ranks in the recommender suggestion list and the total number of users. It is used to determine how condensed good recommendations are in the suggestion ranking list.

The best model was the BPR based one, giving more than 20% on the average hit users metric: the fact that collaborative models worked the best means that the readers tend to follow certain reading patterns and that the recommender is able to detect them in a performing way.
However, the content based model, although less performing, managed to give explainable recommendations to the reader: this means that the reader may know exactly why a book was recommended to him in terms of his reading history and this may allow a richer overall experience.

Acknowledgements

To my family, to my friends and to my grandma who watches from Above.

Table of Contents

List of Tables

List of Figures

Acronyms

BCT
Biblioteche civiche torinesi / Turin public libraries

AI
Artificial intelligence

VR
Virtual reality

TF-IDF
Term frequency - Inverse document frequency

NLP
Natural language processing

BPR
Bayesian personalized ranking

CDF
Cumulative distribution function

ICT
Information and Communications Technology

ALS
Alternating Least Squares

Chapter 1

Introduction

1.1 ICT in today's world

Technology is everywhere: from programmable washing machines to infotainment on cars, its influence on society is evident and firm. Thanks to this, in a world where technology is improving steadily and quickly, both public infrastructures and companies have started to improve the quality of their services using technological innovations as a support. However, this also meant that many entities were pressured in finding a way to integrate these innovations into their services not only as a way to improve but also to be up to date and keep on track the attention of the general public.
This thesis focuses on the concept of "recommendation" as a way to improve the services of the complex of public libraries in Turin by building recommender models able to output book suggestions. In particular, the work aims at finding a solution to integrate technology with the libraries environment in order to offer them innovative services and expand the quality of their reading. By doing so, our desire is to expand the community of readers of Turin libraries by putting to good use its potential to become an hub of culture of people passionate about reading.

1.2 The problem of choice

The outstanding pace at which technology is evolving allowed people have now the power to nurture their passions by finding whatever they like on the Internet. This is, of course, beneficial because it allows each person to fully express their hobbies by discovering new things related to their field of interest. However, this comes with a price: the enormous quantity of possibilities a user can face while searching may lead to the so called "choice overload" [1]. Choice overload is a psychological concept where a person may actually find lack of motivation in having too many

possibilities due to the burden of responsibility of choosing a good candidate. This applies fully to a digital world where possibilities are almost infinite.
To understand how this problem may apply to our case, let's see some statistic: from [2] we can see that there are approximately 130 million unique books in the world. Moreover, 500000 to 1000000 new books are published each year. This means that, for a reader, is not so easy to find new books which may interest him without incurring on the "choice overload" problem. A recommender would allow that user to consult a limited set of books and choose from it, helping him in a choice that is ultimately his. This means that a recommender not only unburdens people from a difficult choice but also makes them more active and satisfied.

1.3 The power of recommendation

The concept of "recommendation model" in digital services exists to allow the user to choose something they like from a limited set of products that is tailored to their preferences. By doing this, the burden of the choice is lifted from the consumer which, in turn, is more motivated to opt for a certain product. Empirical demonstrations of the utility of recommendation systems are not so rare: as noted by the article in [3], it is an established fact that 75% of Netflix's watches comes from recommendations and so are 60% of Youtube's video clicks and, regarding e-commerce, approximately 35% of Amazon sales. This means that a good recommender model is useful, even if not fundamental, to form a community of users able to decide freely. Moreover, recommenders are important because they allow infrastructures to offer better services to their users, thus increasing the activity of their community. Recommendation models are usually divided into three different main types:

- "Content" models use features of items that the user interacted with (in the case of this thesis, features of books read by the user) to output suggestions based on those.
  In a more practical case, content based recommenders are used when we want the suggestion to depend only on the taste of the user like a Netflix recommendation formed by the user's watch history.

- "Collaborative" models find users similar to the current one in terms of interactions and output suggestions based on those. Those recommendations depend not necessarily on the taste of the user but on the taste of users that are similar in terms of tastes to the current user. This is done by supposing that similar people like similar things. Amazon's market basket analysis based on the "Users that have bought this also liked ..." paradigm is a notable example of a collaborative recommender.

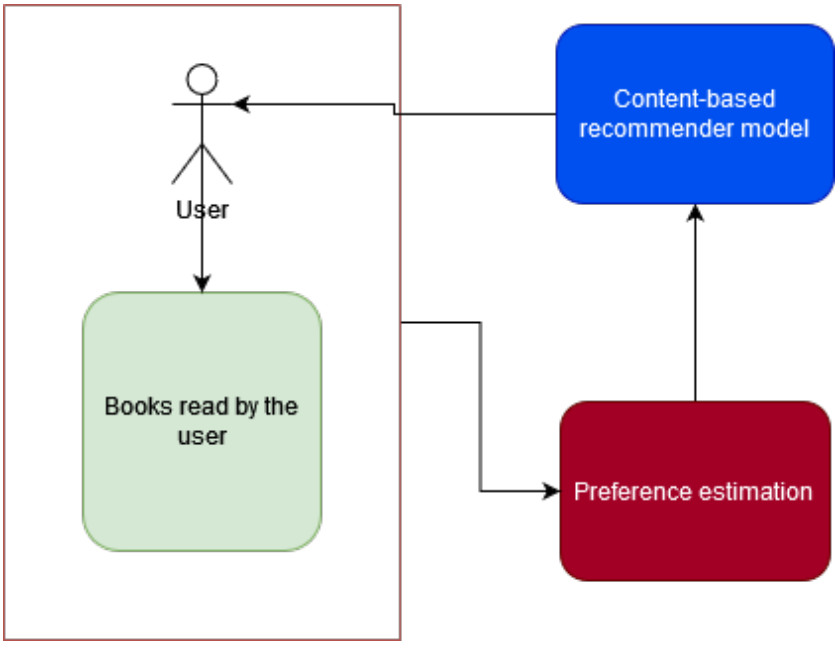

Figure 1.1: Minimal diagram of a content-based model.

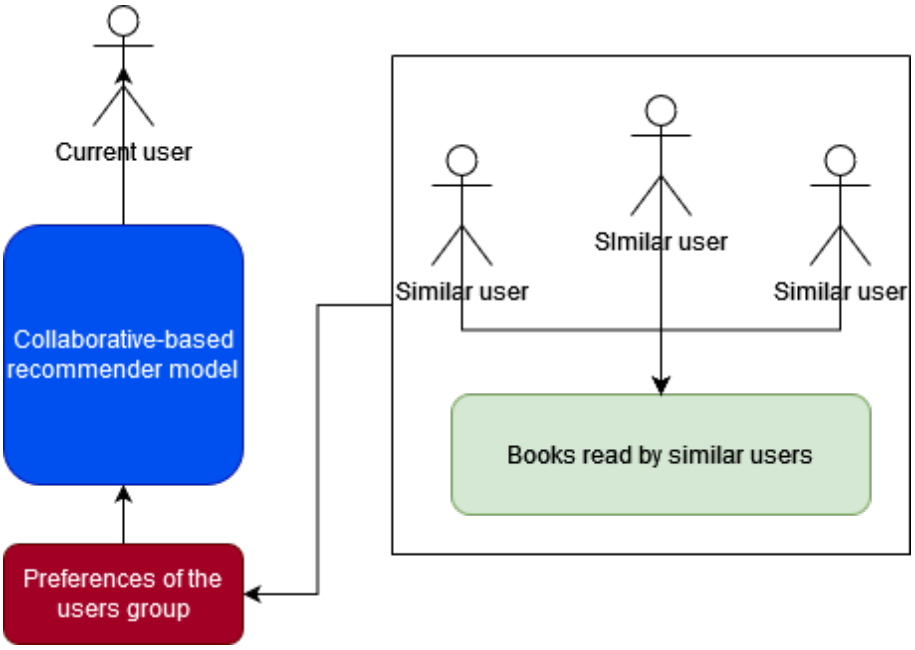

Figure 1.2: Minimal diagram of a collaborative-based model.

- "Hybrid" models output a suggestion by combining the results of the first two types. The combination is done on a way that considers both the user's tastes

and the similar user's tastes trying to combine the advantages of both models in the process.

1.4 Reading&Machine

From what has been explained so far, it is clear that helping a community at discovering new things to nurture their hobby would make their users more active and satisfied. In the world of public libraries, there are not many notable examples of recommender models capable of suggesting books from the catalogue in an automatic way. This innovation, that would bring advantages both from a visibility and an efficiency point of view, is the objective of the project "Reading&Machine" on which this thesis is based on. The project "Reading&Machine" aims at building a data driven system where information related to books is used to model an environment where readers of the public libraries in Turin (BCT) can obtain book suggestions tailored to their needs and preferences. To enrich the quantity and quality of data, other than the BCT database, data provided by Anobii - a reading based social network - were also used. Although the thesis focuses on data science, the overall project also includes a VR section where a virtual reality ambient is shaped as a library, thus giving to the readers the possibility of interacting with a real library and its community even when far away. By joining these two aspects of the project, the objective is to build an environment where BCT users can feel at home and be able to express their passion for reading by having a modern recommendation service to guide them and lift the "choice overload" problem. By following this objective, the project wants to exploit technology as a way to nurture the libraries community, to improve the quality of the reading and to stimulate the participation of a wider audience into the libraries world. My work on the thesis can be distinguished into three main phases.

- The first phase is a data characterization one, where data of BCT and Anobii is defined, quantified and merged. Anobii's data was fundamental in creating a database of books rich of useful features due to the vast quantity of information they offered. Moreover, the presence of a rating dataset - in which Anobii's users gave a vote to books they read - lead to the possibility of implementing a recommendation model based on collaborative filtering.

- The second phase consists on inferring new features from already existing ones. This is important because it allowed the use of new info like keywords and sentiments as input in the recommendation model. To perform this task, studying about specific information extraction techniques of keyword extraction and natural language processing was needed.

- The third phase is the recommendation one where various models are built, tested and tuned for books suggestions. This phase was the most challenging due to the vast world of machine learning and recommender models and the vast quantity of papers focusing on this argument. The flexibility and richness of the recommendations was an important factor to consider due to its application on a real environment and so it was important to think on ways to improve these characteristics.

Chapter 2

Related works

2.1 Inspirations and notable examples of reading and ICT

2.1.1 The "Oodi" library

The purpose of the "Reading&Machine" project is to modernize Turin libraries by integrating VR and recommender models in them in order to offer new and efficient services to please the community of readers. This objective was inspired by "Oodi", a public library situated in Helsinki, Finland. "Oodi" is an extraordinary example of classic public infrastructure improved by cleverly thought architectural designs and technological integrations. In fact, other than the functional and sustainable architecture it presents in [4], it also possesses an AI bot (called Obotti) used to offer book suggestions to users based on their preferences as described in [5]. Obotti is also characterized by a predefined set of "personalities" focused on suggesting books of a certain genre. This was inspiring when finding ways to increase the richness and the customizability of the suggestions.
By using the conjunction of modern designs, family friendly zones and recommendations usage, it managed to build an environment where, through technology, users may enjoy innovative and efficient services and became a cornerstone for all libraries that wish to do the same.

Figure 2.1: Eagle view of the innovative Oodi library, Source: Oodi.com

2.1.2 Anobii, a reading based social network

Anobii [6] is a famous social network for people passionate about reading. Thanks to its contribution on the "Reading&Machine" project, it was possible to use their data related to books and ratings in order to train more efficient models. In fact, although BCT possesses a vast catalogue of book titles, certain features present only on the Anobii database allowed the creation of inferred features and also more flexibility and richness of the quality of the recommendations. The existence of Anobii is a bright example of how communities of avid readers are prone to use technology as a way to find new books they may like: in [7], Anobii's tools for browsing new and possibly interesting books were evaluated. From its result, we can see that the various services Anobii offers for books searching are appreciated by users. In particular, social recommendations (books suggested from friends bookshelves) and feature based recommendation (books suggested by using certain features, like the author) fared pretty well. This is indicative of how readers are open to use recommender models to discover new books. This compatibility was one of the basis of the project.

2.2 Feature augmentation

2.2.1 Keywords extraction

During the data characterization phase, an extraction of meaningful information from books descriptions was needed. Intuitively, keywords extraction seemed like

a good way to summarize a long and unstructured text using a limited set of meaningful words.
In [8], the author applied a series of keywords extraction techniques to a collection of documents in order to prove the efficiency of each algorithm. In particular, TF-IDF was applied as a baseline for scoring words. An example of this can also be found in [9] where, although TF-IDF is not the best algorithm, is considered a reliable enough scoring system.

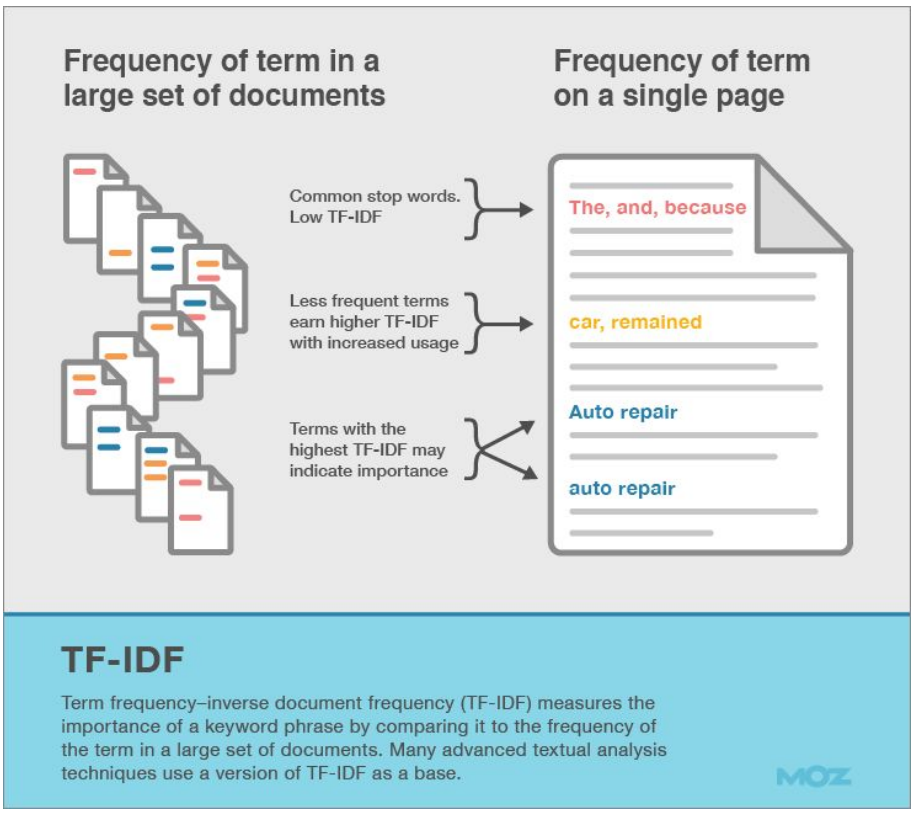

Figure 2.2: Graphic explanation of TF-IDF, Source: Moz.com

TF-IDF is an algorithm used to rank a set of words in a text of a collection of texts by considering both their occurrences in the text and their inverse frequency in the overall collection. As we can see from the image, the TF-IDF score for a given word in a text is strongly dependent on its occurrence in the considered text and its rarity on the other texts in the collection. By doing so, the top scored words should be the not only important for the text but also meaningful due to its rarity on other texts. The common use of TF-IDF as a relatively reliable and easy to apply algorithm for word weighting in a set of documents is what convinced me to apply it to extract information from books descriptions: in fact, although it is

not the best, it fits well in an unlabeled dataset where supervised models are not usable and the main objective is to simply find frequent yet meaningful words.
It is also worth to mention that [10] offers am overview about the importance of a "stemmer" and a "lemmatizer", processing layers that extract the stem part of a word (in the case of the stemmer) or its base form (in the case of the lemmatizer) in order to calculate the score without being biased in considering the same words as different due to a difference in gender or cardinality. As shown in the paper, these layers use NLP models trained on a certain language in order to give the stemmed or lemmatized output. Thus, to increase the efficiency of the keywords extractor, a stemmer and a lemmatizer were applied.

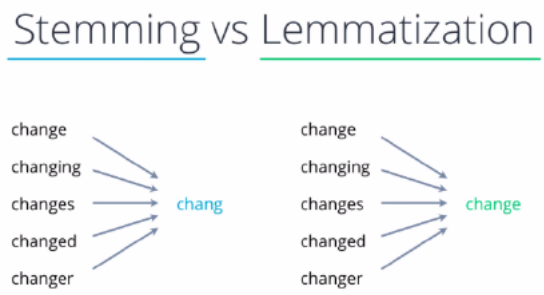

Figure 2.3: Diagram representation of stemmer vs lemmatizer, Source: [10]

2.2.2 Sentiment extraction

To enrich the quantity of features at our disposal, I continued to search for ways to extract new features from already existing ones. [11] offers a good overview of the current state of sentiment analysis and talks about the potential of using deep learning to perform this task. From the point of view of a reader of BCT, it can be useful to know both the feelings expressed and the overall tone of the visualized book; moreover, having those features as input in the recommendation phase can possibly further differentiate group of books. So, although sentiment analysis is usually performed on relatively short texts (like Twitter posts), I was curious to perform it on the collection of books descriptions. In particular, the paper talked about different kinds of sentiment analysis:

- Lexicon based - this approach is based on the use of a dictionary of words labeled with the feelings associated with them. A good example of this approach is explained in [12]. However, this method is fairly dependent on

the quality of the dictionary file. This problem may be further amplified by the need for the dictionary to use the Italian language.

- Machine learning based - this approach is based on the use of shallow or deep models to classify texts with a sentiment label. The models require training with an extensive dataset.

For the task at hand, I decided to apply both of the approaches to compare the results. In particular, the model explained in [13] gave state of the art results in terms of accuracy on unstructured Italian texts and using its pre-trained instance allowed me to overcome the problem of the lack of sentiment data.

2.3 Recommendation systems

Due to their importance and relevance, there are many papers talking about recommendation models and ways to tune them in order to reach good satisfying results. As explained in 1.3, recommender models in literature are often divided in three main branches: collaborative, content-based and hybrid. With respect to our use case, where we have both explicit feedbacks in the form of Anobii ratings and implicit feedbacks as loans, it was necessary to assess the efficiency of these types of recommender models with both.

2.3.1 Content-based models

Content-based models are recommenders that focus on suggesting items to the user basing them only on the past history of said user.
[14] provides a good overview on content based recommender models. As explained in the paper, these models work by comparing a representation of the user's preferences with the item set in order to calculate a similarity score and output a ranking list of items based on said user. In an explicit feedback context, the method to assess the user's preference is usually a classification problem where the model tries to give a score to a feature based on the ratings the user gave to a small number of items. In this context, ratings are fairly important and may easily overfit the model if given without a pattern. However, as the paper suggests, content-based models may work well also with implicit feedbacks, but with a trade-off: given that we are assuming that an interaction is positive, the model may actually not be efficient if that assumption is proven empirically wrong. Content-based they can provide more explainable recommendations to the user with respect to collaborative models (that instead represents preferences by using patterns of a larger group of user, creating clusters of "paradigms") due to the

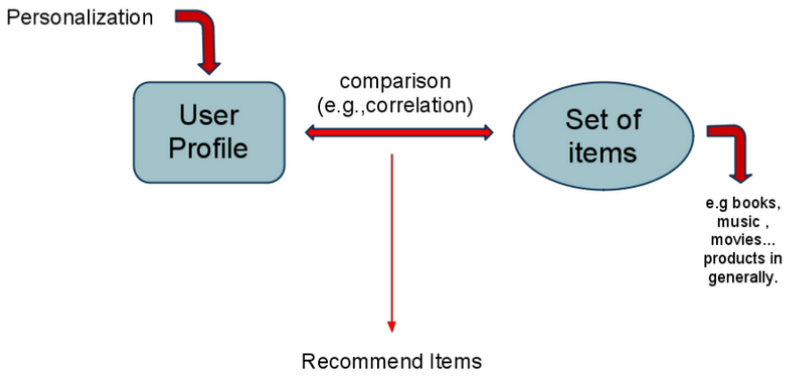

Figure 2.4: Content-based model diagram, Source: [14]

fact that the suggestion is based entirely on its own interactions. However, those recommendations are also less serendipitous and depend largely on the "goodness" on the interactions, both in an explicit and an implicit context.

2.3.2 Collaborative models

Collaborative models are based on the technique of "collaborative filtering". They output suggestions based on similar users using the assumption that similar people like similar things. Collaborative filtering is usually divided into two main branches: neighborhood models and latent factor models. One of the earliest and practical example of neighborhood models is contained in [15] where an overview is provided. From the paper, we can see that neighborhood models are based on finding a similarity measure (the paper uses both the Spearman and the Pearson correlation coefficient) between the current user and the other users and then output suggestions based on the preferences of the top k subset of users. Latent factor models, instead, as explained in [16], focus on represent both users and items as comparable latent vector projected into a n-dimensional space where, through a factorization matrix computed by multiplying these two vectors, scores are computed.

This paper also gives a demonstration of the usability of collaborative models in an implicit feedback context. In fact, although unrated feedbacks formed a bias caused by the absence of negative interactions, the model gave satisfying results. [17] shows the potential of a collaborative filtering model applied on a real life application.

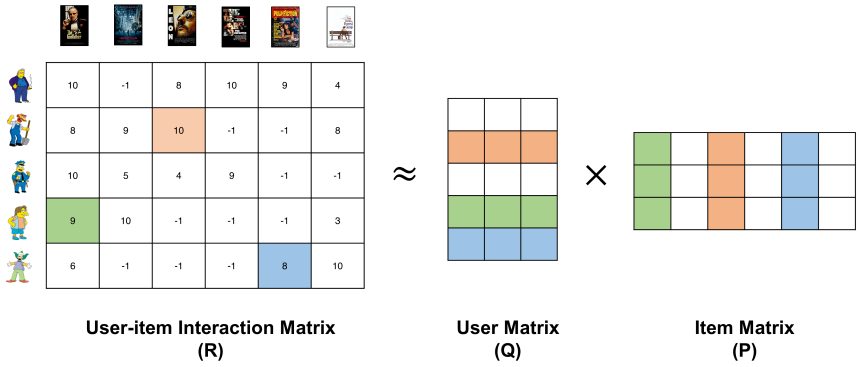

Figure 2.5: Visual representation of a factorization matrix, Source: nvidia-merlin.github.io

As noted in the results section, collaborative models showed great flexibility on their ability to output different suggestions for different group of users while also giving fair results with explicit ratings. In fact, by giving different user features to the model during the computation of the latent vector, it was possible to make richer recommendations that account also for different user characteristics and not only for similar users. This is important because it gives us a way to use the available features of both users and books in an efficient way.

2.3.3 Advanced recommender models

In our use case, we have to deal with a dataset containing sparse user-item interactions. As explained in [18], there was the need to optimize an algorithm to output directly a ranked list of suggestions while also dealing with sparse interactions, something that could penalize matrix factorization scores as calculated in 2.3.2. The BPR algorithm discussed in the paper managed to set a new state of the art for recommendation models by being capable to give better results than other algorithms while also giving directly a ranked list for each user. To work, the algorithm tries not to predict the rating a user would give to an item but instead predict the relative preference of a user with respect to two items. The following image explains that BPR tries to create a set of matrices representing the relative preferences. To optimize the personalized ranking, the stochastic algorithm explained in the paper is used until convergence. The BPR not only dealt with the problem of missing user-item interactions but it also worked well in implicit

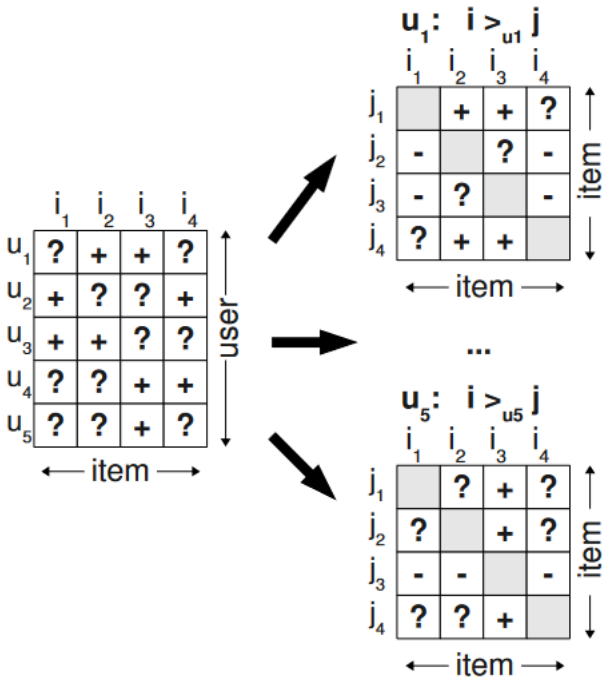

Figure 2.6: BPR relative preferences, Source: [18]

feedback environments, making it apt to being used for our use case.
After reading [19], I also started to think of the integration possibility of a recommender model in a deep learning environment thanks to its clever use of neural networks to calculate ranking scores. The neural collaborative filtering, as called in the paper, is based on the usage of a set of neural layers instead of the traditional matrix factorization. This is explained by noting that the traditional matrix factorization didn't manage to capture the underlying relations between users and items with a low number of latent factors. However, neural layers were more apt to do it as showed on the final results section where the neural collaborative filtering fared way better than the traditional one, thus opening at the possibility of deep learning into the recommendations world.

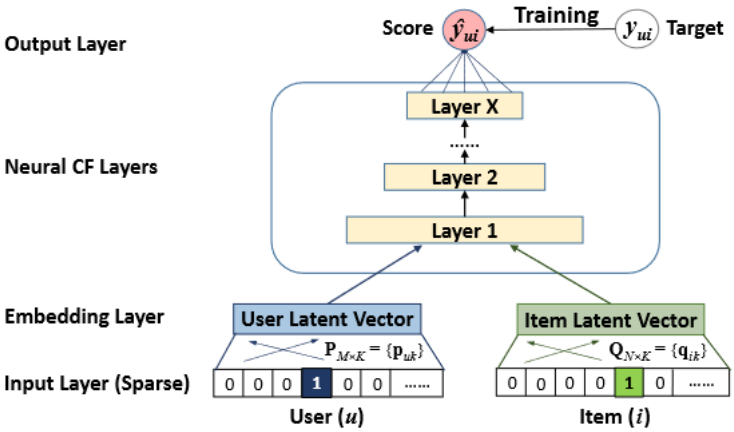

Figure 2.7: Neural collaborative filtering diagram, Source: [19]

Chapter 3

Data characterization and processing

3.1 BCT data

To start the practical work of the thesis we needed to look at the data provided to us by BCT. Our objective during the processing of the BCT dataset was to group enough books to have between 2000 to 3000 books at the end of the merging phase with Anobii data. The filtered set of books must be composed of narrative books for the general purpose reader, so our processing is focused on that.

3.1.1 BCT Characterization

The set of files was in CSV format and contained heterogeneous information about the books and the libraries. In particular, they were:

- item media type - By using an id code, declare what type of objects exist in the libraries database. The id can assume 17 different values:

  A: Audio recordings

  B: Manuscripts

  C: Graphics

  D: Cartography materials

  E: Microforms

  F: Monographies

  G: Multimedia

  H: Music sheets

L: Physical objects

M: Visual projections

N: Electronic resources

O: Braille texts

P: Signed letters

Q: DVD

R: VHS

S: Periodicals

T: Audiobooks

- items - list of every object present in the libraries
- libraries - list of each Turin library provided with a numeric id
- patrons - list of every users of Turin libraries identified by a MD5 hash id
- manifestations - list of all unique objects present in the libraries
- loans - list of all user-book interactions occurred from the 2nd of January 2012 to the 24th of January 2021

A list of the attributes of all the BCT data files is shown in tables 3.1 and 3.2.
As we can see, although we possess a good number of features to identify users, books and loans, we do not have informations that would make the book dataset more rich like a content description or an associated genre.
Let's also note how important features like the title of the books are contained into the manifestations file.
I proceeded to quantify the data at my disposal. Figure 3.1 contains the CDF of loans per user. From that, we can see that the vast majority of the users has less than 100 loans.

We can also visualize the number of occurrences of each item type in the dataset in 3.2: from said figure, it is evident that monografies cover the vast majority of the dataset.
Going on, the dataset contains:

- 163321 users of which 98587 are females and 63885 are males.
  In graph 3.3, we can see the year of subscription of the registered users; 2012 was the year with most subscribers.

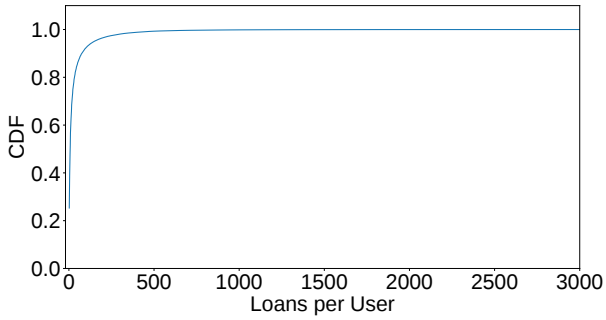

Figure 3.1: CDF of BCT loans per user

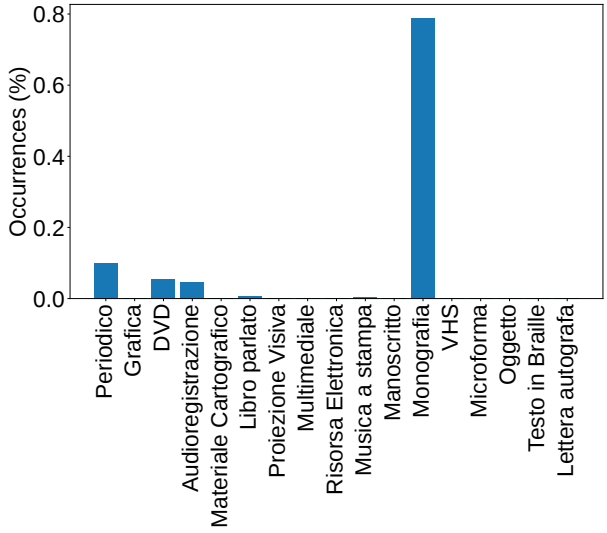

Figure 3.2: Occurrences in % of each object type in BCT data

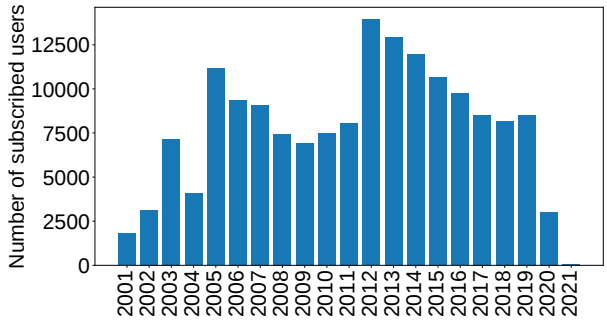

Figure 3.3: BCT subscription in each year from 2001 to 2021.

- 998403 items referring to 290125 manifestations. In particular, the books refer

BCT files - Part 1		
File name	Attributes list	Disk space
Item media	• Item media ID: unique identifier of the media type • Item media: name of the media type	111 MB
Items	• Item ID: unique identifier of the physical book. • Manifestation ID: not unique identifier of the book. affiliation. The manifestation can be present in more than one physical book. • Item media ID: identifier of the media type related to the book. • Home library ID: identifier of the residing library. • Inventory date: date in which the book was added into the catalogue.	30 MB
Patrons	• ID MD5: unique identifier of the user, composed by a MD5 string. • Registration ID: identifier of the user registration process. • Preferred library: information about the favourite library of the user. • Gender: Sex of the user. • Civil Status: Status of the user. • Creation date: registration date of the user. • Last seen: timestamp indicating the last moment of activity of the user in the BCT site.	14 MB

Table 3.1: BCT files and attributes - Part 1

to 97 possible languages. However, in some cases, the formatting was different for the same language. For example, the Italian language was associated both to the 'ita' and 'ITA' label so it was necessary to consider both labels during the language filter. We can have a look at the top 5 languages (for the sake of visualization) of the unique books in 3.4. As we could expect, the Italian language composes the almost totality of the books dataset.

BCT files - Part 2		
File name	Attributes list	Disk space
Libraries	• Library ID: unique identifier of the physical library. • Library name: name of the associated library.	1 MB
Loans	• Patron ID: identifier of the interacting user. • Item ID: identifier of the borrowed book. • begin loan date: starting date of the loan. • end loan date: ending date of the loan (if present). • due date: deadline for the loan. • from library: library where the loan has been initiated. • end library: library where the loan has ended (if present).	593 MB
Manif.	• Manifestation ID: identifier of a version of a book. • Language • Edition date: year of the edition's publication. • ISBN: ISBN code of the book. • EAN: EAN code of the book. • Title: name of the book. Includes the subtitle. • Author: author of the book with a "Surname, Name" format. • Publisher: publisher of the book. • Loanable since: year in which the book started to be loanable. • Creation date: Date in which the book was added to the catalogue.	48 MB

Table 3.2: BCT files and attributes - Part 2

- 49 libraries. Looking at the top 10 of books per libraries in 3.5, as expected, we note that the central library contains the majority of the books.
- An average number of loans per user of 34.
- 53004 different authors.

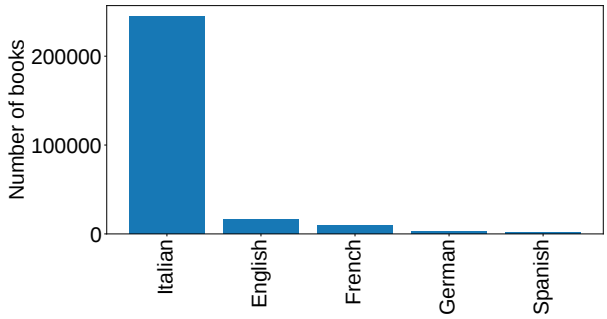

Figure 3.4: Number of books with respect to the top 5 languages

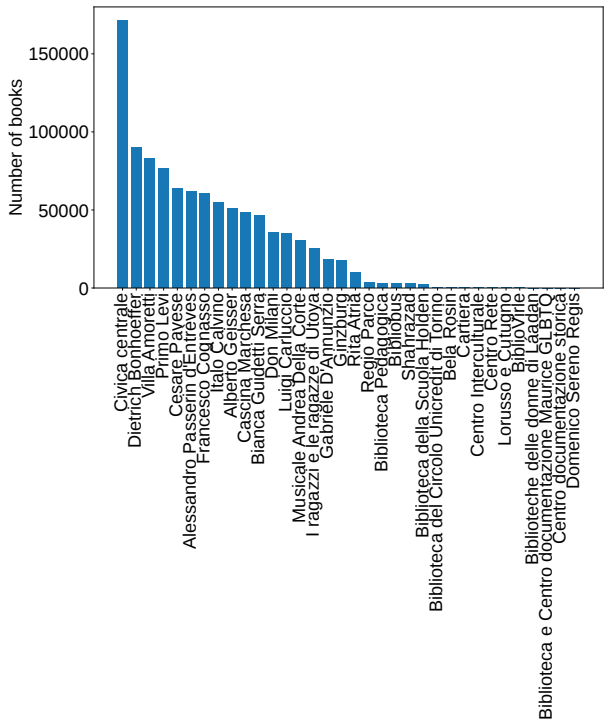

Figure 3.5: Number of books per top 10 libraries

3.1.2 BCT Filtering and processing

Now that we have characterized the BCT data, it is time to preprocess it to make it better apt at being merged with Anobii data and being used as input of the recommender system.
Our objective was to make a recommender able to output books for the general consumer, so it was imperative to use a set of filters to discard books not needed for this objective. The set of applied filters were the following, applied in a sequential

way:

- Removal of items which are not books. The objective of the recommender models is to suggest traditional books to the general public of the libraries. So this filter is used to select a subset of the dataset containing only monographies (which, as seen in 3.2, form the vast majority of the data) and manuscripts.

- Removal of not-Italian books (which form the minority of the data, as we can see in 3.4) from the books dataset. This is done due to the objective of the project to build a recommender model trained on Italian books. To do so, we used the 'edition language' attribute in the items file.

- Removal of periodical books. The dataset contains books that are numbered volumes of a series. To keep the dataset from containing too many volumes referring to the same work, we filtered away famous Italian periodical comics in order to prioritize traditional books.

	manifestation_id	title	publisher
0	405844	Topolino salva il Natale : e altre topostorie	Buena Vista home entertainment
1	271281	Topolino : strepitoso Natale!	Walt Disney Home Entertainment
2	306743	Topolino e i pirati / Walt Disney	Mondadori
3	68838	Topolino e le forme geometriche	A. Mondadori
4	306733	Topolino boxeur / Walt Disney	Mondadori

Figure 3.6: Examples of "Topolino" periodicals in the dataset

- Cleaning of the 'author' field. The authors in the dataset were formatted using the pattern "Surname, Name". To format them in a way that allowed the merging with the Anobii dataset, we transformed this formatting in a more traditional "Name Surname" pattern (the same of Anobii) and we deleted various rumor characters.

- Removal of one word titled books. Considering the need to suggest traditional narrative books to the readers, there was the need to delete monographies books which still did not respect the requirements. Many books in the dataset that contained only one word were deleted due to the fact that they were mostly turistic guides.
  Of course, this also means that we are considering the loss of narrative books with only one word. This was accepted due to the vast quantity of books still at our disposal with respect to the objective explained in 3.1.

	manifestation_id	title	publisher
1	175289	Kenia	Nelles
2	428870	Jérusalem	Gallimard
3	435319	Chicago	Nouvelles éditions de l'Université
4	420325	Amsterdam	Rizzoli

Figure 3.7: Examples of turistic guides in the dataset

- Aggregation of manifestation IDs. From the subsection 3.1.1, we have seen that the title of the books are contained in the manifestations file. However, manifestation ids are different if the edition is different, even for the same book. This means that we had to aggregate the manifestation id of the books with the same title but with different edition in order to have one id per book and to not suggest the same book as two separate entities.

	manifestation_id	title	publisher
0	443600	Il mio credo / Hermann Hesse ; a cura di Maria...	Milano Rizzoli
1	164707	Il mio credo / Herman Hesse ; a cura di Maria ...	Biblioteca universale Rizzoli
2	52752	Il mio credo / Hermann Hesse ; a cura di Maria...	Rizzoli
3	224488	Il mio credo / Hermann Hesse ; a cura di Maria...	Biblioteca universale Rizzoli

Figure 3.8: Example of multiple manifestation ids associated with the same book

- Removal of items from the dataset that have been loaned less than an arbitrary chosen threshold in order to maintain only books with a good number of interactions. The following graph shows how the number of books varies depending on the chosen threshold.

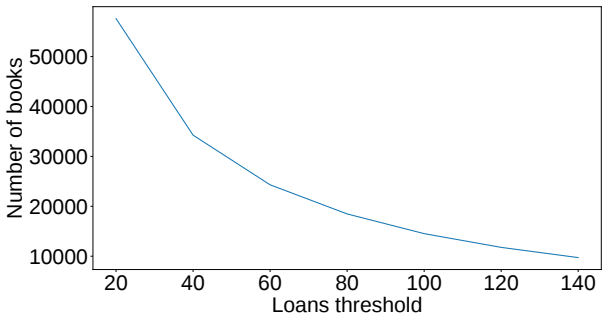

Figure 3.9: Number of BCT books depending on the loans threshold

To retrieve our data, we couldn't decide the threshold yet. In fact, to respect the objective explained in 3.1, we must decide the threshold only after seeing its impact on the merging phase. So, this choice is postoponed on 3.26.

To sum it up, the following diagram shows all the operations applied on BCT data.

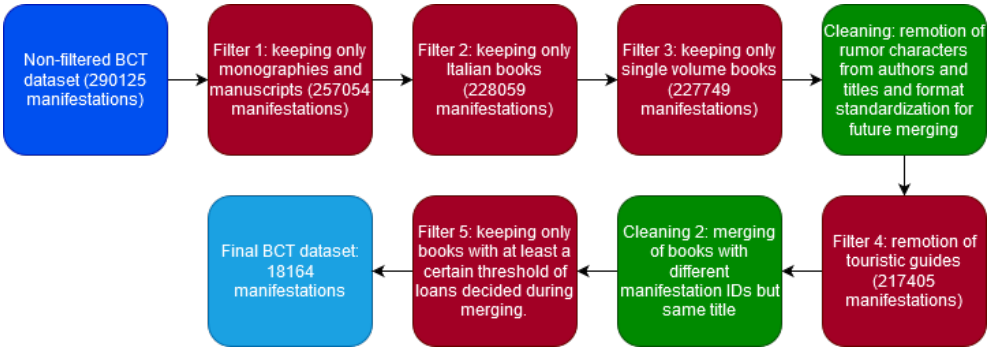

Figure 3.10: Data filtering and cleaning on BCT data

3.2 Anobii data

Let's now have a look to the data provided to us by Anobii. Our objective, in this case, was to merge Anobii data with the BCT one in order to integrate the richer Anobii features into the libraries data. In doing so, we aim to build a well populated dataset of books containing the maximum amount possible of integrated features. As with BCT, even here our objective is to form a limited dataset of approximately 2000 to 3000 traditional books with rich features during merging, so our choices will be focused on that.

3.2.1 Anobii Characterization

The Anobii dataset is composed by a set of CSV files containing books, users, authors and explicit ratings. The files are:

- author display - list of authors identified by an id.
- author items - list of the books a certain author has written.
- cities - list of the cities associated with users data.
- countries - list of the countries associated with users data.
- items - list of the books and their features.

- persons - list of the users with their features.
- item comment feedback - list of the comments the users gave to a book.
- language mapping - list of all available languages in Anobii.
- link person item - list of all the users-books interactions with a vote.
- link person item comment - list of the comments associated with a certain rating.
- item comment vote - list of the comments associated with a rating by other users.
- link person person - list of the users following a certain person.
- link person recommended item - list of books recommended by a user to another user.
- link person recommended message - list of the messages associated with a recommendation.

As we can see, Anobii's data is richer than the BCT one: a bright example of this lies in the presence of explicit ratings of the user (item review) that enable us to extract with more accuracy preferences and reading patterns of the users. This gives us a way to model user preferences with less bias.
We can also notice that, while in the BCT dataset title and subtitle were embedded into the same 'title' field, in the Anobii dataset they are represented by two separate fields.
Going on, the following image shows us the CDF of the ratings per user. As we can note, almost the totality of the users possess less than 100 ratings, making the set of user-item interactions quite sparse, as we could expect.

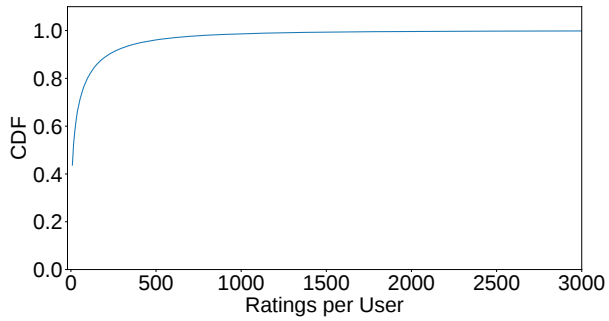

Figure 3.11: CDF of the number of ratings per user

The following list shows info to better characterize the quantity of data:

- 1202909 users. In 3.12 we can see the number of users of each state in the top 10 of Anobii. The 'active users' bar indicates the part of active community, calculated through the 'Active User' flag in the person file.

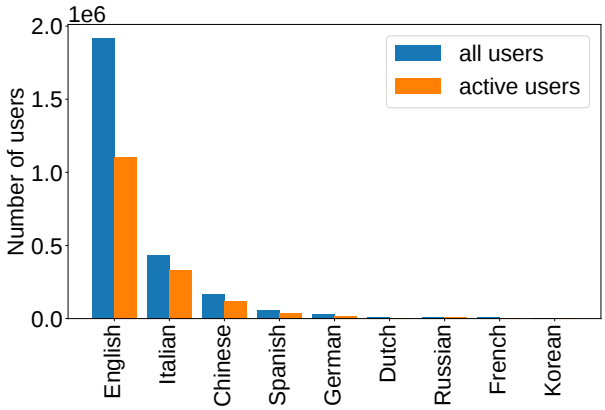

Figure 3.12: Number of Anobii users in the top 10 countries.

- 8021517 books of which 3373430 rated. In particular, as we can see from figure 3.13, the vast majority of users tend to either not rate a book or to rate it positively. Moreover, 3.14 tells us the number of books labeled by the top 10 languages.
  Judging from this figure and 3.12 we can conclude that it was a good choice to use Anobii data for a recommender based on Italian suggestions because the social network possesses a solid Italian community with a very good amount of Italian books.

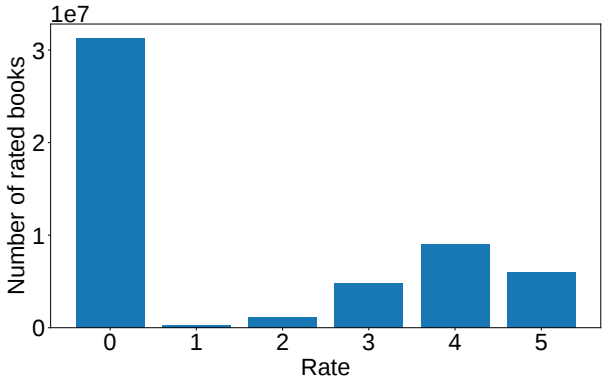

Figure 3.13: Occurrences of ratings in Anobii database

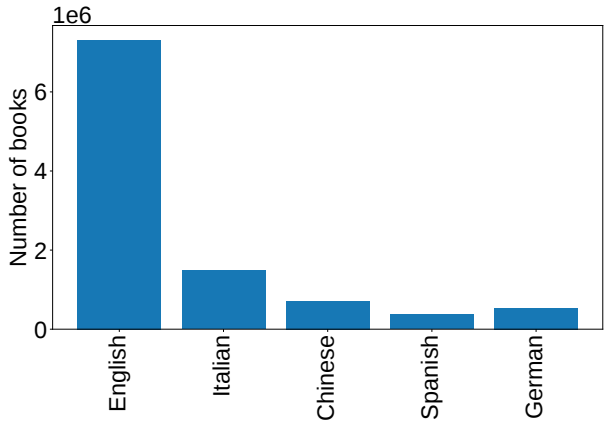

Figure 3.14: Number of books labeled by top 5 language

- 2901170 authors
- 14 rates on average per user

Let's now talk about the description and the genres associated with the books. Not having them at the start of the thesis, we explicitly requested those data to Anobii to have richer features to represent the books.
The description file is composed by the following attributes:

- Item content ID: unique identifier of the description.
- Item ID: identifier of the associated book.
- Source: provider of the description.
- Content: The description itself.
- Language: language of the description.

The file contains descriptions on 245411 different Italian books. We can also note that the item ID is usable as a join key to integrate descriptions on the items file. As we can see from 3.15, descriptions with low characters may contain noisy and not meaningless descriptions.

	item_id	item_content_id	source	content	language
0	3246695	1693807	Book Description	2 volumi	it
1	2578716	1444612	Book Description	Con testo in dialetto italiano e inglese.	it
2	1966373	1432949	Book Description	I Garzanti ; 359	it
3	3277275	1708648	Book Description	Senza data [1963?]	it
4	1192183	1662636	Book Description	Con testo a fronte	it

Figure 3.15: Examples of not meaningful book descriptions

Going on, the genres file is composed by:

- Item ID: identifier of the associated book.
- Name: genre associated with that book. There are 41 different genres in the file, visible in the figure 3.16.
- Language ID: language associated with the book.
- Score: Community voting representing the strength with which the book is represented by the genre.

It is worth to mention that the genre association is not unique: a book can have more than one associated genre, each one voted with a certain score. In particular, each book has on average 5 genres. By looking at 3.16 we can see that the majority of the books in the full dataset are either historic or fictional books.

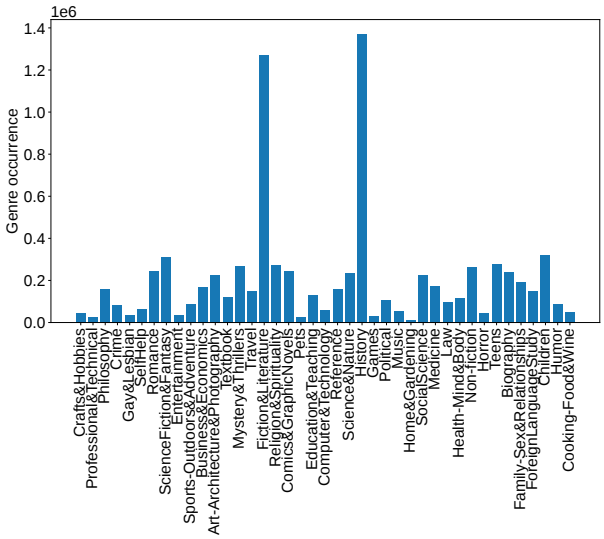

Figure 3.16: Occurrence of genres in the full Anobii dataset

3.2.2 Anobii Filtering and processing

Let's now filter and process the vast quantity of Anobii data.
Our objective is to merge this dataset with the BCT one in order to create a dataset that is rich both in terms of features and of items. So, the filtering operation will be similar to the BCT ones but we will need to also consider the additional features. To fairly compare the content of the dataset with the BCT one, we proceeded as follow:

- Removal of all non-Italian books in the dataset by using the attribute 'language' in the items file. The number of items went from 8021517 to 1202402.
- Removal of all non-books items from the dataset. This was done by using the 'binding' attribute of the items file which contains the type of the cover of the book. In particular, we kept the ones labeled as 'Paperback' and 'Hardcover' because that type was associated with books while we discarded the 'Others' binding because it referred to a larger group of media. This, admittedly, may cause the loss of traditional books labeled under the 'Others' binding but, as we will see from the numbers, the quantity of the data is still satisfying. Moreover, our objective was not to have the maximum possible number of items but rather to have a safe set of identified narrative books. The number of items went from 1202402 to 403090.
- Removal of periodical books in the same fashion as the BCT filter. The number of items went from 403090 to 402070.
- Integration of author information related to books joining the files 'author items' and 'items' on book id.
- Removal of all books with less than a certain threshold of ratings. Like in the BCT case with loans, this threshold should be decided in merging phase to assess the impact of this choice on our objective described in 3.2.

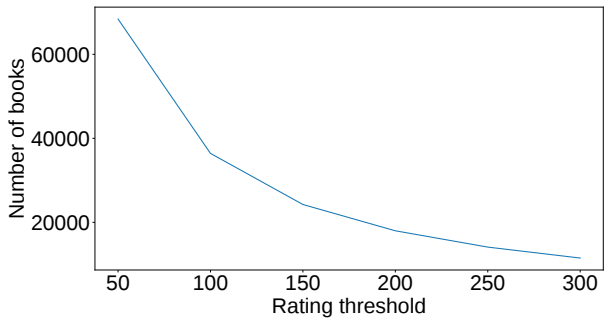

Figure 3.17: Number of Anobii books depending on threshold

It is now important to assess the additional features to filter.
As we have seen from the Anobii attribute tables, we may use the 'item review' field on the 'link person item' file to filter away certain interactions. We noticed that we could consider Anobii ratings both as implicit and explicit by considering just the interactions (as in BCT) in the first case and the votes in the latter. This led us to discard away votes equal to 0 when considering Anobii ratings as explicit because they could not be considered as ratings but, in a stage where implicit feedbacks, those interactions are not only to keep but also precious assets.
Let's now have a look at the genres occurrence after those filtering operations.

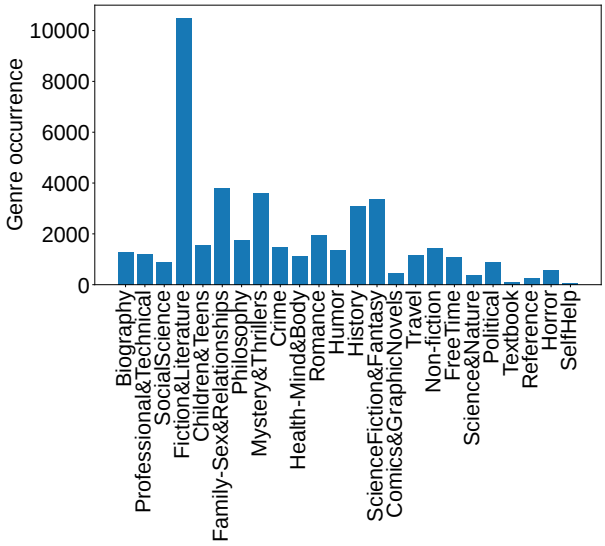

Figure 3.18: Occurrences of genres after filtering

We can see that 'Fiction&Literature' is the most occurrent genre in the filtered dataset. Being present so much with respect to other genres, we decided to remove it because it would be not meaningful in terms of book characterization of a limited set of items. Moreover, 'Textbooks', 'References' and 'Self Help' were also deleted due to them not being necessary for the objective of the project. While handling the genres, we also noticed that the project required a relatively low number of total genres due to the necessity of integrate genre visualization on a VR environment(see 1.4). Because of this, we decided to lower the genres from 41 to 20 by aggregating similar genres in groups as explained in diagram 3.19. To do this, we considered the entropy value calculated on the genre values in the book dataset. Genre aggregations that led to a decreasing entropy were applied because they allowed to form sub-clusters of similar genres.
Finally, considering that each book genre was assigned through community voting and each book may have more than one genre, it was necessary to select a top K

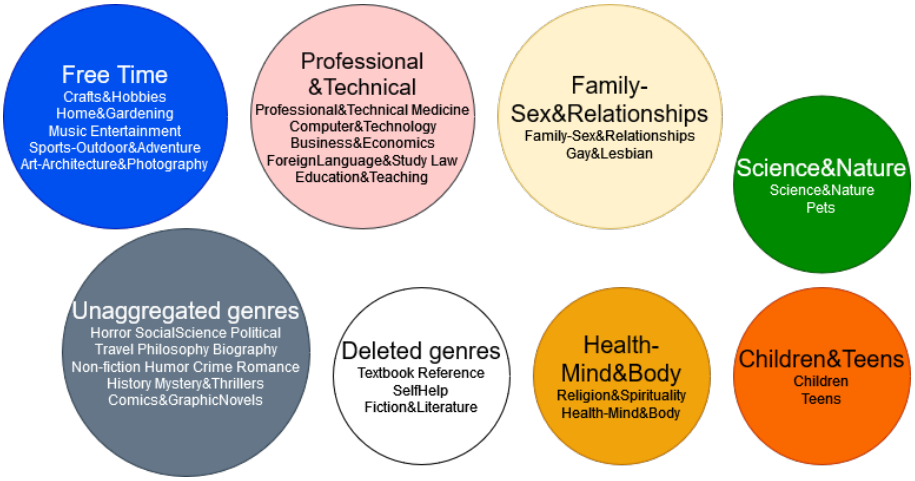

Figure 3.19: Aggregation of the 41 genres into 20

of genres in order to avoid not meaningful genres with very few votes. Due to this, we decided to decided this K value by looking at the data.

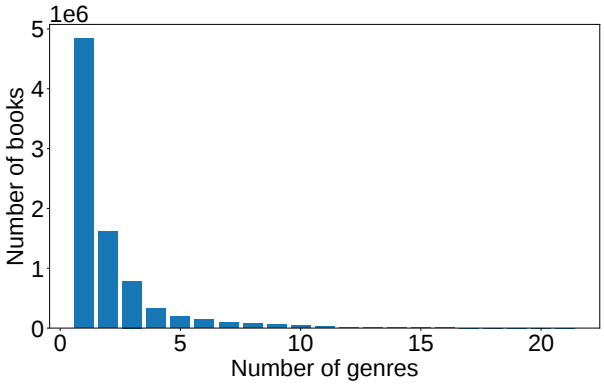

Figure 3.20: Number of books per possessed genres

We can note from figures 3.21 and 3.20 that not only the vast majority of the books possesses at most 4 genres, but the average score of the votes referring to genres decreases very rapidly after the fourth. So, we decided to fix the K value to 4. This means we will keep at most 4 genres per book with the guarantee that we will have meaningful votes and we will also cover the majority of the dataset. The integration of genres into the items dataset was done through an inner join on the book ID. On the genre side, book ID are not necessarily unique due to the presence of many associated genres per row so, after the joining, genres were aggregated for each book, creating a new 'genre' field containing the list of the assigned genres. Going on, I proceeded to integrate the descriptions into the items dataset. To do

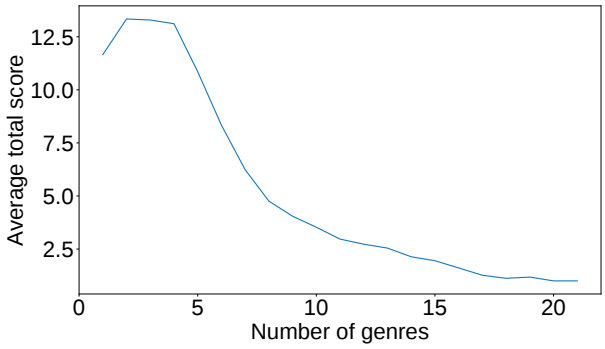

Figure 3.21: Average score per number of genres

so, I filtered away descriptions with less than a certain threshold of characters for two main reasons:

- It is more probable that descriptions with a good number of characters are more meaningful.
- Some short descriptions were actually noisy and to discard as explained in 3.2.1.

Considering that, through a manual review of the description file, the majority of the noisy descriptions are short and do not follow a predefined pattern, I decided to filter away those shorter than 50 characters. To sum it up, the following diagrams represents all the filtering and processing steps on Anobii data.

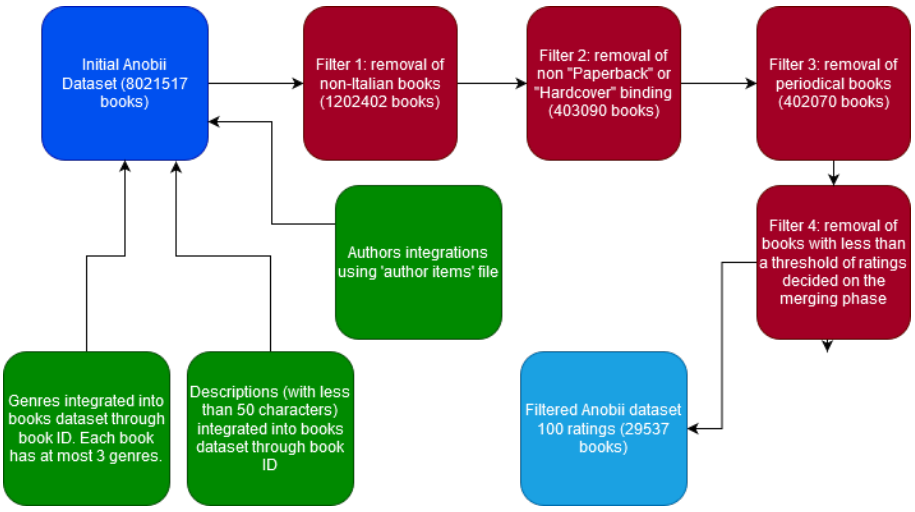

Figure 3.22: Diagram showing filtering operations on Anobii data

3.3 Merging Anobii and BCT data

Merging Anobii and BCT data is the final objective of this chapter.
From the preceding sections of the chapter, we noticed that some choices are to be made during this phase.
In fact, it is important to remember that, for the finality of the project, we want to have a limited set of no more than 3000 books with full features. This is done in order to assure an easily manageable test during the deployment on a real library environment. To such end, we must not only merge the dataset but also check how many books result in output from the merge and tune our remaining filters to reach our objective.
First thing, we should find what attribute to use for the merge. To that end, the BCT item ID and the Anobii item ID are obviously not usable because they relate to different datasets. However, the title should be an efficient enough identifier for a book so we chose that as the merge key.
As notable in tables 3.3 and 3.2, the formatting for titles in the two dataset is quite different:

- Anobii dataset possesses both a title and a subtitle field in the item file and, overall, the title was formatted in a clean way to assure a correct visualization on the social network.

	item_id	title	sub_title
0	724556	Squatter	Una storia di case occupate
1	1564435	Non avere paura! Affronta e vinci il nemico che c'è in te	None
2	1307247	Italiani	Racconto etnografico
3	1010990	Colombia, il paese dell'eccesso	Droga e privatizzazione della guerra civile
4	7969946	Il partito della polizia	Il sistema trasversale che nasconde la verità degli abusi e minaccia la democrazia

Figure 3.23: Examples of Anobii titles

- BCT dataset does not possess a separation between title and subtitle; moreover, the quality of the formatting is way more noisy and requires dedicated processing.

	manifestation_id	title
0	202094	Okey dokey, sono un punk / Domenica Luciani ; illustrazioni di Roberto Luciani
1	910309	Che cosa ho in testa : immagini di un mondo in cui valga la pena / AA. VV. ; a cura di Alberto Rollo
2	382413	Morte a Firenze : [un'indagine del commissario Bordelli] / Marco Vichi
3	278938	I due sergenti / un film di Enrico Guazzoni
4	436078	Schindler's ark / Thomas Keneally

Figure 3.24: Example of BCT titles

By trasforming BCT titles in a way that extracts just the title info as represented in the Anobii dataset, we can merge the two datasets. Although BCT titles are quite noisy, we can infer some rules that defines the way the field is structured:

- Titles and subtitles are separated by each other using the separator " : ".
- The title and other informations about the book are separated by a " / ".
- As we can see from the first row in figure 3.24, sometimes the author is embedded in the title and separated by a " / " from the title and by " ; " from the other info. In that case is important to extract and put them in the correct field because the author field for these row is a deceptive "None" value.

After having applied these rules, the BCT dataset titles looks like figure 3.25.

	edition_language	edition_date	title	author	publisher
15	ita	1994	Queen	Alex Haley	Rizzoli
16	ita	2003	Filastrocca a colori	Massimiliano Maiucchi	Pika
17	ita	2003	La prova del fuoco	Tami Hoag	Sperling & Kupfer
18	ita	2003	I sette delitti di Roma	Guillaume Prévost	Sellerio
19	ita	2004	La nuvola Olga e il temporale	Nicoletta Costa	Emme

Figure 3.25: BCT dataset with Anobii-formatting style

Admittedly, BCT dataset was very noisy and the before-mentioned rules only applied to the majority but not all the books. This should lead to a certain loss of books during the merging phase. However, as long as we choose threshold values (see preceding sections) that allow us to respect our objective, this is acceptable.
By tuning the filters thresholds described in the preceding sections to 50 for BCT and 300 for Anobii, we managed to extract 2431 books with integrated features which meets our objective requirements. Then, we decided to only keep the Anobii book ID as an identifier given that we have no need for two identifiers at the same time.
Finally, we proceeded with the attribute selection: the attribute should not only be considered by its potential to be used for feature extraction or for the recommender models but also by its capacity to offer metadata visualization on the VR environment (see 1.4). By using the attribute 'encrypt item ID' on 3.3, it was possible to provide cover images for the books due to the structure of Anobii image links containing this identifier. Moreover, visualizing additional data like the average rating can make the user more aware about the book and the opinions of other people.

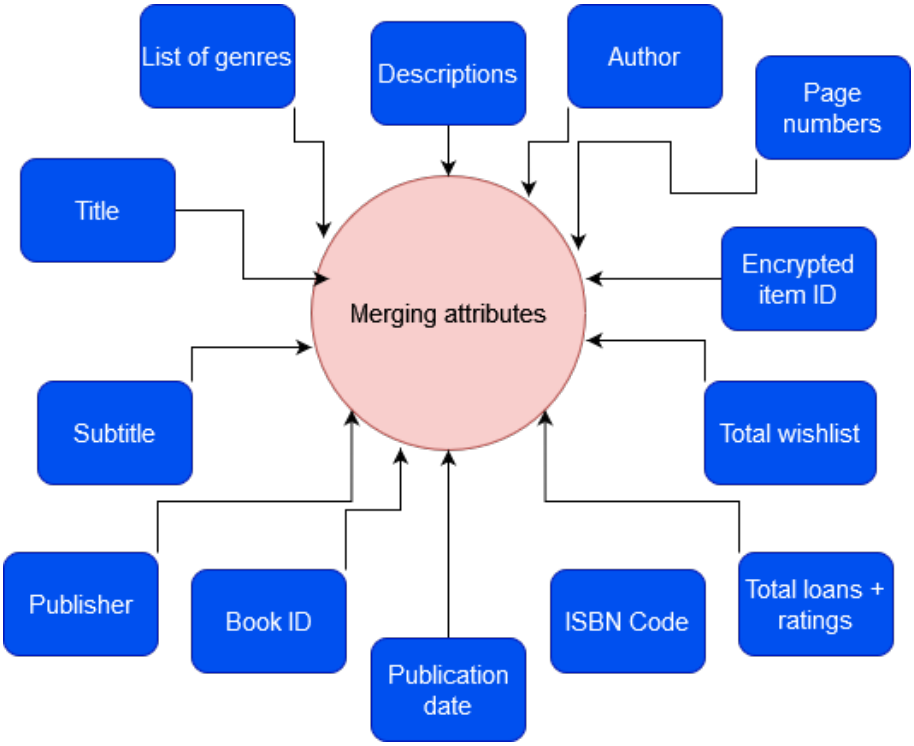

Figure 3.26: Diagram that shows the chosen attributes for merging

Anobii files - Part 1		
File name	Attributes list	Disk space
Author display	• author ID: unique identifier of the author. • author name: author name in "Name Surname" format. • language: language spoken by the author.	111 MB
Author items	• Item ID: identifier of the books written by the author. • Author ID: identifier of the author.	161 MB
Cities	• City ID: unique identifier of the cities. • Country ID: identifier of the country related to the city. • City name: Name of the city. • Language: language spoken in the city.	2 MB
Countries	• Country ID: unique identifier of the countries. • Country symbol: Link to the flag of the country. • Have city: tells if the country has cities in the city file. • Language: Language spoken in the country.	111 MB
Items	• Item ID: unique identifier of the book. • ISBN: ISBN code of the book. • Title • Subtitle: sub-name of the book. • Publisher • Binding: Book binding of the book. • Last update • Average rating: average of all the explicit ratings. • Total reviews • Language: language with which the book is written. • Total wishlist: tells how many users wished that item. • Added date: indicates when the book was added. • Publication country ID: publishing country identifier. • Encrypt item ID: encrypted identifier of the item. • Number of pages	1269 MB

Table 3.3: Anobii files and attributes - Part 1

Anobii files - Part 2		
File name	Attributes list	Disk space
Language mapping	• Language ID: unique identifier of the languages in the database. • Language name • Language name English: language name translated in English.	1 MB
Link person item	• Person ID: identifier of the interacting user. • Item ID: identifier of the rated book. • Item review: Explicit rate given by a user to a book. Goes from 1 to 5. If 0, the book has not been rated. • Finish date: Date when the book has been finished by the user (if provided).	5091 MB
Link person item comment	• Person ID: identifier of the commenting user. • Item ID: identifier of the commented book. • Comment content: content of the comment in string form. • Comment date • Language: language of the comment	1922 MB
Person	• Person ID: unique identifier of the user. • Language • Country • City • Sex • Date of birth • Total followers • Total following • Total reviews • Active User: flag used to indicate if the user is currently active	59 MB

Table 3.4: Anobii files and attributes - Part 2

Chapter 4

Algorithms for feature extraction

4.1 Keyword extraction

Keyword extraction techniques are used to create a set of meaningful words related to book descriptions.
This task has been applied on our book dataset for two main reasons:

- Representing descriptions as a structured set of words instead of a block of unstructred text is more manageable. This will allow us to use the keywords as a feature for a content-based or an hybrid recommender system.

- Due to one of the objectives of the project to build a VR environment (see 1.4) that could wrap the recommender model in a user-friendly interface, a very limited set of keywords representing each of the 20 possible genres (see 3.2.2) were needed in order to allow the user to visualize couples of genre/keyword pairs and choose couples to filter certain type of books from the suggestions.

In diagram 4.1 we can see a simple flowchart representing this double objective.

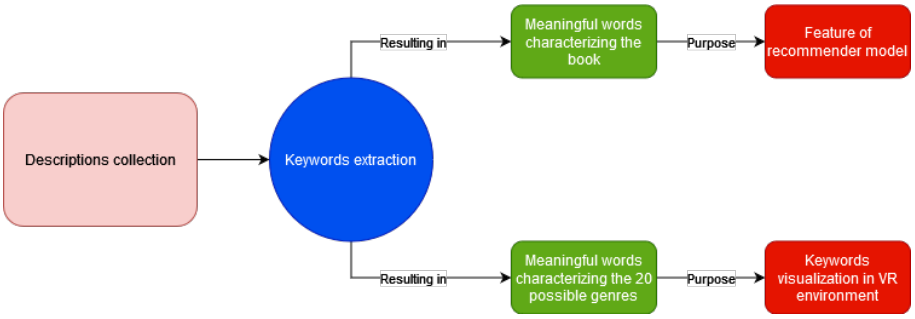

Figure 4.1: Diagram representing the keyword extraction objectives

The first output will be a list of notable keywords associated with the books in order to try to use them for a recommender model. The second output will be a list of a very limited set of keywords identifying a whole genre out of the 20 present in the dataset.
Our starting point is the merged dataset of approximately 2500 books formed in 3.26. As we have seen during the Anobii filtering phase, the description contains a "content" field composed by a string that has been integrated into the dataset after a filtering on the number of characters to limit the noisiness and keep the descriptions as meaningful as possible. From that, to apply a keyword extraction technique, we will need to apply some pre-processing. This requires the use of NLP to perform a task called part of speech (POS) tagging. To perform this on our descriptions collection we should use a dedicated library. I chose SpaCy [20] NLP library to perform these operations due to its completeness. In fact, it possesses modules to deal with the totality of the pre-processing steps:

- A tokenizer to transform a text into a set of tokens. This is used to handle unstructured data in an easy way by transforming it into a structured form.

- A pre-trained POS tagger compatible with the Italian language using a dedicated dictionary file. This is used to remove words that are not nouns. This means that our objective is to remove all the other parts of speech. In fact, our objective is to extract meaningful characteristics better represented by that part of speech.

- A lemmatizer and a stemmer module (see 2.2.1 for details). This module can be used to remove differences between words that come from the same base.

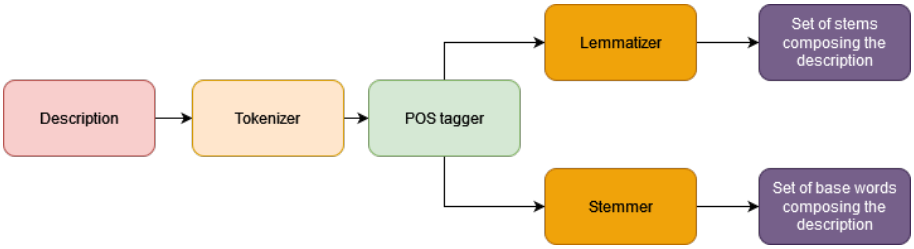

Figure 4.2: Description passed through SpaCy modules

Keywords extraction material		
Component	Usage	Library
TF-IDF vectorizer	Object used to tokenize a collection of documents. It possesses the functions: • "fit transform" - Tokenize a text and create a TF-IDF score for each word related to each document in the collection. • "to coo" - transform the scores structure into a COO matrix [21] for efficiency. • "get feature names out" - Extract the terms from the terms-score matrix	sklearn
scores ordering	Orders a set of scores in a COO matrix	user-defined
top k extraction	Extracts the first top k keywords per scores	user-defined
get keywords	Retrieves the keywords associated with the top scores in output from top k extraction	user-defined

Table 4.1: Required objects for keywords extraction

As we can see from figure 4.2, we have the choice between a stemmer or a lemmatizer. However, we must consider that the descriptions are written in Italian. Many Italian words may be processed as having the same stem although they mean different things (for example, 'mostra' means 'show' while 'mostro' means 'monster', although they both have the same stem 'mostr'). So, I considered the lemmatizer to be the safe choice to avoid ambiguous cases.
After obtaining the filtered nouns-only, descriptions, we can apply TF-IDF (see 2.2.1) without having the risk of obtaining stopwords or not meaningful parts of speech in output.

The TF-IDF algorithm consists in giving a score to a word related to a text calculated as:

$$w_{i,j} = f_{i,j} \cdot log(\frac{N}{df_i}) \tag{4.1}$$

The formula explains that the score $w_{i,j}$ related to word i in document j is calculated as the occurrence frequency $f_{i,j}$ of the word in the considered document multiplied by the logarithm of the total number of documents N over the number of documents that contain the considered word df_i. The first term is used to give relevance to frequent words while the second term is used to penalize frequent words that are also frequent in many other documents. By doing this, we are extracting words that are both important and characteristics of the considered document.

4.1.1 Book keywords

To apply the TF-IDF algorithm to the collection of descriptions, I used the components described in table 4.1. To extract the top K keywords defining the individual book, we proceeded in this way:

1. Create a dataframe with the structure (book identifiers, descriptions). This will be used to create a list of descriptions with ordered indexes with respect to the dataframe. By doing so, we can re-obtain the identifier after finding the top keywords and proceed with the data integration.

2. Instantiate a TF-IDF sklearn vectorizer and create the token vocabulary by passing the list of filtered descriptions (called 'corpus') in input to the "fit transform" function.

3. Call the "get feature names out" function of the vectorizer to obtain the list of individual terms.
   The list will contain all the terms contained in the collection.

4. Pass each filtered description of the corpus and the indexed list of terms to the "get keywords" function and call it.
   The function will transform the passed description into a set of (index, scores) elements, convert it into a COO matrix and order it through the "scores ordering" function.
   The COO Matrix will be a matrix having indexes (representing the location of the term in the list of step 3) as rows and scores as columns.

5. Call "top extraction" by passing a k constant (indicating the quantity of keywords per description) and the ordered matrix: it will use the indexes to retrieve the original terms and will output a list of k keywords for the description. This function will be called for each description of the corpus.

After obtaining the keywords output, we can integrate them into the dataset (see Step 1 of the preceding list).

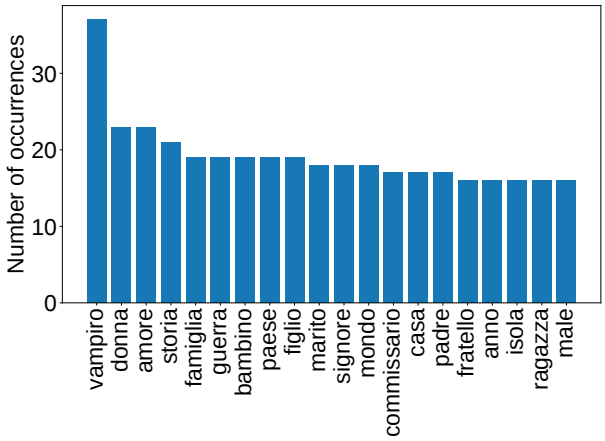

Figure 4.3: Top 20 keywords in top 5 extraction algorithm

The most 20 frequent ones with a constant k of 5 can be seen on figure 4.3. From the bar chart we can see how some frequent keywords are noisy, like "ugrave" (which is simply noise) or "libro" (which means "book" in Italian and is not very meaningful). To attenuate this effect, I manually filtered frequent and meaningless words with a user defined function after the descriptions filtering. However, from 3.2.1, we can note that we possess the comments the users gave to books. Anobii contains standard social network comments but users also write reviews on books on that section. So, we tried to concatenate descriptions to comments to see its impact on the extraction.

In figure 4.4 we can see that there is an explosive number of comments with very

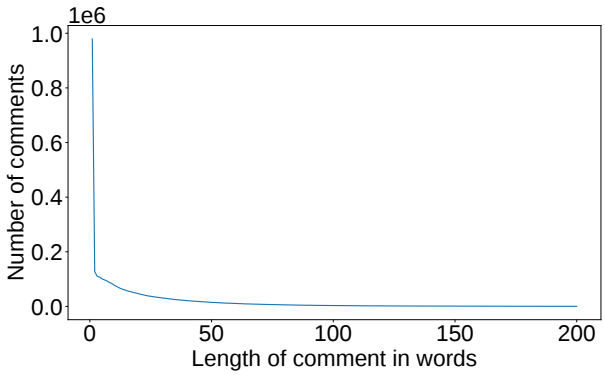

Figure 4.4: Number of comments per length in words

low words. We can infer that they are probably spam or appreciation comments. At approximately 80 words, the number of comments begin to stabilize in the order of thousands. I chose to consider only comments with at least 80 words for two reasons:

- The stabilization in numbers means that less users tend to comment that long and thus is probable that these type of comments are informative in terms of the book.
- The fact that comments with 80 words (and more) are less than 10000 per category means that the SpaCy tokenization takes less computational time. In fact, the computation is longer for big texts and even longer for big datasets. By chosing a safe threshold, time is saved while keeping informative comments.

By looking at the impact of the "description plus comments" concatenation on keywords extraction with k equal to 5, we can note that, although the top keywords themselves are mostly the same, way more books now contains them.

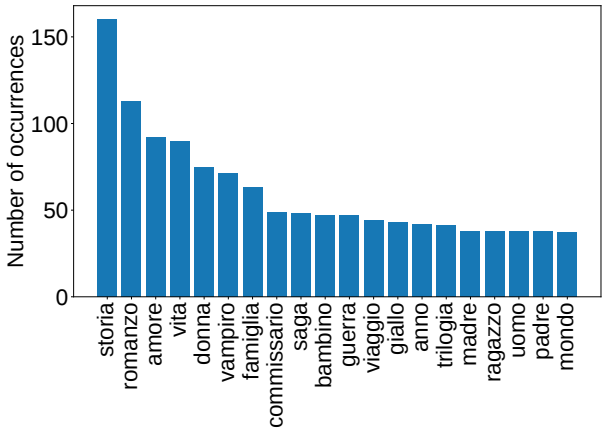

Figure 4.5: Top 20 keywords in top 5 extraction algorithm with comments

This means that concatenate comments with descriptions (at least in a top 5) gives more relevance in terms of frequency to already frequent keywords and tends to penalize the less famous ones. This is confirmed by 4.6 where we can see that concatenation of descriptions and comments results in a set of keywords with an higher entropy than the one resulted from just the descriptions.
The evaluation of the keywords is not so simple: in fact, we do not have labeled data and the quality of the keywords is quite subjective. We should remember, though, that these keywords will be used as feature for the recommender model: an indirect way to evaluate their quality would be to assess their influence on the output of the recommender. Thus, this assessment is postponed to chapter 5.

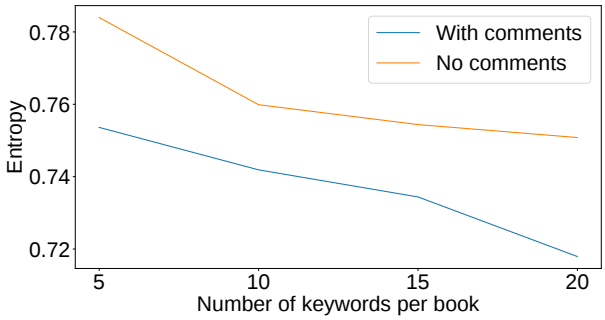

Figure 4.6: Entropy value per number of keywords per book

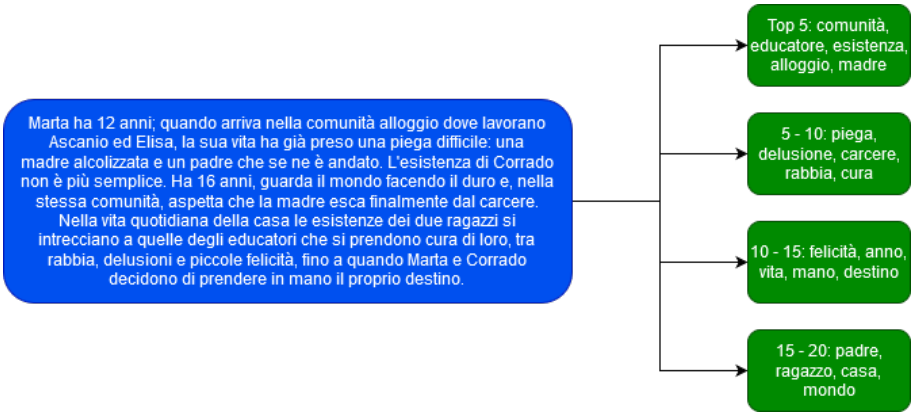

Figure 4.7: Real example of keyword extraction with a dataset description

4.1.2 Genre keywords

Now, we need to extract a very limited set of keywords for representing genres for our second objective: said extraction heavily depended on the requirements of the VR environment and parameters for the algorithm were chosen based on that. Due to that, each genre should be represented by no more than 20 keywords.
The keyword extraction is almost equivalent to its first version but with a few tweaks: instead of using the entire dataset, we will use a genre-thematic dataset containing only books of the considered genre. This means that we will use the algorithm 20 different times for 20 different genres. This is done in order to find common words between clusters of books with the same genre and thus to extract more specialized keywords.

However, for the genre-themed keywords algorithm we need to also add a 'keyword selection' phase in order to consider two main factors to make the keywords usable in a user-friendly environment:

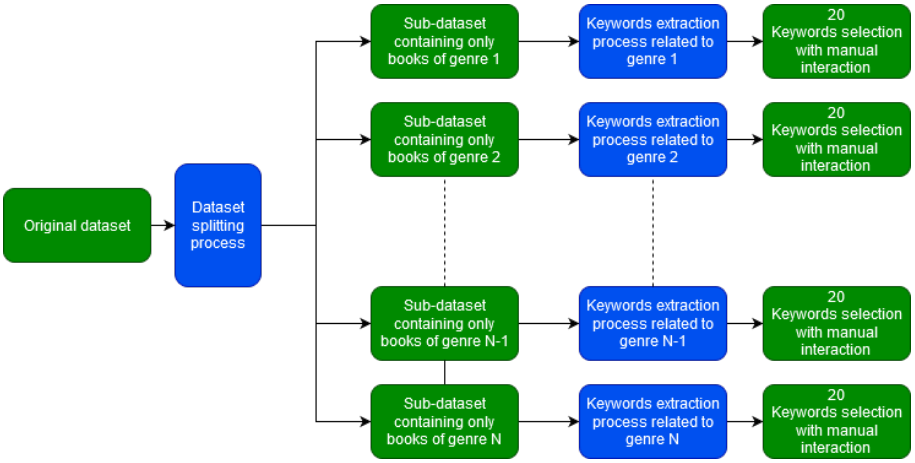

Figure 4.8: 20 keywords extraction process

- Coverage - the 20 keywords should refer to the maximum possible number of books in the dataset. Considering the 20 keywords will be used by the BCT user to filter and explore books, it is auspicable that the keywords permit an almost full exploration of the items. However, although ideal, maximum coverage is unobtainable due to the 20 keywords limit. Moreover, taking just the top 20 keywords in terms of occurrence does not give us any guarantee that the coverage is at his maximum possible.
- Interpretability - Supposing we manage to cover the maximum amount possible, we still need to verify if the keywords are meaningful and usable to be used in a user-friendly space.

The following measures will be applied to respect these two factors.
The keywords extraction phase will output a CSV file containing the genre, the list of chosen keywords and a "coverage" value. The coverage indicates the percentage of books in the genre dataset that have a reference in the keywords list. The higher it is, the more books are represented by the 20 keywords.
The subsequent phase is keyword selection where manual tuning is performed in order to increase the quality of the words in terms of interpretability by using human interaction. Using a command line interface, the user can choose from the list of chosen keywords the one to substitute and then can visualize the list of possible substitutes (taken from the full list of words extracted from the dataset). After choosing a substitute word, the user will see the impact on coverage of the substitution and may then decide on a trade-off.
As we can see from 4.9, the coverage increases steadily by choosing higher k constants for keywords extraction and becomes almost 100% with k equal to 50. So that was the chosen constant.

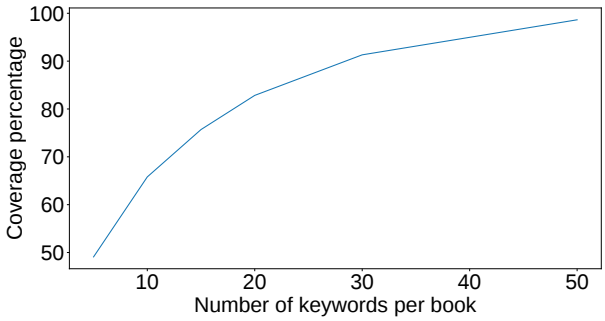

Figure 4.9: Coverage per number of keywords per book

In 4.10 we notice that, although the keywords have a very high coverage, they sometimes are not very meaningful. To attenuate this effect, I implemented a command line interface used to make the user that handles the quality of the keywords swiftly substitute the worst ones. This interface allows the procedure with these user steps:

- User chooses the genre via keyboard input.
- After visualizing the current keywords list, the user types the one to substitute
- The algorithm calculates all the possible keywords like during extraction. A dictionary containing the keywords with their occurrence is visualized.
- The user chooses the new keyword.
- The interface makes the user see the old coverage percentage against the new one.
- The interface asks the user if he wants to proceed. Then, with a positive answer, substitutes the two keywords and saves the new file.

The interface (showed in a minimal example in 4.11) allows for the user managing the BCT virtual environment the possibility to control the trade-off between book coverage and keywords quality, thus enhancing the readers experience.

4.2 Sentiment analysis

As already explained in 2.2.2, sentiment analysis is a way to extract new knowledge about our books by using data we already possess. By extracting the mood (the overall tone of the description) and the sentiment (the feelings expressed by the description) we may:

	genre	keywords	coverage
0	Comics&GraphicNovels	['storia', 'anno', 'vita', 'fumetto', 'morte', 'strada', 'amico', 'mondo', 'indagine', 'gemelle', 'mossa', 'pomeriggio', 'quaderno', 'sfericità', 'vampiro', 'manga']	100.000000
1	Family-Sex&Relationships	['storia', 'vita', 'romanzo', 'anno', 'uomo', 'famiglia', 'donna', 'saga', 'mondo', 'marito', 'raccolta', 'paura', 'ricerca', 'corpo', 'colpevole', 'vampiro', 'attore', 'ghiaccio', 'fantascienza', 'dolore']	97.474747
2	Humor	['storia', 'vita', 'anno', 'romanzo', 'mondo', 'donna', 'viaggio', 'raccolta', 'teatro', 'caso', 'giorno', 'risata', 'riferimento', 'saga', 'prospettiva', 'confronto', 'voglia', 'cassetto', 'saggio', 'casa']	99.392097
3	History	['storia', 'romanzo', 'vita', 'anno', 'uomo', 'mondo', 'avventura', 'morte', 'giallo', 'paese', 'realtà', 'opera', 'stato', 'rivoluzione', 'religione', 'campo', 'legge', 'tomba', 'resistenza', 'leggenda']	96.783217
4	ScienceFiction&Fantasy	['storia', 'romanzo', 'vita', 'mondo', 'saga', 'anno', 'uomo', 'mistero', 'morte', 'avventura', 'ragazza', 'indagine', 'ragazzo', 'filo', 'territorio', 'coscienza', 'signore', 'colore', 'club', 'ugrave']	96.593674
5	Romance	['storia', 'vita', 'romanzo', 'amore', 'saga', 'moglie', 'anno', 'uomo', 'piano', 'mistero', 'voce', 'guerriero', 'amicizia', 'scrupolo', 'giallo', 'ricerca', 'dama', 'famiglia', 'studentessa', 'nausea']	99.288256
6	Travel	['storia', 'romanzo', 'vita', 'viaggio', 'anno', 'avventura', 'uomo', 'mondo', 'mare', 'azione', 'figlio', 'vacanza', 'ragazzo', 'casa', 'conclusione', 'bordo', 'appassionato', 'ambientazione', 'taverne', 'lettera']	99.680511
7	Mystery&Thrillers	['storia', 'romanzo', 'anno', 'vita', 'caso', 'uomo', 'giallo', 'saga', 'indagine', 'mondo', 'morte', 'ragazzo', 'ragazza', 'ora', 'ambiente', 'mistero', 'marito', 'ugrave', 'signore', 'soldo']	96.683417
8	FreeTime	['storia', 'romanzo', 'vita', 'mondo', 'anno', 'donna', 'caso', 'avventura', 'viaggio', 'arte', 'pesce', 'musica', 're', 'humour', 'arma', 'uomo', 'esistenza', 'terra', 'uso', 'marito']	97.002725
9	Non-fiction	['storia', 'vita', 'anno', 'mondo', 'romanzo', 'uomo', 'paese', 'scienza', 'poesia', 'esperienza', 'arte', 'cronaca', 'luogo', 'stato', 'segno', 'scrittura', 'signore', 'spirito', 'creatività', 'monte']	95.744681
10	Biography	['vita', 'storia', 'anno', 'romanzo', 'uomo', 'paura', 'mondo', 'nonno', 'paese', 'orrore', 'giornalista', 'scienza', 'sala', 'luce', 'madeleina', 'impresa', 'carcere', 'maschera', 'blu']	100.000000
11	SocialScience	['storia', 'vita', 'mondo', 'uomo', 'anno', 'concetto', 'romanzo', 'viaggio', 'raccolta', 'motivo', 'prefazione', 'opera', 'modello', 'allievo', 'padre', 'scienza', 'democratico', 'vecchiaia', 'seno', 'globalizzazione']	99.190283
12	Political	['storia', 'anno', 'mondo', 'vita', 'romanzo', 'concetto', 'prova', 'paese', 'guerra', 'idea', 'spettacolo', 'spia', 'impresa', 'terra', 'nemico', 'domenica', 'globalizzazione', 'prigionia', 'scalpore']	100.000000
13	Crime	['storia', 'romanzo', 'anno', 'caso', 'delitto', 'uomo', 'mondo', 'polizia', 'thriller', 'morte', 'vita', 'comunità', 'faccia', 'sfida', 'collo', 'sorveglianza', 'preda', 'quaderno', 'rapporto', 'stazione']	99.487179
14	Children&Teens	['storia', 'anno', 'romanzo', 'vita', 'avventura', 'mondo', 'amore', 'fratello', 'scuola', 'viaggio', 'ragazzo', 'ruolo', 'spettacolo', 'missionario', 'pallone', 'gemelle']	100.000000
15	Philosophy	['storia', 'vita', 'romanzo', 'uomo', 'idea', 'mondo', 'scrittura', 'concetto', 'opera', 'anno', 'prefazione', 'verso', 'caso', 'componente', 'crisi', 'allievo', 'luogo', 'dio', 'normalità', 'bruttezza']	98.022599
16	Horror	['storia', 'vita', 'giorno', 'uomo', 'ragazzo', 'delitto', 'anno', 'ragazza', 'ugrave', 'creatività', 'lotta', 'media', 'ape', 'manuale']	100.000000
17	Health-Mind&Body	['vita', 'storia', 'uomo', 'romanzo', 'mondo', 'amico', 'concetto', 'raccolta', 'casa', 'chiesa', 'manifestazione', 'scuola', 'cibo', 'avventura', 'metamorfosi', 're', 'vendetta', 'disco', 'tragedia', 'discepolo']	99.626866
18	Professional&Technical	['storia', 'vita', 'mondo', 'romanzo', 'uomo', 'caso', 'anno', 'informazione', 'indagine', 'sera', 'ecc', 'opera', 'sistema', 'scuola', 'scienza', 'scrittura', 'suspance', 'modello', 'concetto', 'eco']	97.735849
19	Science&Nature	['storia', 'vita', 'scienza', 'mondo', 'anno', 'città', 'cibo', 'organismo', 'piè', 'giornata', 'cristallo']	100.000000

Figure 4.10: Output of the genre-keywords extractor

```
Welcome to the manual tuning interface. Choose a genre.
Write exit to close the interface.
 Comics&GraphicNovels
These are the chosen keywords for that genre.
['storia', 'anno', 'vita', 'fumetto', 'morte', 'strada', 'amico', 'mondo', 'indagine', 'gemelle', 'mossa', 'pomeriggio', 'quaderno', 'sfericità', 'vampiro', 'manga']
Choose a keyword to substitute or write exit to exit.
 anno
Calculating possible keywords..
{'storia': 74, 'anno': 49, 'fumetto': 48, 'vita': 42, 'mondo': 33, 'romanzo': 31, 'uomo': 30, 'avventura': 23, 'amore': 21, 'casa': 19, 'ragazzo': 18, 'bambino': 17,
```

Figure 4.11: Extract of the manual tuning interface

- Use these informations as a feature for recommender models.
- Provide additional informations about the BCT environment.

However, models capable of label unstructured text with a sentiment are very complex and extensively trained. Moreover, the model must be able to process Italian language. As cited in 2.2.2, the 'feel-it' model provided both a compatibility with the Italian language and state of the art performances so I decided to use it on our corpus of descriptions.
'feel-it' library assigns, to each description, one out of 4 different sentiments (fear, anger, sadness and joy) and one out of 2 moods (positive and negative).
By visualizing some data in 4.12, we can see that the 'negative' labels are more than the 'positive' ones.
Feel-it also labeled descriptions in order to output a sentiment like we can see in 4.13.
By these two informations, we can see that 'feel-it' considers the majority of the books as having negative vibes considering that the three negative emotions (anger, fear and sadness) and the 'negative' mood are the most frequent. Although this is reasonable, because having joyful books with a majority of negative books would have been contradictory, we must remember that this library (and, in general, the

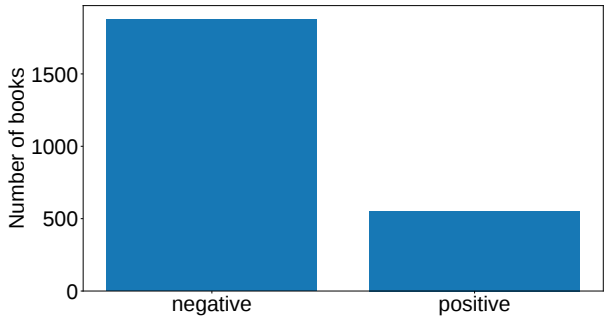

Figure 4.12: Number of books per mood label with 'feel-it'

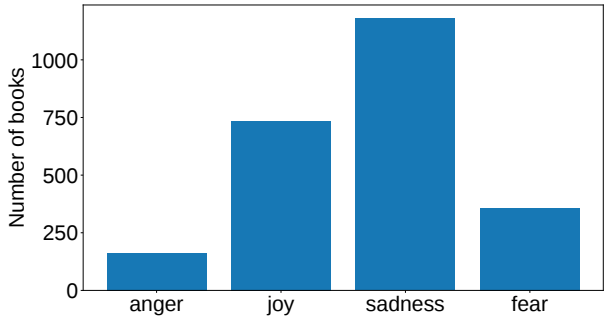

Figure 4.13: Number of books per emotion label with 'feel-it'

most famous sentiment analysis libraries) are trained on social network posts that are shorter than descriptions. So, it is possible to have mistakes. To attenuate this effect we will use two additional sentiment analysis libraries to perform a majority voting on the mood. However, other sentiment libraries compatible with the Italian language are rare and the most famous ones are just in English. To resolve this problem, I have used a Google Translate API in order to translate the descriptions in the English language.

Now, we can use famous sentiment analysis libraries to obtain a 'positive' or 'negative' output. The libraries I will use for this purpose are VADER and Textblob.

As we can see from 4.15, the library assigns more 'positive' labels than 'negative', contrarily to feel-it.

However, VADER gives more or less the same number of 'positive' and 'negative' labels.

To choose the final mood label I decided to perform a majority voting between the 3 labels. Considering that the mood is a binary value, having 3 different labels will surely lead to a winner in terms of votes.

To evaluate the mood and the sentiment labels, we will assess their impact as features of the recommender models due to our data being unlabeled.

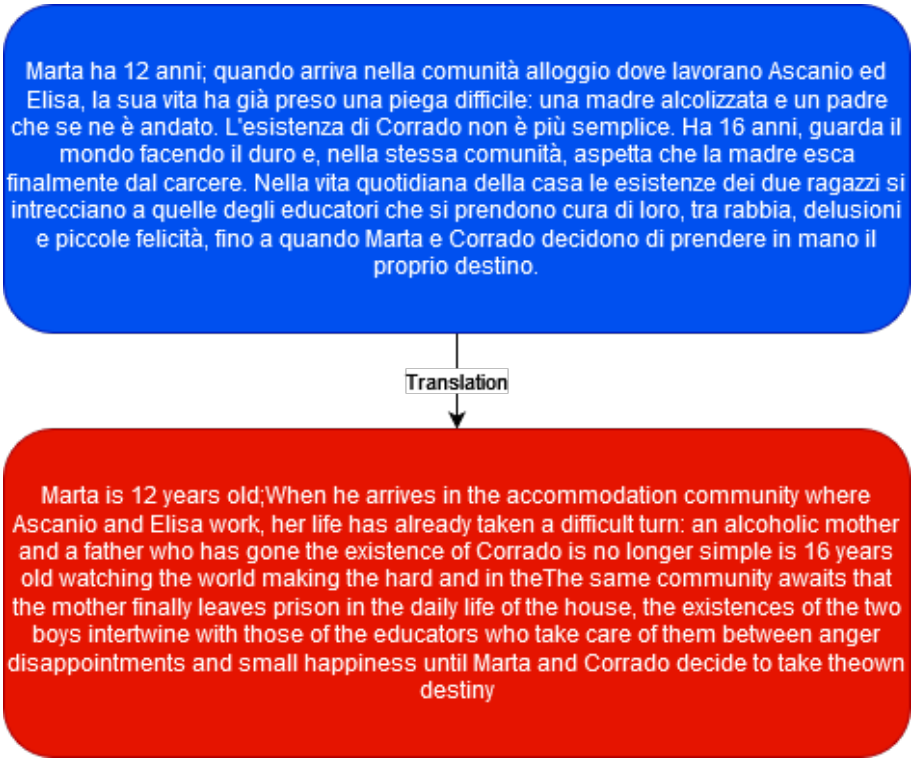

Figure 4.14: Description translated in English with Google API

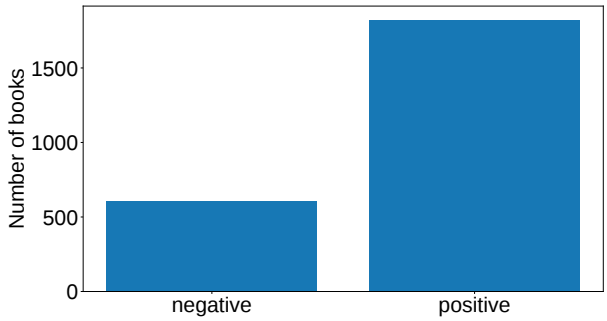

Figure 4.15: Number of books per textblob labels

At the end of our sentiment analysis we have integrated features about the emotion and the mood transmitted by the book. This are usable both to make the user visualize them in the VR environment but also, as we will see, as features for recommender models.

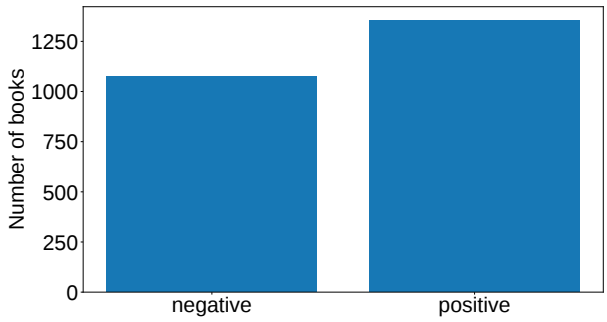

Figure 4.16: Number of books per VADER labels

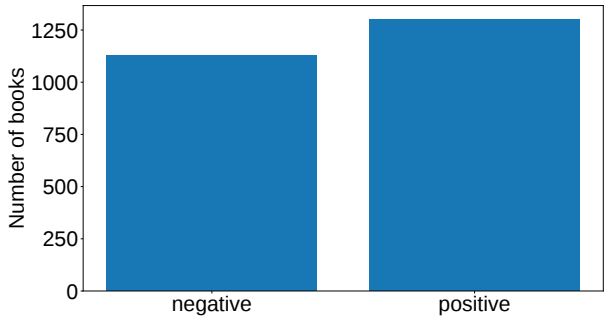

Figure 4.17: Number of books per majority labels

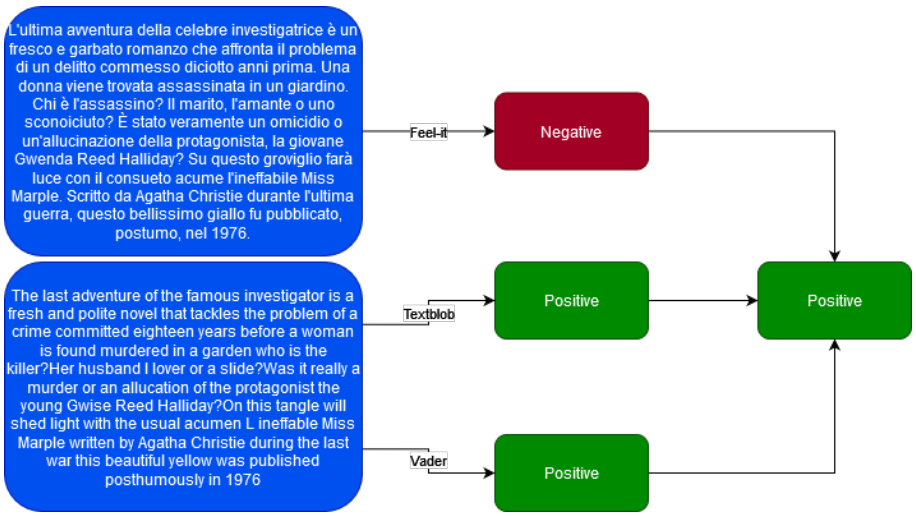

Figure 4.18: Diagram explaining the majority voting

Chapter 5

Recommender models & Observed results

5.1 Recommender models

In this section we will tune our recommender models to make it work with our data and features. Our objective in this section is to output suggestions for users by making good use of the interactions dataset and the features we have formed in the preceding sections.
Before starting to talk about the particular models, there are some points to cover:

- Interactions (rating and loans) management.
- Training and test splitting.
- Evaluation metrics.

Interactions, as we have seen in the data characterization chapter, have different characteristics if they come from Anobii with respect to BCT. In fact, in the first case, the ratings are both implicit (represented by the zero-valued votes) or explicit. However, BCT interactions are represented by loans that are implicit. One of the key problems to solve was how to use both types of interactions although they are inherently diverse in nature. The possible solutions were either:

- Consider BCT loans as rated 4 which is based by the supposition that a user that asks for a book also likes it.
- Consider Anobii ratings as implicit by comparing them to loans. This causes the fact that all positive ratings will be represented with the same strength.

However, considering a rating of 4 for BCT loans would be a biased choice so we decided to consider Anobii rating as implicit. Admittedly, this lead to a certain information loss but also gives us the guarantee that the interactions are genuine. In particular, ratings from 1 to 2 will be considered negative interactions while ratings from 3 to 5 will be considered positive interactions. The interaction dataset was filtered following these rules:

- Only ratings and loans referring to book of the filtered book dataset were kept.
- Interactions refer to users with at least 10 loans/ratings. This was done in order to avoid the 'cold start problem'.
- The dataset splitting in train, validation and testing kept intact the percentages of ratings and loans per user. In particular, the train dataset contains, per user, the 64% of the loans and ratings, the validation dataset contains 16% of them and the test dataset 20%. By doing this, we are considering the capacity of the recommenders to output a suggestion for a defined set of users with a moderate or richful history of books and, thus, the meaningfulness of their suggestions.

For evaluation, we will consider the metrics explained in table 5.2.

Metric name	Description
Average of hit users	Measures the average number of users that have a recommendation given by a model trained on the train dataset that is contained in test data.
Average of hit recommendations	Measures the average number of recommendations that are also contained in the test dataset.
Average rank	Measures the average of all the positions of the recommendations with respect to the scoring list.

Table 5.1: Metrics used for the recommenders

In 5.2, the metrics used to evaluate the recommenders are listed. They are based on the concept of "relevant recommendations": a recommendation is relevant when the book associated to it is contained in the test interaction dataset. This means that the recommender was capable to predict that the user would have liked that book and is performing well.

- Average of hit users - This gives a value representative of how many users were surely satisfied by the recommendation. It is calculated as:

$$u_{hit} = \frac{u_{rel}}{u_{tot}} \tag{5.1}$$

  Where u_{hit} represents the average of hit users, u_{rel} is the number of users with a relevant recommendation and u_{tot} represents the total number of users.

- Average of hit recommendations - This gives a value representative of how many relevant recommendations were given in total. It is calculated as:

$$r_{hit} = \frac{r_{rel}}{u_{tot}} \tag{5.2}$$

  Where r_{hit} represents the average of hit recommendations, r_{rel} is the number of relevant recommendations and u_{tot} is the total number of users.

- Average rank - It is an average of the ranks of the best recommendations given to each users. The best recommendation is considered the one highest on the sorting list of the recommender that is also in the test dataset interaction. It is calculated as:

$$rank_{avg} = \frac{rank_{min}}{u_{tot}} \tag{5.3}$$

  Where $rank_{avg}$ is the average rank, $rank_{min}$ represents the minimum rank between the given good recommendations and u_{tot} represents the total number of users.

To compare recommenders in a fair way, each recommender gives 20 suggestions in output and are trained (when trainable) with the same training dataset.
Our objective is to make a recommender able to satisfy (in terms of average number of hit users) more than 20% of our considered user population in order to make it usable by the BCT environment. To better characterize the interactions, we have:

- A population for training of 57054 users.
- 557633 train interactions.
- 197799 test interactions.
- 138847 evaluation interactions.

In the following subsections the model are explained and commented while in chapter 5 the results will be showed.

5.1.1 Item-item similarity model

As already talked in 2.3.1, content based models are recommenders that focus on the interactions of the user without considering other users. This means that the suggestions will be more explainable with a "We suggest this item because you interacted with this other item" formula but the suggestions tend to also be not serendipitous and may limit the user to just a restrict set of items. However, the possibility to allow the user to know why that book was suggested to him was an intriguing scenario for the BCT environment and a content based model was implemented.
The content based recommender is a simple model that outputs suggestions on the fly without need for training by considering the following features:

- The book description, transformed in a vector embedding by the sentence-transformers library.
- A genre vector of 20 values, formed by considering the presence of genres associated to the book. Considering that the genres were rated by the community, we decided to create a vector containing the percentage of the genre score with respect to the total score. By doing this, we not only know the presence of a genre like a one hot encoding pattern but also its strength.
- A mood and a sentiment vector. This was formed by considering a one hot vector of dimension 2 (to indicate positiveness and negativeness) and a one hot vector of dimension 4 (to indicate fear, anger, joy or sadness).
- The title of the book, transformed into a vector in the same fashion of the description.
- The keywords extracted in the preceding subsections of this chapter. They were transformed in a string and then treated like the books descriptions.

The implementation was done by considering item-item similarities (see 2.3.1): the algorithm considers, at random, a book read by the user and transform it into an information vector V with the already mentioned features.
With this transformation, we can create a similarity matrix M of shape (number of books, number of books) by performing a dot product with the vector book V and its transpose.

$$M = V \cdot V^T \tag{5.4}$$

The matrix will contain, for each cell, a float value indicating how much, in terms of the utilized features, the row indexed book is similar to the column indexed book.

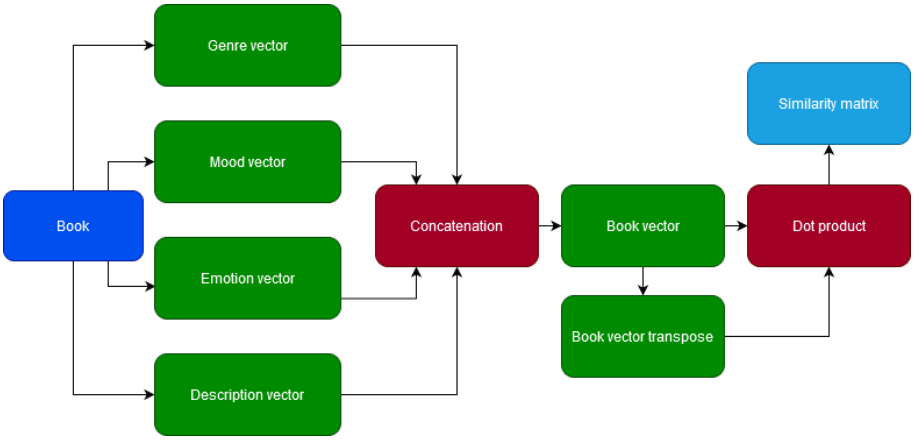

Figure 5.1: Item-item Content based diagram

Then, we can consider a book read by the user and suggest a recommendation by taking the one with the highest similarity value. In particular, in order to obtain 20 books, we take a random book from the user interaction history and make the suggestion; then, in a loop we continue to suggest this way until we reach 20 books (being aware to not give overlapping recommendations).

```
Please, input a user for a content based recommendation.
 156880
Because you read vieni via con me, we suggest you ["l'inattesa piega degli eventi", 'crimini italiani', 'madre notte', 'la neve era sporca']
Because you read sonderkommando auschwitz, we suggest you ['il farmacista di auschwitz', 'madre notte']
Because you read la fine è il mio inizio, we suggest you ["l'arte di jiro taniguchi", 'forza del male']
Because you read se non ora, quando?, we suggest you ['ogni promessa', 'il padrone della città', "l'inattesa piega degli eventi"]
Because you read la parola contro la camorra, we suggest you ['memoria del vuoto', 'troppo umana speranza']
Because you read soldati di salamina, we suggest you ['storia di roma', 'la caccia al tesoro', 'la figlia del podestà']
Because you read memoria e identità, we suggest you ['la cantina', 'memoria del vuoto', 'una storia quasi soltanto mia']
```

Figure 5.2: Example of content based suggestion for a user.

As we can see from 5.2, with this system it is possible to make the user aware of why something was suggested to him (something that can be hard to do with collaborative filtering) making the recommendation more rich. However, by using this approach, recommendations will depend solely on a particular read book: this means that not only user with few books may find the recommendations a bit constrained but we are not finding underlying reading patterns given that the similarity is done on a book-book fashion.

5.1.2 ALS and BPR models

The ALS and BPR models are collaborative filtering models built around the concept of latent factors (see 2.3.2). In that section, we noted that a collaborative filtering may be neighborhood or latent factor based. Being latent factor based

means that these models are trained by considering the user-book interactions as a set with a user-defined number of latent parameters.

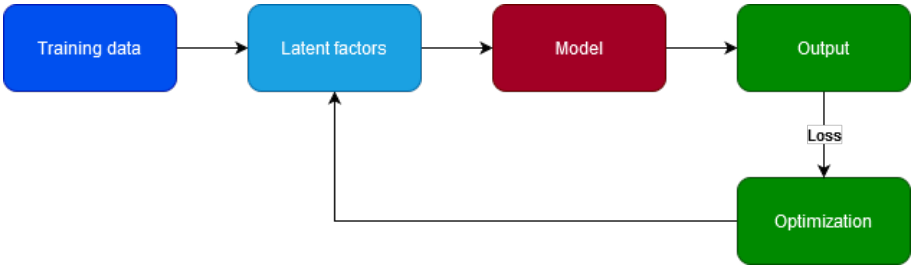

Figure 5.3: Diagram showing a standard latent factor model

As we can see from 5.3, these type of models are based on an iterative optimization of a loss value until convergence to make the latent parameters work better with our model that, by consequence, will output better results.
ALS is an algorithm explained in [22]: it is a collaborative filtering algorithm useful for sparse matrices (like our use case and, more generally, user-item interaction matrices) that consists in optimizing the loss calculated on a score (like a matrix factorization user-item affinity) by considering two steps:

- We freeze the user latent factors in order to optimize the item latent factors.
- Then, we freeze the item latent factors in order to optimize the other.

In ALS, the used loss function is composed by two equations :

$$x_u = (Y^T Y)^{-1} Y^T p(u) \tag{5.5}$$

$$y_i = (X^T X)^{-1} X^T p(i) \tag{5.6}$$

X and Y are, respectively, the user and item vectors while p(u) and p(i) represent the goodness of the interactions (equals to 1 if the interaction is an observed one or 0 for unobserved ones). Finally, xu and yi are the new user and item vectors. Then, the optimization continues until the value of the latent factors (represented by the user and item vectors) stabilizes. This means that, by applying the first equation we are creating a new user vector that gives higher scores to observed interactions and then, by applying the second equation we do the same thing with the item vector.
To build the ALS model I used the 'implicit' library, a library used to build implicit feedback models.
In the case of BPR (whose paper is cited in 2.3.2), we have decided to build this model due to its adaptability for our use case. In fact, this model's advantages are fitting with our environment:

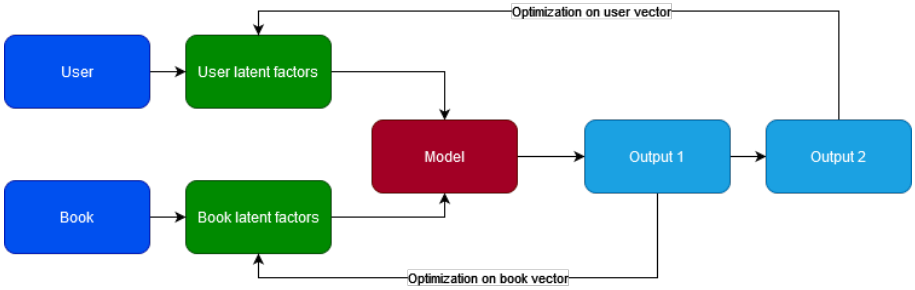

Figure 5.4: Diagram showing the ALS model working

- As seen in the paper, the model works natively and good with implicit feedbacks.

- The model outputs a ranked list directly, allowing to have a tailored suggestion set per user in a simple way.

With this algorithm, the optimization is done on user-item pairs interactions instead of singular user-item ones to optimize not just the interactions but the relation in terms of user preference between couple of items used for ranking. In particular, the optimization in BPR happens by finding the set of parameters θ that maximize the probability that a user prefers a item he interacted with against an item with which he has not interacted.

$$p(i_u j|\theta) = \sigma(score_i(\theta) - score_j(\theta)) \tag{5.7}$$

In the formula, $p(i_u j|\theta)$ indicates the posterior probability of user u preferring item i over item j depending on the set of parameters θ while σ represents the sigmoid function. The sigmoid has in input the difference between $score_i(\theta)$ which is the calculated score of item i dependent on the set of θ and $score_i(\theta)$ which is the calculated score of item i dependent on the set of θ.
Through this optimization, we are finding the latent factors that best show the difference in scores between a preferred item and a less preferred item related to a user for all possible pairs of items, thus creating a ranking list.

To build the BPR model I used an implementation reference coded by Dr. Greta Vallero with the 'ALS' library and then implemented my own version with 'implicit'. Being a collaborative filtering models, we can not explain why some suggestions were given to the user: however, by capturing reading patterns of similar users, recommendations may contain serendipitous suggestions that may increase the user's choice of different books.

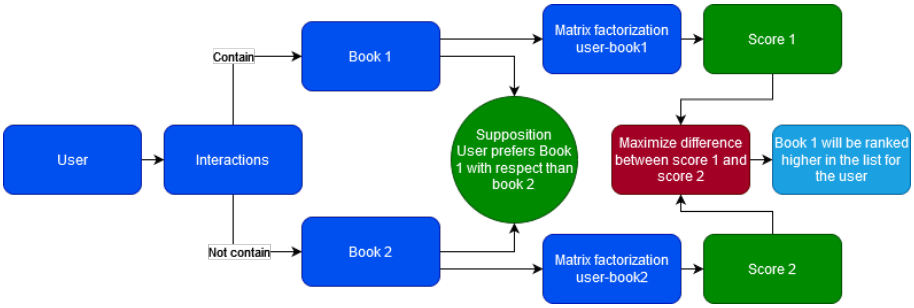

Figure 5.5: Diagram representing the BPR steps for a single user

5.1.3 Retrieval model

The retrieval model is a model used to retrieve, with a fast serving, a variable number of recommendations depending on a user-decided K input. It is a model offered by 'Tensorflow-Recommender' (TFRS), a library part of the Tensorflow framework.

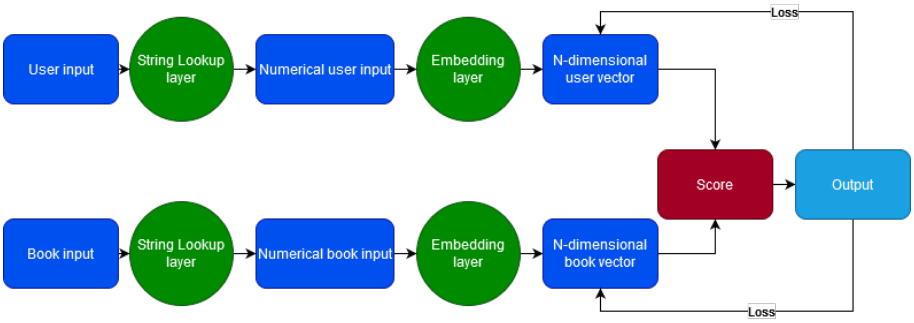

Figure 5.6: Diagram of the TFRS retrieval model

The model takes two inputs: a user vector and a item vector. Due to its shape (which we can see in 5.6), it is also called 'two tower' model. This is because the user and the item input are first processed in parallel and are then concatenated into a single vector with which a score is computed. Said score may be computed with traditional means (for example matrix factorization) or with deep learning layers (like in [19]). Through the use of a "string lookup" layer, we can convert the user and book identifiers to numerical values that are then processed by an embedding layer. The embedding layer allows to represent users and books in a N-dimensional space by converting them into vectors of dimension N. These embedding vectors are trained during the learning phase of the model in order to shorten the distance in the N-space of users and books with high affinity, creating clusters in the N-space. By doing so, during the calculation of the score, highest

scores for a user will be the ones of books that are most liked by similar users. This is powerful because to serve a recommendation to a user, the model can just evaluate the affinity of the user and the books by looking at the positions of the inputs in the trained embedded space and picking the most close to the considered user vector. This is a very efficient operation that is performed by using 'serving layers', as explained in diagram 5.7.

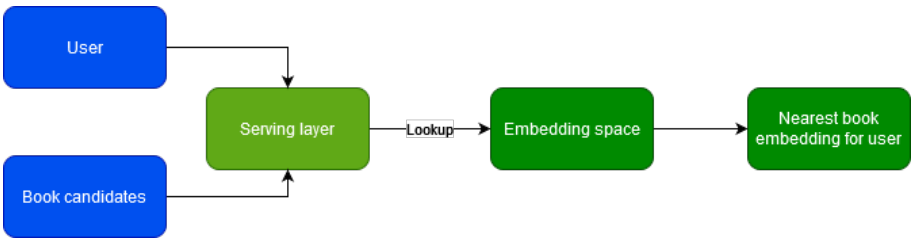

Figure 5.7: Diagram of TFRS serving layer

For training, the model uses cross entropy loss. This means that, for each interaction, it outputs a value in a range between 0 and 1 indicating the probability that the user-item interaction is pleasant. Then, looking at the ground truth (1 for scores related to observed interactions and 0 for scores related to unobserved interactions), it calculates the loss with the formula:

$$L = -\frac{1}{2}(y^{(1)}_{true} \cdot log(y^{(1)}_{predicted}) + (1 - y^{(2)}_{true}) \cdot log(y^{(2)}_{predicted})) \tag{5.8}$$

where $y^{(1)}_{true}$ and $y^{(1)}_{true}$ are the true labels (observed or unobserved interaction) and $y^{(1)}_{predicted}$ and $y^{(2)}_{predicted}$ are the predicted labels by the model. The loss is then backpropagated to the embedding layers in order to minimize the loss.
To try to exploit the potential of the retrieval model to look up embeddings in an N-space and choose the best candidate, we can use dedicated serving layers offered by the TFRS library (layers used to retrieve the books per user) that act as a filter, giving us the top 20 books with a certain genre. By looking at 5.8, in fact, we can notice that the users tends to prefer (in terms of interactions) certain genres over others. So, inferring the user preference in the serving layers may prove fruitful.

Although the dataset is not massive (considering that the books are in the order of thousands), it should be interesting to see the performance of this model: Tensorflow recommender serving layers are optimized to offer good efficiency even with bigger datasets so it provides a scalable model even for future and bigger book datasets.

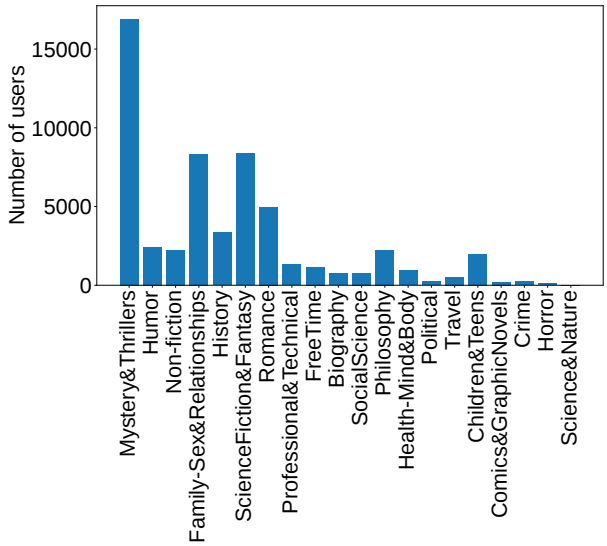

Figure 5.8: Favourite genres of users in terms of interactions

5.2 Observed results

Metrics results			
Model	Average hit users	Average hit recommendations	Average rank
ALS	0.15	0.18	200
BPR	0.26	0.35	124
Retrieval normal	0.168	0.20	327
Retrieval custom	0.18	0.21	327
Content based	0.12	0.155	-

Table 5.2: Metrics values for the considered models.

5.2.1 Item-item similarity model

The item-item similarity model explained in chapter 4 serves two purposes:

- Assess the influence of the features (extracted and not) on the average hit users (calculated with the metric explained in 5.1).
- Give to the readers of BCT a way to have an explainable recommendation, calculated directly from the user books log.

As we already explained in 5.1.1, the model gives recommendations on the fly and is non-trainable because it performs a look-up on an already calculated similarity matrix. So, we made the model output book-based suggestions on all the test population and calculated the metrics. In particular, to calculate the similarity matrix, we considered one feature at a time to assess the impact of the feature in the quality of the suggestion. However, calculating the average rank for this particular model was not considered useful due to the fact that the model outputs suggestions using item-item affinities and not user-item like expected, for example, in collaborative filtering models.

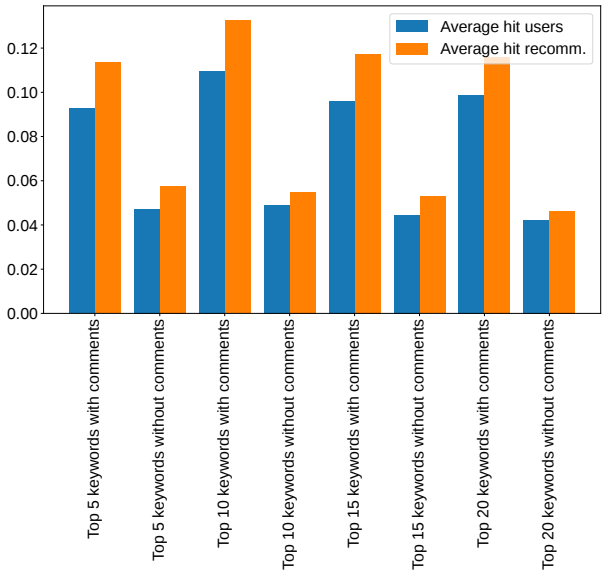

Figure 5.9: Influence of keywords number on the hit users and hit recommendations metrics

In 5.9, we can see the influence on the metrics of the number of keywords considered per book: it is notable how, overall, the top 10 keywords with concatenated comments are giving the best results and thus they were choosed during the calculation of the similarity matrix with all features.
As desired, using all the features (moods, emotions, titles, description, genres and keywords) for the book-book similarity calculation leads to the best results in terms of metrics (see 5.10), resulting in more than a tenth of the dataset that is surely satisfied by the suggestions. This is good because it means that, by using data features, we can offer to readers suggestions that come directly from their tastes and, thus, making recommendations richer.
On 5.11 we can see the genre of the books that were good recommendations

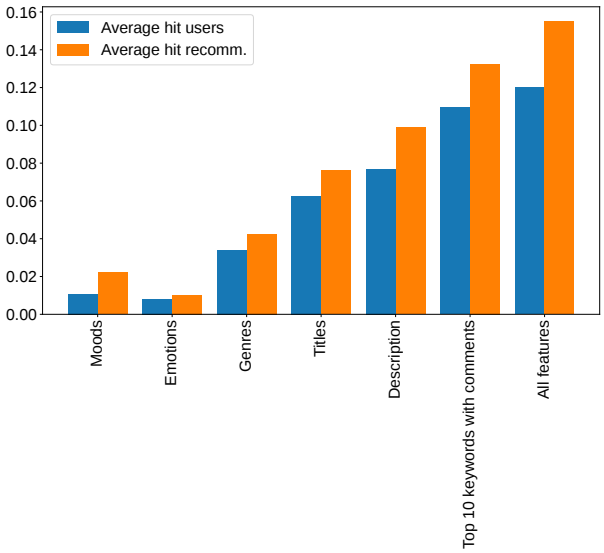

Figure 5.10: Influence of features on the hit users and hit recommendations metrics

weighted by the occurrences of the genres in the book dataset: interestingly, we can see that horror books were often good recommended, probably because horror readers tends to read similar books in terms of content.

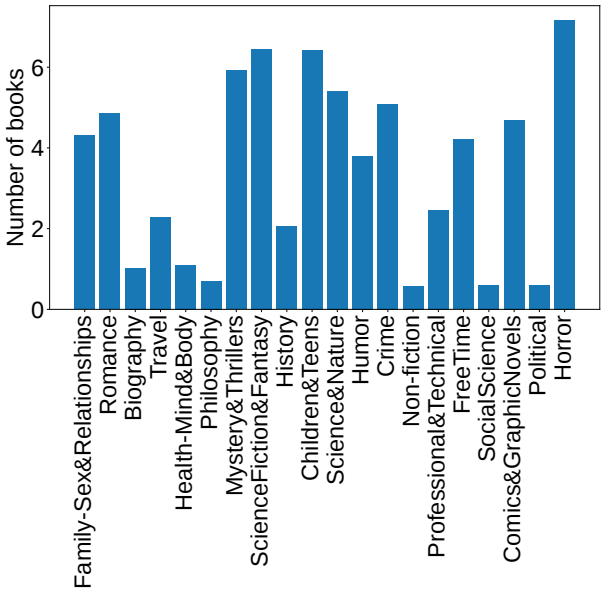

Figure 5.11: Number of good recommended genres in content based model

5.2.2 ALS and BPR models

As seen in 5.1.2, these models were used for their pure implicit feedback adaptability. Contrarily to the item-item similarity model, these models required training due to the need of latent factors to be iteratively optimized and, thus, different number of latent factors representing users and items were considered to assess how much that number affects the metrics. In particular, we considered the metrics with both the ALS and BPR model with 300 iterations and 10 possible values of the latent factors number: 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100.

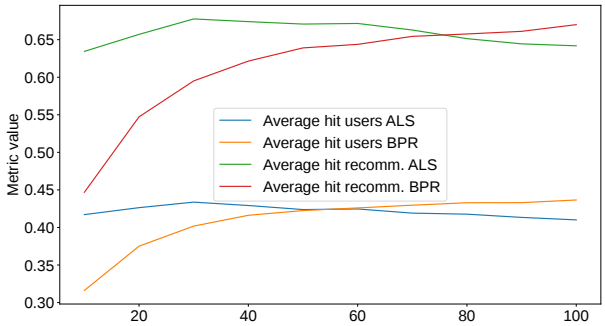

Figure 5.12: Average hit users and average hit recommendations during evaluation in ALS and BPR models

As we can see from 5.12, the ALS model performed better during evaluation with less latent factors while the BPR one performed better by setting an higher number of latent factors. Moreover, by looking at the average rank figure (5.13), we can note that the ALS model initially has a relatively low average rank and comparable to the BPR one but increasing the number of latent factors worsens the average rank while, in the case of BPR, it remained more or less constant.

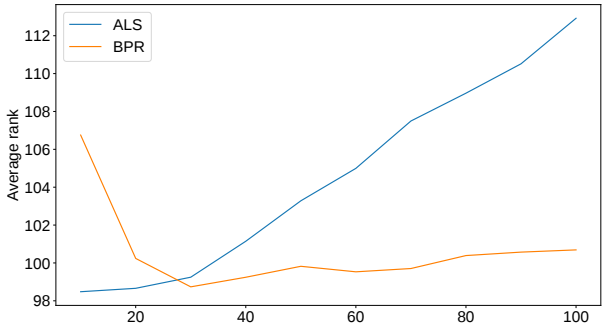

Figure 5.13: Average rank in ALS and BPR models

We decided to test the BPR model with 100 latent factors (which performed the best with respect to average hit users and average hit recommendations) and the ALS model with 30 latent factors (which performed the best in terms of average hit users with respect to other ALS configurations).

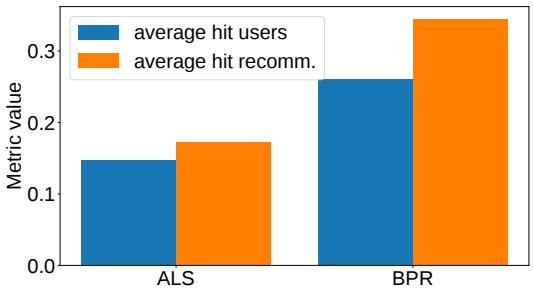

Figure 5.14: Test performances of the ALS and BPR models

As we can see in 5.14, evaluation results are neatly transported onto the test dataset: the BPR model performs better than the ALS one, surpassing our objective to have more than 0.20 with the hit users metric. Looking at the number of good recommended books per genre (weighted by the occurrence of the genre in the book dataset) in 5.15, we can note that BPR fares better than ALS in almost every genre out of the 20.

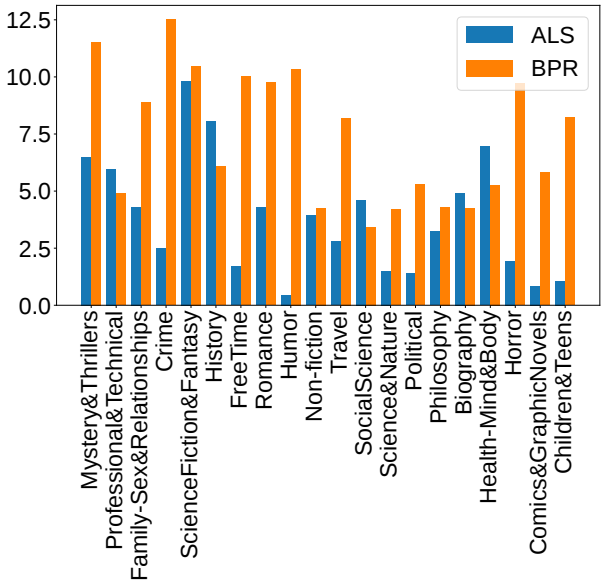

Figure 5.15: Genres of good recommended books in ALS and BPR.

From 5.16 we can see that the average rank also remained almost equal during test, giving better average rank with the BPR model.
The results of the BPR and ALS models during test can be observed in table 5.2.

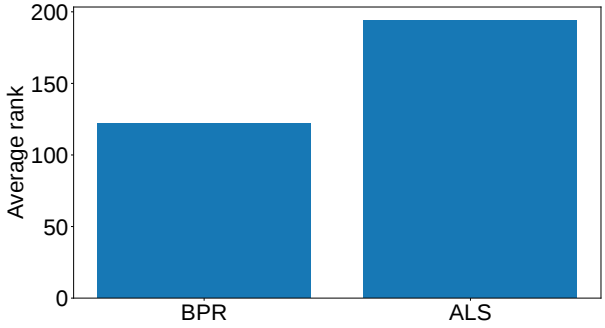

Figure 5.16: Average rank ALS and BPR during test

5.2.3 Retrieval model

In this model, we can decide the size of the embeddings that represents users and books. The loss which allows the training is the cross entropy, as explained in 5.1.3. Figure 5.17 shows the value of the loss with different sizes for the embedding vector.

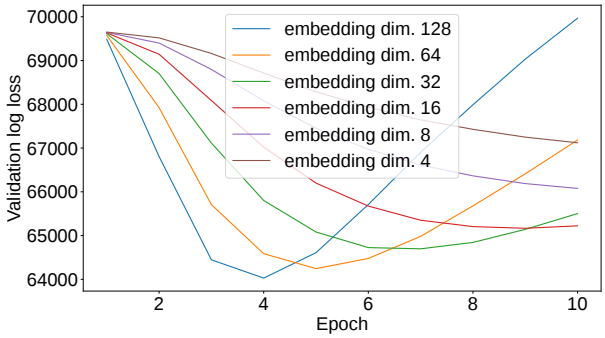

Figure 5.17: Validation losses in 10 epochs with different embedding sizes.

As we can see, the validation loss is at its minimum with an embedding vector of 128 elements: the fact that the loss tends to increase after a few epochs means that the model is overfitting very fast to the train dataset. On the test dataset, we use the configuration of embedding vectors of 128 dimensions and 4 epochs which provides the best performances.
Then, we tried to consider the favourite genres of a user during the recommendation

servings (like explained in 5.1.3). This was done by suggesting the top 10 books of the inferred favourite genre (in terms of interactions) and the top 10 books without the genre constraints. As we can see in 5.2, although the retrieval models are not the best, the custom servings were a good choice, allowing us to increment the metrics value. This means that considering user inferred features extracted from the interactions may lead to even better models.
On 5.18 and 5.19 we can also see which genre were the hit books.

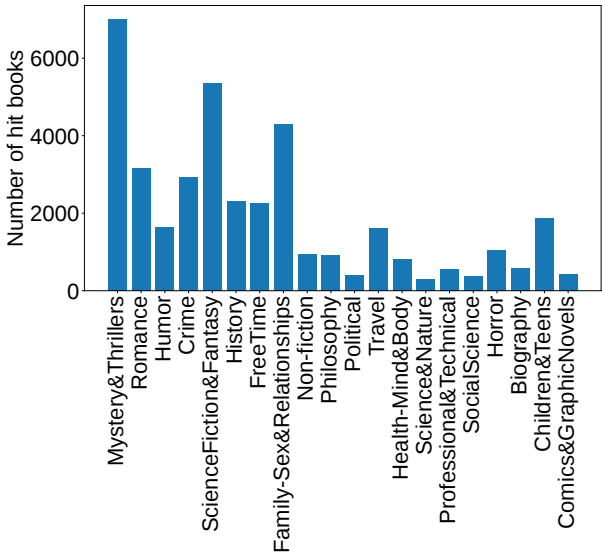

Figure 5.18: Number of good recommended book with the retrieval model.

We can notice, as expected, that serving the top 10 recommendations of the favoured genre for the user boosted the number of good recommended books with the most frequent genres while good recommended books with less frequent genres were decreased in numbers.
The results of these models during test can be observed in 5.2.

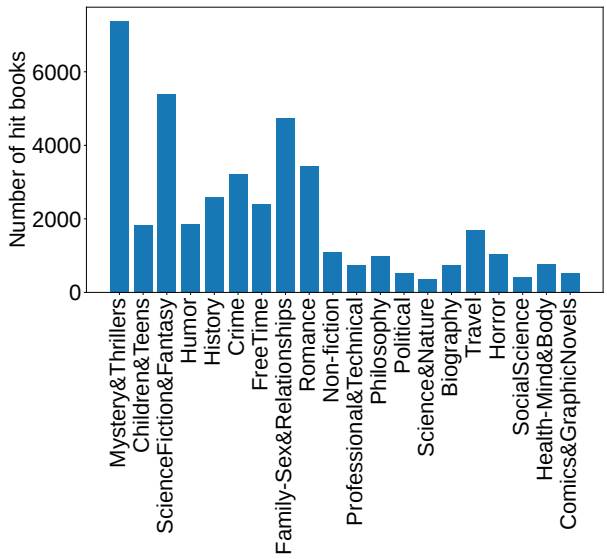

Figure 5.19: Number of good recommended book with the retrieval model with favoured genres.

Chapter 6

Conclusion

This thesis was focused on build recommender models in order to support readers of the public libraries in Turin by helping them nurturing their passion through the discovery of new books.
Moreover, through feature extraction we not only could help the building of the recommender systems but also to help the users to visualize informations about the book in a clearer way during the VR experience.
Trying various recommender models on our book dataset gave us various useful insights.
As we can see in 5.2, the BPR model worked the best out of the tried ones. This means that the readers in the dataset had defined reading patterns depending on certain stereotypes. This enforces the fact that there are common paths of read books characterized by social interactions, proving the point of the 'Reading&Machine' project.
Although the other models gave worse performances, we must not undervaluate the fact that both gave acceptable results for two different reasons:

- The content based model shows that the native book features and the extracted book features are both usable to create a model capable of suggesting books in a punctual and explainable manner.
- The retrieval model shows that using features based on user interactions improves the quality of the recommendations.

By looking at the good performances of the collaborative filtering model it is clear that reading may be seen not as a individual hobby but also as a social experience. Readers are interested in what other similar users read and can be pleasantly satisfied with suggestions based on them.
However, it is also true that users often wants suggestions based on their own readings, without being considered part of a group. The fact that a relatively

simple content based model resulted in a functional, working system represents the potential of features in a book dataset environment: by using the data at our disposal we can extract features that are able to capture the user's interests directly from their reading logs. This means that we are implementing a model where recommendations are usable by the users to understand how to find books similar to one he really likes and, as such, may offer a good experience in the library environment.
By using these recommenders, the population of readers can find books that they like without being overwhelmed by choosing between a set of thousands of millions of books. Moreover, the visualization of the book features (keywords, genres, overall tone of the book, emotion expressed) in the VR environment help the user to judge his affinity with a book even without using the recommender system, giving life to an environment where readers may always discover new books with which to nurture their passions.

6.1 Future works

The thesis may provide a basis to continue working on improving recommender models for book suggestions.
In the future, by using temporal features (like the date in which a rating was given or when a loans was started), it would be possible to allow richer recommendations by considering the change in tastes and the decay of interests that naturally occur during a person's life. By making suggestions considering newer interests we are allowing the users to discover books that are fitting to their current interests and, thus, the recommendation itself will be more impactful.
The fact that not only book features can be used for improving recommender models is also something to be considered. Features related to users that allow the recommender to detect similar users by using demographic informations (for example, the age of the reader) may improve the performances of collaborative filtering model. Moreover, by using external informations, like the most trending topics of the moment, recommenders can make suggestions based not only on the individual user interests but also on important and current themes that may broad the user's interests and culture.
It is also to be noted that, while certain models worked better than others, even less performing models managed to obtain good results in recommending certain books. Thus, it would be interesting to assess the effect in terms of performances of combining different types of recommenders together.

Appendix A

Source code

The code written for this thesis can be find on the following link.

`https://github.com/AleSpex/Reading_and_Machine.git`

Bibliography

[1] Sheena Iyengar and Mark Lepper. «When Choice is Demotivating: Can One Desire Too Much of a Good Thing?» In: *Journal of personality and social psychology* 79 (Jan. 2001), pp. 995–1006. DOI: 10.1037/0022-3514.79.6.995 (cit. on p. 1).

[2] www.theatlantic.com. «Google: There Are Exactly 129,864,880 Books in the World». In: (). URL: https://www.theatlantic.com/technology/archive/2010/08/google-there-are-exactly-129-864-880-books-in-the-world/61024/ (cit. on p. 2).

[3] Dietmar Jannach and Michael Jugovac. «Measuring the Business Value of Recommender Systems». In: *ACM Transactions on Management Information Systems* 10.4 (Dec. 2019), pp. 1–23. DOI: 10.1145/3370082. URL: https://doi.org/10.1145%2F3370082 (cit. on p. 2).

[4] ayşe Sirel. «READING SUSTAINABLE ARCHITECTURE VIA THE 'HELSINKI| OODI LIBRARY CONSTRUCTION TECHNOLOGY'». In: Apr. 2021 (cit. on p. 6).

[5] helsinkitimes. «Oodi's artificial intelligence app Obotti finds reading material for customers». In: (). URL: https://www.helsinkitimes.fi/themes/themes/science-and-technology/19426-oodi-s-artificial-intelligence-app-obotti-finds-reading-material-for-customers.html (cit. on p. 6).

[6] Anobii. «Anobii, a reading-based social network». In: (). URL: https://www.anobii.com (cit. on p. 7).

[7] Muh-Chyun Tang, Yi-Jin Sie, and Pei-Hang Ting. «Evaluating books finding tools on social media: A case study of aNobii». In: *Information Processing & Management* 50.1 (2014), pp. 54–68. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2013.07.005. URL: https://www.sciencedirect.com/science/article/pii/S0306457313000770 (cit. on p. 7).

[8] H M Hasan, Falguni Sanyal, Dipankar Chaki, and Md Ali. «An empirical study of important keyword extraction techniques from documents». In: Dec. 2017. DOI: 10.1109/ICISIM.2017.8122154 (cit. on p. 8).

[9] Wengen Li and Jiabao Zhao. «TextRank Algorithm by Exploiting Wikipedia for Short Text Keywords Extraction». In: *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*. 2016, pp. 683–686. DOI: 10.1109/ICISCE.2016.151 (cit. on p. 8).

[10] Divya Khyani and Siddhartha B S. «An Interpretation of Lemmatization and Stemming in Natural Language Processing». In: *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology* 22 (Jan. 2021), pp. 350–357 (cit. on p. 9).

[11] Nhan Cach Dang, Marìa N. Moreno-Garcìa, and Fernando De la Prieta. «Sentiment Analysis Based on Deep Learning: A Comparative Study». In: *Electronics* 9.3 (Mar. 2020), p. 483. DOI: 10.3390/electronics9030483. URL: https://doi.org/10.3390%2Felectronics9030483 (cit. on p. 9).

[12] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. «Lexicon-Based Methods for Sentiment Analysis». In: *Computational Linguistics* 37.2 (June 2011), pp. 267–307. ISSN: 0891-2017. DOI: 10.1162/COLI_a_00049. eprint: https://direct.mit.edu/coli/article-pdf/37/2/267/1798865/coli_a_00049.pdf. URL: https://doi.org/10.1162/COLI%5C_a%5C_00049 (cit. on p. 9).

[13] Federico Bianchi, Debora Nozza, and Dirk Hovy. «"FEEL-IT: Emotion and Sentiment Classification for the Italian Language"». In: *Proceedings of the 11th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, 2021 (cit. on p. 10).

[14] Charilaos Zisopoulos, Savvas Karagiannidis, Georgios Demirtsoglou, and Stefanos Antaris. «Content-Based Recommendation Systems». In: (Nov. 2008) (cit. on pp. 10, 11).

[15] Jon Herlocker, Joseph Konstan, Al Borchers, and John Riedl. «An Algorithmic Framework for Performing Collaborative Filtering». In: *ACM SIGIR Forum* 51 (Aug. 2017), pp. 227–234. DOI: 10.1145/3130348.3130372 (cit. on p. 11).

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. «Collaborative Filtering for Implicit Feedback Datasets». In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22 (cit. on p. 11).

[17] Muh Hanafi, N. Suryana, Abdul Bin, and Hasan Bashari. «Paper survey and example of collaborative filtering implementation in recommender system». In: *Journal of Theoretical and Applied Information Technology* 3195 (Aug. 2017) (cit. on p. 11).

[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. «BPR: Bayesian Personalized Ranking from Implicit Feedback». In: UAI '09. Montreal, Quebec, Canada: AUAI Press, 2009, pp. 452–461. ISBN: 9780974903958 (cit. on pp. 12, 13).

[19] Charilaos Zisopoulos, Savvas Karagiannidis, Georgios Demirtsoglou, and Stefanos Antaris. «Content-Based Recommendation Systems». In: (Nov. 2008) (cit. on pp. 13, 14, 57).

[20] «SpaCy, an extensive NLP framework». In: (). URL: `https://spacy.io/` (cit. on p. 38).

[21] «COO Matrix documentation». In: (). URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo_matrix.html` (cit. on p. 39).

[22] Gábor Takács and Domonkos Tikk. «Alternating least squares for personalized ranking». In: (Sept. 2012). DOI: `10.1145/2365952.2365972` (cit. on p. 55).