UCLouvain

LOUVAIN
School of Management

Louvain School of Management

Enriching recommender system by joining multiple sources of unstructured dataset : a wine recommender use-case

Auteur: Maxime Dillion
Promoteur: Corentin Vande Kerckhove
Année académique 2021-2022
Travail de fin d'études (TFE) en vue d'obtenir le titre de
Master (60) en Sciences de Gestion
Horaire de jour

Abstract

Over the past few years, we observed the expansion of the use of recommendation systems in many fields. Even though wine recommendation did not escape this rule, researches for this use-case are limited, due in particular to the lack of comprehensive open datasets on this topic. In this work, we approach this problem by merging two datasets, one containing user reviews and the other containing wine descriptions, in order to evaluate how it affects the performances of wine recommendation systems. To be able to make this evaluation, we designed a wine recommender system, and assessed its accuracy for different parameters. Finally, we looked at how the results are affected, when the model is trained on the merged dataset, compared to when it is trained only on the dataset containing reviews.

Au cours des dernières années, nous avons pu observer une expansion dans l'utilisation de systèmes de recommandation dans de nombreux domaines. Même si le sujet de la recommandation de vin n'a pas échappé à cette règle, les recherches sur le sujet sont limitées, notamment en raison du manque d'ensembles de données ouverts compréhensifs. Dans ce travail, nous abordons ce problème en fusionnant deux jeux de données, l'un contenant des avis d'utilisateurs, et l'autre contenant des descriptions de vins, afin d'évaluer comment cela affecte les performances de systèmes de recommandation de vin. Pour pouvoir faire cette évaluation, nous avons conçu un système de recommandation de vin, et évalué sa précision pour différents paramètres. Enfin, nous avons examiné comment les résultats sont affectés, lorsque le modèle est entrainé sur le jeu de données fusionné, par rapport à lorsqu'il est entrainé uniquement sur le jeu de données contenant des avis.

Acknowledgements

First of all, I would like to thank my promoter, Mr. Corentin Vande Kerckhove, for his various advice, his availability, his constructive feedbacks, and more generally his knowledge sharing on a subject that I did not know.

I would also like to thank my family and friends, for their proofreading and support throughout the completion of this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advent of Big Data and the emergence of large scale online platforms such as Netflix, Amazon or YouTube, we have seen a significant increase for needs of efficient user recommendations during the last two decades. In fact, providing efficient and relevant recommendations for users can make a non-negligible difference in the time spent by users or the number of sales on those platforms (Zhou, Khemmarat, & Gao, 2010).
Recommender systems (or recommendation systems or RSs) can be defined as "software tools and techniques that provide suggestions for items that are most likely of interest to a particular user", such as defined in (Ricci, Rokach, & Shapira, 2015). RSs can be applied in a large range of applications for different goals. Popular RSs topics includes series, films, and videos recommendations on platforms such as Netflix or YouTube, or goods recommendation on Amazon.
An important issue when creating a recommender system can be the lack of clean, important enough datasets in order to train the recommender. This problem is especially present for wine recommendation due to the large number of wines in existence and the relatively small number of reviews available for them. Moreover, the available open datasets[1] are often not in the same formats and not centralized, leaving us with multiple scattered datasets from multiple sources.
This work will start by doing a literature review of the field of RSs. Then, we will continue by exploring wine open datasets and create a merged dataset from two distinct datasets found online.
We will continue by evaluating the performances of wine recommender system with different parameters.
Finally, we will compare performances of a model trained on this merged dataset to the ones of a model trained only using one dataset.

[1]Data that can be freely used, re-used and redistributed by anyone – subject only, at most, to the requirement to attribute and sharealike ("What is open Data?", n.d,)

Chapter 2

Theory and literature review

In this chapter, we will start by defining the main concepts of RSs and their different types, with a focus on content-based RSs. Then, we will discuss different numerical and non-numerical evaluation methods commonly used to assess RSs performances. Afterwards, we will identify some major challenges in the field. Finally, we will explain the work's objectives and the question it aims at answering.

2.1 Definitions

We will start this chapter by defining several key notions in the field of RSs, that will be used throughout the entire work.

2.1.1 Item

Items are the objects that can be recommended to the users. They can be defined by a wide range of features[1]. For example, items on a platform such as YouTube are all the videos posted, and their features can be their authors, lengths and types (e.g., sport, travel, ...).
In this work, items will be the different wines that can be recommended to the users. As detailed in section 3.1.1, wines have several features such as their colors, wineries, ...

2.1.2 Ratings and transactions

A *rating* is the value of an item in the eyes of a specific user. Ratings generally come from *transactions*, which can be defined as different logged interaction between a user and the RS (they directly ensue from human-computer interactions). They are

[1] Set of characteristics of an item

then used, by the recommendation algorithm, to make recommendations (Ricci et al., 2015). There are two types of ratings : (Claypool, Le, Wased, & Brown, 2001)

Explicit

An explicit rating is a rating that is consciously given by the user. The user, often after interacting[2] with the item, will input feedback to the system. Transactions used for explicit ratings include, but are not restricted to : like and dislike buttons, numerical note attribution and ordinal note attribution.

Implicit

An implicit rating is a rating that is not expressly entered by the user and that is inferred by the system based on user interaction(s). Transactions used for implicit ratings include but are not restricted to : sentiment analysis from text, inference from mouse click and movement and inference from search history.

2.1.3 User-item matrix

The *user-item matrix* is a matrix of size n x m, where n is the number of users and m is the number of items in the system. Each entry $r_{u,i}$ of the matrix corresponds to the rating associated to user u to item i.Figure 2.1, taken from (Melville & Sindhwani, 2010), provides a visual representation of a user-item matrix :

		Items					
		1	*2*	...	*i*	...	*m*
	1	5	3		1	2	
	2		2				4
Users	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		
	a	3	5		?	1	

Figure 2.1: User-item matrix representation (Melville & Sindhwani, 2010)

[2]Interaction may be a purchase for an e-commerce site, watching a film for a streaming platform, ...

Note that the user-item matrix almost always has some missing entries, because it is very unlikely that every user will have ratings for every item. In some cases, this matrix may even be very sparse.

2.1.4 Recommender systems

Now that we have defined the important concepts of the RS field, we will define what is a recommender system and what are the different types of those systems. As defined in (Ricci et al., 2015) : "*Recommender systems* are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user". It means that RSs should predict items in which the different users using the system might be interested, which implies the need for a comparison (generally with the predicted rating defined below) within the different items. Figure 2.2, taken from (Khusro, Ali, & Ullah, 2016) shows the general model of a RS.

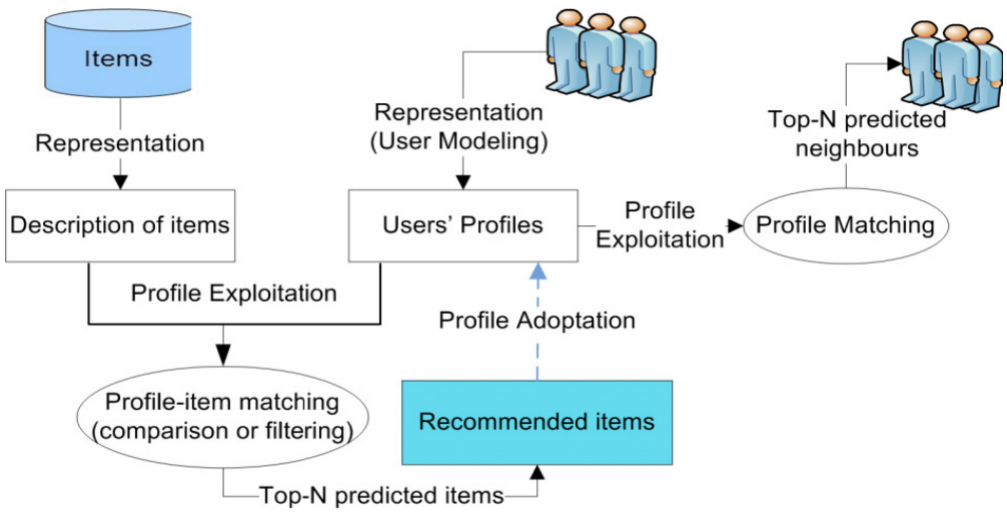

Figure 2.2: General model representation of a recommender system (Khusro et al., 2016)

More formally : if we define the set of user as U and the set of items as I, we can define the function $R(u,i), u \in U, i \in I$ as the rating of the user u for the item i (it corresponds to the entry $r_{u,i}$ in the user-item matrix). With the previous definitions, we can then define the functional task of an RS as a function $\hat{R}(u,i)$ where $\hat{R}(u,i)$ is an approximation of $R(u,i)$. Therefore, for a given active user u, we can compute $\hat{R}(u,i), \forall i \in I$. Then, the recommender system can recommend

items based on the predicted score (for example, the k largest scores predicted with k being the number of desired predictions) (Ricci et al., 2015).

There exists different types of RSs, each with their specificities. We will briefly describe them in the following subsections, with a greater focus on content-based recommender, which will be use later in this work.

Collaborative filtering

As defined in (Aggarwal, 2016), "*Collaborative filtering* (CF) models use the collaborative power of the ratings provided by multiple users to make recommendations". The main assumption for CF models is that there is a high correlation between ratings of similar users.
The main drawback of CF is that it generally does not perform well when the user item matrix is too sparse, and it can be hard to evaluate if a new (unrated) item should be recommended or not. However, the adaptive power of the model will increase over time as more and more ratings are collected.

Knowledge-Based

Knowledge-based recommender (KB) is defined in (Burke, 2000) as a RS that "uses knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements".
An advantage of KB systems is that they do not require a lot of user ratings and can generally be quite efficient even with highly sparse user-item matrices. The main drawback of this approach is that it often requires a large amount of user interaction prior to having items recommended, which can lead to less smooth experience for users.

Content-Based

As defined in (Ricci et al., 2015), "*content-based recommender systems* (CB) relies on item and user descriptions (content) to build item representations and user profiles to suggest items similar to those a target user already liked in the past".
CB systems take therefore the history of items for which the user interacted with in addition to the user-item matrix's rows of the user as input. Then, it will recommend to the user items similar to the ones it liked, and/or opposite items from the ones that it disliked.

We will go a bit deeper into the general architecture of CB recommenders and detail how we will be using this type of recommender in the context of wine

recommendation. Figure 2.3, taken from (Lops, Gemmis, & Semeraro, 2011), shows the general architecture of a CB recommender.

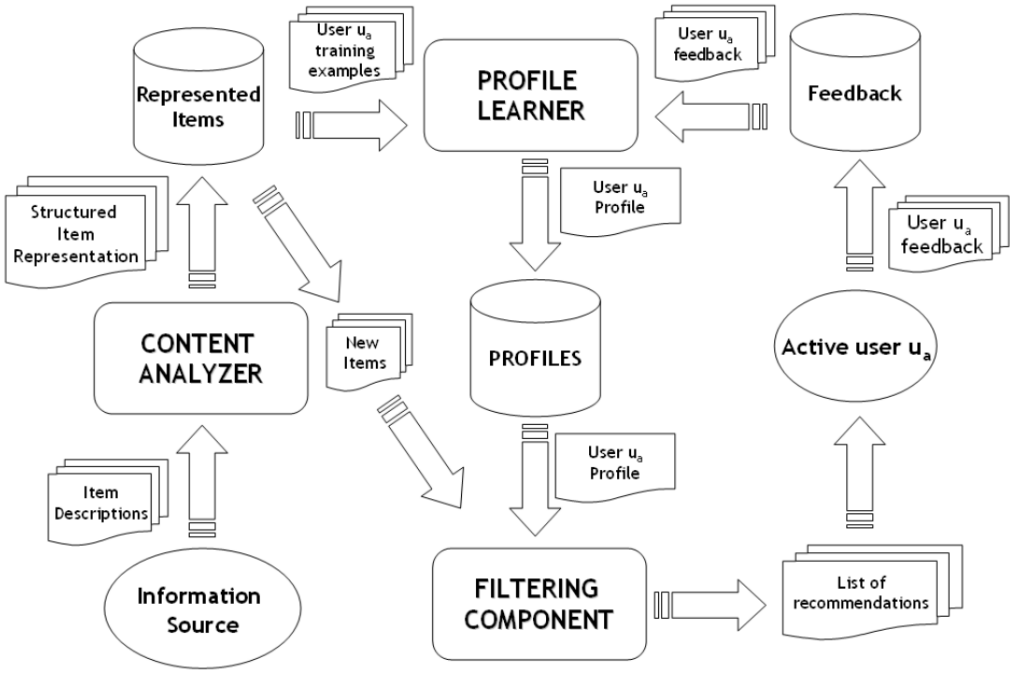

Figure 2.3: High level architecture of a CB recommender (Lops et al., 2011)

The *content analyzer* is responsible for extracting the main information about the different items. It often uses techniques from Information Retrieval systems and natural language processing. Those structured representations of items are then stored in the *represented items* repository. The repository *feedback* contains the ratings of users for the different items (it is the user-item matrix). By taking the represented items and the feedback repositories as input, the profile learner can then apply supervised learning algorithms in order to extract the different profiles of the users (which are stored in the *profiles* repositories). Then, for every item (stored in the *represented items* repository), the *filtering component* predicts whether it is susceptible to interest the active user or not (for instance, it can try to approximate the ratings for those items to evaluate the levels of interest). Based on the interest of every item, the filtering component produces a *list of recommendations*, generally by taking a specified number of the most interesting recommendations (recommendations therefore need to be sorted by level of interest). Finally, because user tastes generally change over time, the feedback from the user to the recommended items is then taken into account and the cycle repeats by executing the profile learner with the new feedback.

We will now take a look at the advantages and drawbacks of this type of RSs. First, CB recommenders are user independent, meaning that the recommendation are unique to each user's history independently of the other users. Also, this kind of systems can be relatively transparent compared to other methods. It is possible to explain the different features that caused an item to be recommended to a specific user. Finally, CB recommenders are able to recommend items that have never been rated. However, CB recommenders need to have enough information about the items in order to efficiently find similar ones. They also suffers from the new user cold start problem (section 2.3.1), which implies that they may not be able to provide relevant recommendations for new users or users with almost no history of items. (Lops et al., 2011)

CB recommenders are well-suited for wine recommendation, where we have numerous items (with no or almost no rating) with not a lot of ratings. It is therefore important to be able to recommend those items even though they have never been rated before. We also have a lot of distinctive features in the dataset that we will use, which makes it possible to extract substantive data for each wine. In this work, we also make the assumption that a user is likely to like wines similar to the one it liked before, thus the choice of this type of RS.

We will now have a deeper look at different popular methods used to learn user profiles by the profile learner.
A common approach is the use of *statistical methods* (where we want to approximate $P(C|F_1, ..., F_n)$, the probability of the class C (generally the rating for RSs) knowing features $F_1, ..., F_n$. Popular statistical methods include Naive Bayes classifiers and multi-variate Bernoulli models. Both models are defined in (McCallum & Nigam, 1998).

The major current field of research on the topic of CB recommenders is the creation of hybrid recommenders (recommenders combining different RS types). (Geluvaraj & Sundaram, 2021) uses a hybrid RS by combining a CB recommender with CF recommender in order to improve performances over non-hybrid recommenders. In a similar way, (Wei et al., 2021) aims at tackling the cold start problem (section 2.3.1) by hybridizing a CF recommender with a CB model.

Cosine similarity

RSs often need to evaluate how far (or how different they are) two users (for CF recommenders) or two items (for CB recommenders) are from each other in order to make recommendations. A popular metric that can be used to compare two vectors and that we will use later in this work to evaluate the distance between two wines is the cosine similarity metric. This metric measures the similarity between two vectors by using the angle θ between them (the larger it is, the more different the vectors are). Figure 2.4, taken from (Singh, Maurya, Tripathi, Narula, & Srivastav, 2020), shows this angle between two 2-dimensional vectors $v1$ and $v2$:

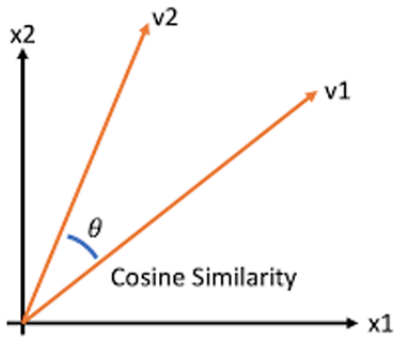

Figure 2.4: Graphical representation of the cosine distance in 2D-space (Singh et al., 2020)

The cosine distance is equal to $cos(\theta)$ (it ranges between 0 and 1, with 0 being two completely different vectors and one being two exact same vectors). Mathematically, this measure equals

$$\frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

.

2.1.5 Wine recommenders

We will now take a look at a specific RS use-case, which is wine recommendation. Wines may be of numerous varieties and have many flavors. In this context, the challenge is to abstract the tastes of users (often considered as a strongly subjective human sense) as good as possible.

An approach is the use of a PCA-K-Means algorithm, such as in (Katarya & Saini, 2022). It uses a clustering model, trained on a wine descriptions dataset of

6497 wines with 12 attributes. Different wines are put in different clusters, and are then recommended to the target user with a greedy technique.

A specificity of wine recommendation is also the difference in wine descriptions that can occur between experts and non-experts in the field. Indeed, it is unlikely that neophyte in wine tasting will describe wine the same manner as a sommelier. An approach to bridge the vocabulary gap is given in (Cruz, Van, & Gautier, 2018). In order to accomplish this goal, the paper uses a LSTM[3] architecture and evaluates the results on a hybrid recommendation system incorporating content-based ontology and keyword-based ontology.

2.2 Evaluation

It is important to evaluate RSs, to assess if recommended items are relevant to the users. In this section, we will discuss different evaluation metrics, which are used later for wine recommendation, and techniques that can help to apprehend the performances of a RS.

2.2.1 Metrics

Metrics are used to evaluate RSs performances numerically and compare them between one another. There exists multiple metrics that can be used to assess performances of a RS. We will describe some of the most popular in the subsections below.

Accuracy metrics

Those metrics are from a machine learning perspective. They evaluate the error of the predictions, and therefore, the accuracy of the model. The application of those metrics needs a prior dataset divided into a training and testing set. The model is then trained on the training set and evaluated on the testing set. Then, the error is defined as the deviation between the predicted and the real value (it should hence be minimized). Three main metrics used to estimate the accuracy are identified in (Chen & Liu, 2017). Firstly, there is the *Mean Absolute Error*, equal to

$$\frac{\sum_{(u,i)\in Q} |r_{u,i} - \hat{r}_{u,i}|}{|Q|}$$

[3]Long-Short Term Memory

(with Q being the test set, $r_{u,i}$ the true rating of user u of item i and $\hat{r}_{u,i}$ the predicted rating for user u of item i), which is simply the average of the differences between real and predicted values. The *Mean Squared Error*, which equals

$$\frac{\sum_{(u,i)\in Q}(r_{u,i} - \hat{r}_{u,i})^2}{|Q|}$$

, is similar, but squares the differences, which implies that larger differences increase exponentially the MSE. Finally, the *Root-mean-square Error*, which equals

$$\sqrt{\frac{\sum_{(u,i)\in Q}(r_{u,i} - \hat{r}_{u,i})^2}{|Q|}}$$

, is similar to MSE but is more easily interpretable because it is directly in terms of measurement units. The smaller those metrics are, the more accurate the model is.

In this work, we will use the RMSE to evaluate the model, since it is a directly interpretable in terms of wine score.

Other metrics

Other popular metrics to evaluate RSs include the precision, the recall and the F-measure. However, those metrics are more suited for classification problems.

2.3 Challenges

Building an efficient RS is not an easy task. Indeed, the performances of a RS can vary widely according to the use cases. However, some recurrent challenges have been discussed in the literature and will be reviewed in this section.

2.3.1 Cold Start

The *cold start problem* occurs when a new item or a new user is added to the system (a new empty row or column is added to the user-item matrix). Indeed, in the case of a new user, it may be hard (especially for CB recommenders) to know which items are relevant since we have no history to infer its tastes. The same issue is true for new items, where it may be difficult (especially for KB recommenders) to know the item value and recommend it since it has no ratings from any user. This issue can also lead to the apparition of 'sleepers', which refers to items actually good but never recommended due to their lack of ratings (Khusro et al., 2016).

The cold start problem is itself a wide topic in the field, with numerous papers experiencing different methods to try to tackle it. Some basic solutions, provided in (Khusro et al., 2016), are to ask the user to rate a few items or to explicitly to give their tastes. We can also make initial recommendations using the user's available demographic information such as the ZIP code or external data to estimate the rating of a new item. Note that the first two solutions correspond to the use of KB recommenders, which are often quite resilient to this problem.

Our wine recommender will suffer from the new user cold start problem. In this work, we will provide a recommender (the global recommender described in section 3.3.2), that provides a potential solution to this problem. However, a lot of potential further research can be made on this use-case, due to its complexity.

2.3.2 Sparsity

The *sparsity* of the user-item matrix can be a major issue for making an efficient RS. Indeed, the lack of information may cause difficulties to the RS to provide good recommendations, especially for CF recommenders. Popular approaches to tackle this problem include multidimensional recommendation models, SVD[4] techniques and demographic filtering.

In this work, we will be impacted by this issue due to the large number of wines reviewed by a few reviewers. However, because we are using a CB recommender, the lack of ratings does not affect the capacity of recommendations that much, since we can still provide similar items even though the reviewers did not rate a lot of them.

2.3.3 Lack of (good-quality) datasets

As for a lot of fields in computer science, the creation of a good RS often needs a large-enough *good-quality dataset* to train the model. Even though this issue is not specific to the use-case of RSs, there are a large variety of use-cases that lacks good datasets. A lack of a good-quality dataset may also impact the evaluation of the RS. Ignoring the intrinsic impact on performances caused by a model trained on a bad dataset, it may even not be possible to decently evaluate the model without a proper testing set (Khusro et al., 2016).

[4]Singular Value Decomposition : (Stewart, 1993) defines it as "the factorization of a matrix A into the product $U\Sigma V^H$ of a unitary matrix U, a diagonal matrix Σ, and another unitary matrix V^H"

We will try to address this problem in this work, in the context of wine recommendation, which is a field that greatly suffers from this issue. For that, we will try to combine two open datasets to create a usable one for wine recommendations.

2.4 Datasets combination

Datasets combination (or datasets merging) can be summarized as the process of finding same or similar entries between two or more datasets, in order to obtain a new merged one. It is often not possible to match entries directly, as they are generally not exactly of the same format and because multiple attributes may mismatch between datasets. The challenge is therefore to find techniques to evaluate the similarity between entries to evaluate if they refer to the same item, with taking into account the potential differences between attributes. This problem has become more relevant to the massive yield of data we observe in the current era (Rosenman, Basse, Owen, & Baiocchi, 2020). Datasets merging is strongly dependent on the use-case. In this work, we will try to solve this problem on the topic of wine recommendation. In the following sections, we will take a look at three main approaches the literatures provides on the subject.

2.4.1 Deep learning

A recent approach to dataset merging is the use deep learning techniques such as in (Srinivas, Gale, & Dolby, 2018). It utilizes two datasets which contains company names and people's name retrieved from Wikidata[5] which contains a collection of open datasets. Those two datasets are then used with other datasets to map those company names and people's name. The model abstract features from the datasets thanks to a deep learning model, and then performs a nearest neighbors model to perform the merging.

2.4.2 MapReduce

Another approach is the use of a MapReduce[6] framework to evaluate string similarity (which can then be used to find similar entries) called *MassJoin* (Deng, Li, Hao, Wang, & Feng, 2014). Each string is compared to another one based on their set of signatures (themselves computed on a set of their substrings). In the filtering phase, potential similar strings (strings that share at least a signature) are identified. Then, in the map phase, we use the signatures as keys and the string as values. Finally, during the reduce phase (processed in parallel), nodes take a key-pair value,

[5] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

[6] Framework proposed by Google to process large-scale data in parallel

split the value in half, and returns a pair of string with their similarity from the list of potential matches.

2.4.3 String similarity functions

Similarity functions as functions that "map a pair of strings s and t to a real number r, where a smaller value of r indicates greater similarity between s and t" (Cohen, Ravikumar, Fienberg, et al., 2003). The paper identifies two main types of similarity functions. The first one is the *Levenstein distance* :

$$\text{lev(s,t)} = \begin{cases} \max(|s|,|t|), & \text{if } min(|s|,|t|) = 0 \\ \text{lev(s-1,t-1)}, & \text{if } a[0] = b[0] \\ \text{c} + \min\begin{cases} \text{lev(s-1,t)} \\ \text{lev(s, t-1)} \\ \text{lev(s-1, t-1)} \end{cases}, & \text{otherwise} \end{cases}$$

, which assigns a cost c to operations that needs to be made to string s to obtain string t (third function's term). Operations are character substitutions, insertions and deletions.
The other one is the *Jaro* metric. By defining s' as the characters which are common to t, t' being analogous and $T_{s',t'}$ being half the number of transpositions from s' to t', the Jaro distance equals

$$\frac{1}{3}(\frac{|s'|}{|s|} + \frac{|t'|}{t} + \frac{|s'| - T_{s',t'}}{|s'|})$$

. A variant, the *Jaro-Winkler* distance also includes P, the length of the common prefix along with P' which equals $max(P, 4)$. This distance equals

$$Jaro(s,t) + \frac{P'}{10}(1 - Jaro(s,t))$$

(Cohen et al., 2003). The Jaro-Winkler distance gives good results, especially for shorter strings.

It is tedious to make a direct comparison of the performances between the three methods just described, since they do not use the same evaluation metrics, same datasets, and the deep learning method is not directly aimed toward string matching in contrary to the two other ones. However, machine learning methods needs large set of data to be trained ,and the MapReduce method is aimed towards big data, while distance metrics are better suited for smaller datasets with simpler strings.

Also, because deep learning methods and MapReduce are quite complex methods that are harder to implement, demands more computing power, and because the string we will compare (wine designation and varieties) are quite short, we will use the Jaro-Winkler distance to merge datasets in this work.

2.4.4 Unstructured data

Another challenge, closely related to dataset merging, is the processing of unstructured data. Unstructured data are, such as defined by (Boulton & Hammersley, 2006) "data that are not already coded in terms of the researcher's analytical categories". Such data includes written text, reports from different institutions, ... Indeed, it may be complex to merge datasets we need entries that are as close as possible and non-ambiguous. In the context datasets, such as done in (Cruz et al., 2018) (section 2.1.5), different NLP techniques can be used to infer the meaning of wine descriptions. Other techniques include data mining, clustering and probabilistic models. (Feldman, Sanger, et al., 2007)

2.4.5 Benefits

Dataset combination can add many benefits, that greatly differs depending on the techniques on which the database is used and the specific use-cases.
The main benefit, is the ability to match entries between multiple datasets, in order to increase the available information. For example, such as done in (Srinivas et al., 2018), we can match people names in different datasets to retrieve their family status, work information, ... Another advantage, in the context of supervised learning, is the information enrichment that the merging can offer. Indeed, some datasets may focus or lack some specific features, and combining can be a good way of increasing the information for the different entries. This increase in features can itself increase the performances of the trained models.

2.5 Work's objectives

Despite the fact that a lot of researches has been done on RS, the specific use-case of wine recommendation has been a lot less studied. A potential reason may be the important lack of open datasets. Indeed, even though some large database exists, such as Vivino[7] or Global Wine Database[8], they are either non-open datasets or does not provide APIs or other ways to exploit data. Some open datasets can be

[7] `https://www.vivino.com/`
[8] `https://www.gwdb.io/`

found on Kaggle[9], however those are scattered and use various formats, which makes them hardly usable for creating a RS.

Even though some papers has applied RS techniques to the wine recommendation use-case, the combination of multiple datasets, in order to improve performances of the recommenders, has not been studied yet.

In this work, we will investigate whether or not it is possible to merge multiple open wine datasets, in order to create a merged dataset containing both wine descriptions and ratings ?
If the creation of such a merged dataset is possible, we will continue by trying to answer the question : How does the accuracy of a wine recommender system vary when it is trained on this dataset, for different parameters ?
Finally, we will try to answer the following question : What is the difference in accuracy between a recommender system trained on this new merged dataset and one trained on a dataset containing only ratings ?

[9]https://www.kaggle.com/

Chapter 3

Model and methodology

Now that we reviewed the literature and introduced the theoretical background, we will continue by explaining the models and methodologies used for this work, in order to perform a dataset merging and create a wine RS. Those models and methodologies will prepare us to create a merged dataset and to evaluate the accuracy of a recommender system trained on this merged dataset. It will also help us assess the accuracy of a recommender when it is trained on the merged dataset and when it is trained only on the dataset that contains the ratings. The link to all the code that has been used for this work can be found in appendix A

3.1 Datasets

In this work, we will attempt to merge two different open datasets. The first dataset contains wine reviews (we will refer to it as the *reviews dataset*), and the second one contains wine descriptions (we will refer to it as the *wines dataset*).

3.1.1 Datasets description and visualization

We will start in this section by describing the two datasets that will use along all this work. The figures used to visualize the number of missing and distinct values per feature can be found in appendix B.

Wines dataset

This dataset is available on *Kaggle* (`https://www.kaggle.com/datasets/dev7halo/wine-information`) and has been posted by the user 김광륜. It contains a total of 21605 entries with 31 columns. From these columns, after excluding the wine's ID and merging the different varieties and regions, we can extract 16 features. Those features are the wine's name, producer, country of origin, region(s), variety(ies),

type, use, abv[1], drinking temperature, sweetness, acidity, body, tannin, price (in won), production year and bottle volume.
Four features contains a large amount of missing values (appendix B.1). Those are, by ascending order, the years, the drinking temperature (degree), the abv, and the price.
Regarding the number of unique values (appendix B.2.1), there are 4825 different producers. Except for wines' names and prices, the rest of the features does not have a lot of distinct values. If we take a look at the taste-related features (sweet, acidity, tannin and body), we can see that they only have five values, since those are ordinarily ranked from 1 to 5.
An example of wine from the dataset with all its features is shown in the table below :

name	Margaux Tradition
producer	앙드레 뤼통 Andre Lurton
nation	프랑스 France
type	Red
use	Table
abv	13
degree	16∼18
sweet	SWEET1
acidity	ACIDITY4
body	BODY3
tannin	TANNIN3
price	nan
year	2015.00000
ml	750.00000
local	['보르도 Bordeaux', '마고 Margaux']
varieties	['Cabernet Sauvignon', 'Merlot', 'Cabernet Franc']

Table 3.1: Example of entry in the wines dataset

Reviews dataset

This dataset, titled "Wine Reviews" is available on Kaggle (`https://www.kaggle.com/datasets/zynicide/wine-reviews`) and has been posted by the user ZACKTHOUTT. It contains 130000 wine reviews from 78214 different reviewers. Points (or scores) for each review are given between 80 and 100 (only wines that scored higher than 80 have been kept) and the dataset contains the reviewer name for each review. Finally, each entry contains some basic information about the wine.

[1] Alcohol by volume

Those are a description (a text from the reviewer describing the wine), the country of origin, the winery, the price, the province, up to two regions, the title and the designation. Let's now take a look at some features from this dataset.
There are a lot of missing values for taster name (appendix B.1.2), which implies for those entries that we do not know who made the review. There are also a great amount of designations missing. However, because designations are like titles, but with less information, and because we see that no title is missing, it should not be a problem for the next of this work. Regarding distinct values (appendix B.2.2), they are 30000 wines reviewed are from 16757 wineries, which makes an average of 1.8 wines per winery. Moreover, there are 18 different tasters (we remove the missing value from the count). It may be either an advantage or an inconvenient, since it implies more data for each taster but may not be representative of a real world situation since this value is relatively small.

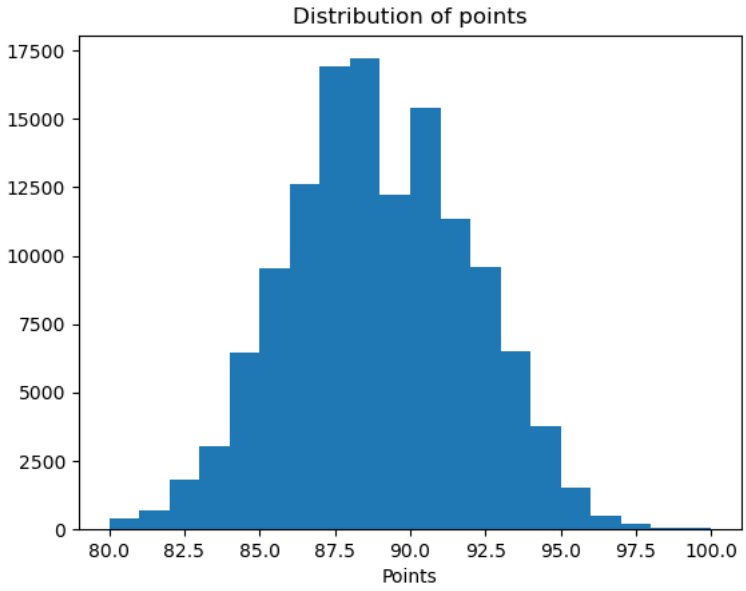

Figure 3.1: Distribution of points in the reviews dataset

On figure 3.1, we can see that the distribution of the scores approximately follows a normal law and that the score with the highest occurrence is 88. Note that the dataset only contains wines that received a score higher or equal to 80 by the reviewer.

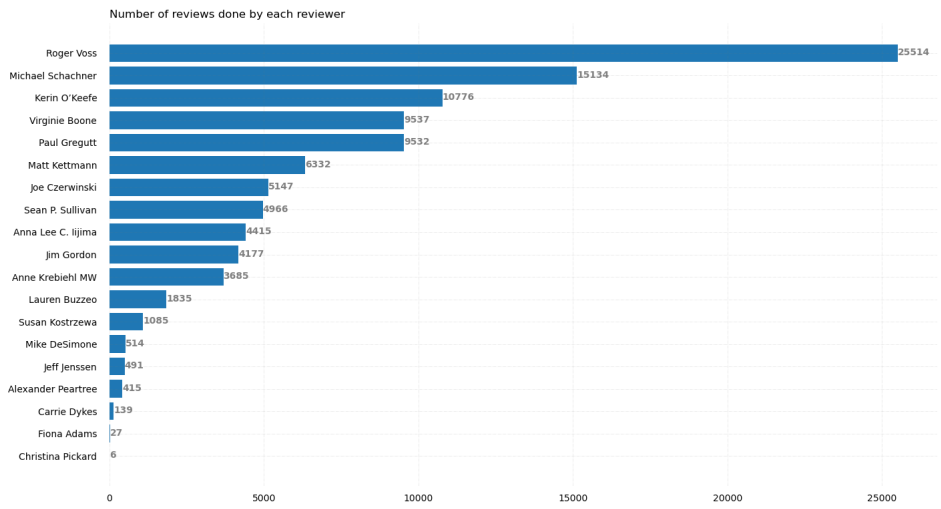

Figure 3.2: Number of reviews performed by each taster (excluding missing value)

Figure 3.2 shows us that the number of reviews for each taster varies a lot. It is useful since we will be able to test recommendation results on both reviewers with a small amount of reviews and ones with a large amount.
Finally, the table below shows us an example of a typical entry of the dataset (the description and the title has been cut off since it is very long) :

country	US
description	Cooked cranberry is spiced with anise [...]
designation	Guidotti Vineyard
points	91
price	64.00000
province	California
region_1	Santa Lucia Highlands
region_2	Central Coast
taster_name	Matt Kettmann
taster_twitter_handle	@mattkettmann
title	Testarossa 2013 Guidotti Vineyard Pinot Noir [...]
variety	Pinot Noir
winery	Testarossa

Table 3.2: Example of entry in the reviews dataset

Note that the *local* and *variety* features have been processed such that all the columns containing those features are merged into a list in a single feature.

3.1.2 Feature processing

Now that we have described and visualized the datasets, we need to process data so that they are formatted similarly, in order to perform the merging. Indeed, if data is formatted differently in both datasets, it will be harder (or even impossible), to detect similar entries and perform the merging.

Wines dataset

First, the *local* (which refers to the region of the wine) and *variety* variables are merged in a list into a single feature for readability.
Then, because some features in this dataset contains korean characters, we remove those characters in order to only have ASCII ones. Next, the discrete values of the taste related features (sweet, acidity, body and tannin) are converted to integers from 1 to 5 (for instance, "SWEET3" becomes 3). Then, because the abv and degree are sometimes defined as an interval of 1 (for abv) or 2 (for degree), separated by "∼", we take the average of the interval to replace the value (for instance, a degree of "16∼18" is replaced with 17). We also use one-hot encoding[2] on the *nation* and *type* features, since it will be useful to train our models later.

The next step is the creation of a new feature "*designation*", that will be the most important variable used for merging. The objective of this variable is to match as close as possible the one present in the reviews' dataset, in order to use it for merging later. The creation of this new feature is based on different format of the *name* feature.
First, if the name does not have a ", " in it, the following cases can occur :

- The name is equal to the producer : we set the designation as missing[3]
- The name contains the producer : we set the designation as the name without the producer
- The name does not contain the producer : we set the designation as the name

However, if we find a ", " in the name, the following cases can happen :

[2] A binary column is created for each distinct value of the feature, indicating if the entry has this feature (1) or not (0)

[3] missing values are represented as nan

- If the string before the ", " is equal to the producer : we set the designation as the string after ", "
- Otherwise : we set the designation as the name but with the ", " replaced by a space

Finally, we check if a variety of the wine is present in the former made designations. If it is the case, we remove the variety from the designation.
All those steps are performed in order to standardize as most as possible the feature, to permit a merging of the best quality as possible.
Some example of name and designation illustrating different cases are shown in the table below :

name	**designation**
Concha y Toro, Terrunyo Cabernet Sauvignon	Terrunyo
Santa Margherita Merlot	nan
Porca de Murca White	White
Noble, Medoc	Noble Medoc

Table 3.3: Example of name and designation features after processing

Reviews dataset

Regarding the reviews dataset, we first remove all accents from texts to only have ASCII characters. Then, in order to have both datasets in the same language, we translate English words (like the wine types) to French. Then, all the reviews with the taster name being not specified are removed, since we need to know which user made the review in order to evaluate the RS later.
We also one-hot encode the features *country*, *variety* and *province* to use them later to train a model.

3.2 Datasets merging

Now that all data is standardized, we can move to the merging of the datasets. This process is done in an iterative way, starting with an intersection on the wine's *winery*, followed by the wine's *designation* and *varieties*.

3.2.1 Winery

The first feature which is used to merge the datasets is the winery (or the wines' producers). Indeed, both datasets contain this information in the same format,

which makes it an interesting attribute to do a first selection of potential common entries in the datasets. Since this variable is defined exactly in the same manner in both dataset, we can simply exclude all wines for which the winery is not present in neither dataset. By doing that, we found that there are 1361 common wineries between the datasets, which reduces the wines' dataset size to 4203 (removing 17402 entries) and the reviews one to 17462 (removing 112538 entries). This step alone therefore reduces the number of entries for both dataset to over 75%.

3.2.2 Designation

Now that we are left with wines of common wineries, we need to find potential matches. In order to determinate matching candidates as accurately as possible, we will use the designation attribute crafted before (section 3.1.2). After data processing, the *designation* feature of wines in both datasets should refer as closely as possible to the designation given to the wine by the producer. This designation often contains a wine's name, color; or may even be undefined if the wine is only designated by the producer's name.
In order to compare the designations of both datasets, the Jaro-Winkler distance is used. This distance is well-suited for this use-case since it performs well for simpler and shorter strings, and because it can be easily and efficiently implemented (section 2.4.3). This distance, ranging from 0 to 1 (with 0 being two completely different strings and 1 being the same strings), is computed between each wine that share a common winery between the datasets, such as illustrated on figure 3.3 (Jaro-Winkler distances are the different d_i).

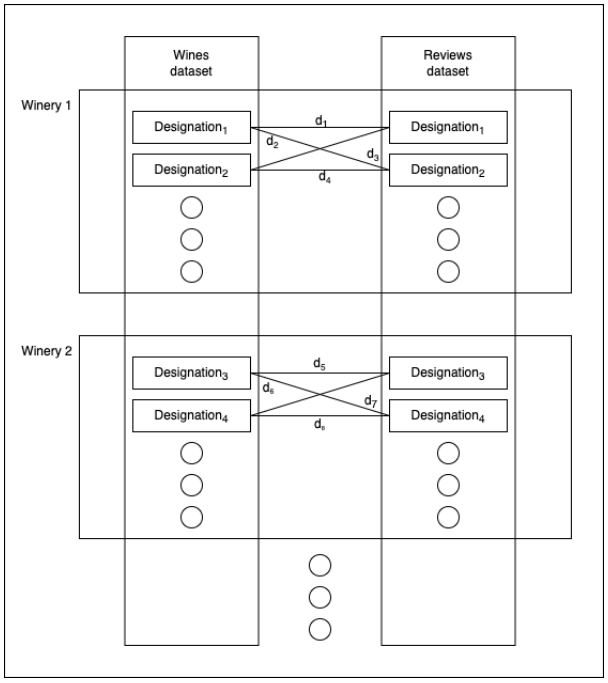

Figure 3.3: Computation of the Jaro-Winkler distance for designation between the wines and reviews datasets

More formally, this corresponds to n matrices of size M_i x N_i, where n corresponds to the total number of different wineries, M_i is the number of entries in the wines dataset that belongs to winery i and N_i is the number of entries in the reviews dataset that belongs to winery i (with $i \in [0, n-1]$).

Injectivity constraint

Before eliminating potential matches, we first have to define the injectivity constraint that needs to be satisfied. Indeed, it is important that a review never matches with more than one wine (otherwise, it would not make sense since a review only rates one specific wine). However, a wine may be referred to by multiple different reviews. In order to enforce this constraint, at the end of the filtering (after the applying the discrimination of variety), in case a review refers to different wines, we will select the wine that has the highest Jaro-Winkler distance (for the *designation*, or the *variety* in case of ties) as the matching wine for this review. By doing that, we follow our general assumption that the higher the Jaro-Winkler distance is, the more similar entries in the datasets are.

Threshold selection

Once all the distances have been computed, we need to define a threshold value to eliminate wines and reviews with designations far from each other (which will be considered as non-matching entries). If the threshold value is too high, the vast majority of remaining entries (after filtering) will certainly be matching. However, it will probably also omit other matching entries by being too restrictive, and reduce the datasets size by a large amount. On the other hand, a threshold value that is too low will result in a higher number of selected entries, but will also select entries that are clearly not matching.
In order to define this threshold, we tested and compared different potential values. Note that those different threshold values will be evaluated more in depth in the next chapter (section 4.1.2), in order to evaluate how those threshold affects the RMSE and to see how efficient our selection is. The steps we performed to select the best threshold value are the following : after setting a seed for reproducibility, a set of threshold candidates (0.1, 0.5, 0.75, 0.8, 0.85, 0.9, 0.95) has been defined (those values have been chosen by doing a rough initial estimation of the results). For each of those values, after filtering, we randomly picked 20 entries and compared the designations between the reviews and the wines dataset. As we can see in appendix C, designations below 0.5 are mostly not matching, with almost no perfect match. Increasing the threshold to 0.75 and 0.8, we see a great improvement, with 16 perfect matches out of the 20 designations (note that even though the other entries are not perfect matches, there is still a high chance that most of them refer to the same wines). However, the number of perfect match decreases to only 9 with a value of 0.85 and some likely non-matching wines can be identified (the appreciation for this statement is left to the reader, however). This result may be explained by a "lucky" pull of entries for the 0.8 threshold. A threshold value of 0.9 results with 16 perfect matches, with 4 other entries being almost certainly the same. Indeed, this value will be used for filtering since it captures enough differences to be flexible enough between both datasets, while being restrictive enough to eliminate non-matching candidates.
The comparison results with different threshold values can be found in appendix C. Now that we have selected an appropriate threshold value, we can eliminate the entries with a Jaro-Winkler distance greater than 0.9 (this operation corresponds to removing links on figure 3.3). This step removes a total of 93859 potential matches from the 99930 initial ones.

3.2.3 Varieties

The second feature used to find matches between both datasets is the wine *variety*. Indeed, discriminating matching candidates only with designation is sometimes

not enough, since some wines may have the same designation (for example, only the type of wine) but originate from different varieties (thus, there are certainly not the same). Even though this filtering greatly reduces the number of dataset entries, it ensures that the remaining entries are almost certainly the referring to the same wines in both datasets. This step removes 4574 potential matches and leaves us with 545 wines linked to 1497 reviews. Finally, after enforcing the injectivity constraint on the 98 reviews that refer to more than one wine, we are left with a final dataset of 1383 reviews that refers to 510 wines. This corresponds to a total reduction of the size of the original datasets of 98,7% for the reviews dataset (when omitting missing taster entries) and of 97,6% for the wines dataset.

3.3 Content based recommender

In this section, we will describe the content based recommender we designed for wine recommendation. The objective of this recommender will be to predict ratings of wines based on a prior set of reviews (like the history of a user's reviews for instance). When the scores of each wine has been predicted by the recommender, it is then possible, based on a heuristic (such as recommending the top-n rated wines), to provide recommendations to the user.

3.3.1 K-nearest neighbors content-based recommender

K-nearest neighbors (KNN) recommenders are a type of RSs that can be both used for collaborative filtering and content based recommendation. In this work, we will use it to approximate the score of wines for different user (CB approach). Our recommender will be similar to the one applied in the context of movie recommendation in (Singh et al., 2020). A KNN recommender predicts the score of a new item using the k closest items, based on a distance metric from this item. The score prediction formula is generally the average of the weighted average of the scores of the k closest items. The main principles of our KNN recommender are the following : in order to approximate the score (or rating) of a specific wine (unrated by the user), we will select the k nearest wines from it. Then, we will compute, based on a formula explained below, an estimation of the wine's rating. The parameters are k, the number of neighbors we want to select and r, the final number of recommendations. A k value which is too high can induce underfitting, which makes the recommender unable to consider wines' specificities, while a k value which is too low can induce overfitting, that may lead to too much sensibility to noise. For our recommender, we will select the k value that yields the best RMSE on the test set. The r parameter needs to be adjusted based on the use-case of the recommender. For example, in the context of a wine selling platform, 5 to 10

recommendations may be more logical than providing only 1, or more than 20 recommendations to the user. Note that r does not affect the RMSE of the model, since the RMSE is computed on all the score predictions.

3.3.2 Global recommender

The first version of RS that we crafted is a global recommender. It is referred as "global" because it estimates the score of all wines based on all the other ones (it is therefore not user-specific). The assumption of this model is that a wine that is liked by a large amount of users (or "globally" liked) is more likely to be liked by other users (that have not tasted that wine yet). To evaluate the score for a wine w, we proceed as the following : first, we select a subset C_w of size k from the set of all wines W. C_w contains all the k wines that are the most similar to w, excluding w itself. Then, we compute the predicted score of w, $\hat{R}(w)$, as the weighted average (with the weights being the similarities $Sim(c, w)$) of the scores for each wine c in C_w, and we repeat the operation for all wines in W. Note that we use the weighted average to prioritize wines that are the most similar to the one for which we want to estimate its rating. We can formally define the estimate rating $\hat{R}(w)$ as

$$\hat{R}(w) = \frac{\sum_{c \in C_w} Sim(c, w) R(c)}{\sum_{c \in C_w} Sim(c, w)}$$

.
The main issue with this recommender lies in the fact that it does not use users history, it estimates the same scores for each wines for all users. This recommender may be used to have an idea of the global appreciation of a new wine, even if it has never been rated by any user. This information may then be used to increase novelty (for example, we could add wines that are globally liked in the recommendation of a user), or to provide better recommendations for users with few reviews. Indeed, it can be used to tackle the cold start problem; if the user has no or a small review history, it may be hard to abstract its tastes, and providing (at least some) globally liked wines may be a good idea to try to figure out what it likes or dislikes.

3.3.3 Taster specific recommender

Even though the first recommender can be useful in some specific situations, in a majority of cases, we want our RS to make recommendation specific and adapted for each user independently. This second type of recommender, referred as "taster-specific", uses the same principles of the global recommender, but adapts them to have personalized recommendations for each user. It works as the following : we first define the set of all wines that have not been rated in the past by the user u,

W_u, and the set of all previous reviews made by a user u, H_u. For each wine w in W_u, we take a subset C_w of W_u, which contains the k most similar wines to w in H_u, excluding w itself. Then, we define the estimated rating of w for u, $\hat{R}(w,u)$, as the weighted average of the scores $R(c,u)$ attributed by u to each review of c in C_u (with the weights being the similarity $Sim(c,w)$). The difference with the global recommender is that the neighbors of the wines not yet rated by the user can only be selected in the wines it has already rated (in its history), instead of all the wines. We can formally define the recommender as

$$\hat{R}(w,u) = \frac{\sum\limits_{c \in C_w} Sim(c,w) R(c,u)}{\sum\limits_{c \in C_w} Sim(c,w)}$$

.
After we have our predicted scores (from either of the previously described recommenders), we can define a heuristic to provide r recommendation(s) to the end user. An example of heuristic that may be use is to trivially select the top r wines with the best predicted ratings, since those are the ones that the user is the most prone to like. However, several other heuristics could be used, in particular by adding randomness in the predictions, in order to increase novelty or other characteristics.

3.4 Evaluation

In this section, we will first describe the different evaluation protocols we will use to evaluate the accuracy in different scenarios.

3.4.1 Evaluation method

In order to evaluate our models, we will perform a 5 times repeated 5-fold cross validation. We use repeated cross validation in order to alleviate potential large variance between runs of the cross-validation (it can especially occur when the dataset is small, which is the case for the taster-specific RS). We have chosen a number of repetition of 5 because it provides a great balance between the alleviation of large variances and performances (10 times repetition was too computationally intensive). The evaluation metric we will use across all the comparison is the RMSE.

Recommender systems evaluation

We will start by evaluating the different RSs. Those RSs are the global, taster-specific and random recommenders. The last recommender makes score predictions by following a continuous uniform distribution between 0 and 1. For each recommender, the RMSE will be evaluated with different k values (the repeated 5-fold is performed

for each of those), in order to determine which one of these values yields the best result. The set of k values that will be tested is 1, 3, 5, 10 and 15. For the global and random recommender, we will simply perform the repeated cross-validation on all the reviews in the merged dataset. However, for the taster-specific recommender, the repeated cross-validation will be divided on reviews for each taster from the same merged dataset (since the recommender only uses reviews specific for each taster to predict the scores). As we can see on figure 3.4, there are great disparities between the number of reviews for each reviewer. In order to avoid datasets that are too small for the cross-validation, we will only consider the 10 tasters for which we have 31 reviews or more. Even though we will get training sets of size as low as 24, we will also evaluate the RMSE for different training set sizes to see how it impacts the accuracy, and to assess if it is too low or not.

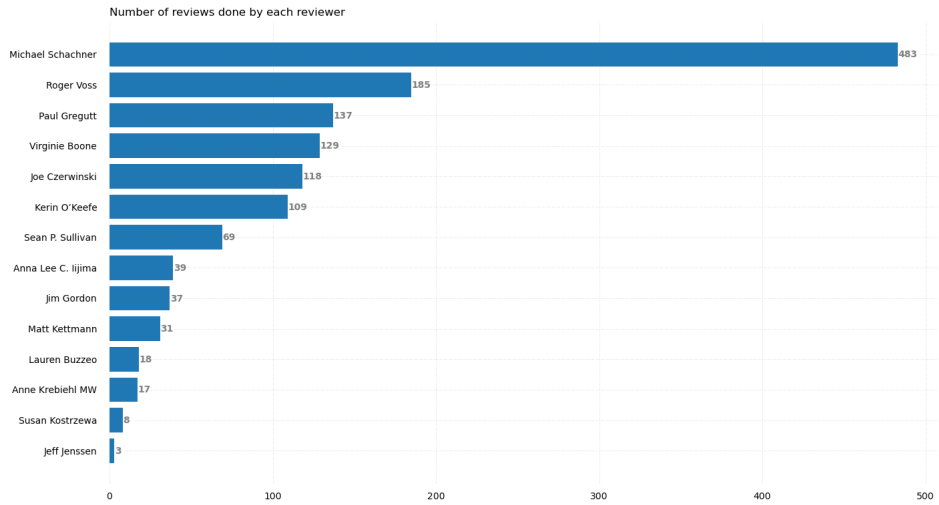

Figure 3.4: Number of reviews for each taster in the merged reviews dataset

Next, we will compare the RMSE for different Jaro-Winkler threshold values, in order to see how this variable affects the performances, and how our threshold of 0.9 compares to other thresholds.
Then, thanks to the wide variety in the number of reviews per taster, we will also compare how the RMSE of the taster-specific RS is affected by the number of previous reviews of the user. In order to do that, we will compare the RMSE for each specific taster (with their own different reviews number), and see how the number of reviews affects the RMSE.
Regarding the features we will be using in the wines dataset to make our predictions,

the four main ones are the taste related features : *sweet*, *acidity*, *tannin*, *body* (which ranges all from 1 to 5). Indeed, those features, not present in the original reviews dataset, should extract most of the taste of a wine and be able to best approximate the user tastes. Even though features *abv* and *degree* might give some indication of the tasters preferences, we will not use them since almost 25% of the wines in the wines dataset have missing values for both those values. We will perform an evaluation using only the taste related features, and then with an extended set of features with the *type* and *nation* features (which have been one-hot encoded before) to see how it affects the performances.

Dataset merging evaluation

In a second time, we will compare how the performances vary when the model is trained only using the reviews dataset and when it is trained on the merged dataset. Our reference results for the merged dataset will be the best performing model trained before (with taste only or extended features trained model). For the model trained on the original reviews dataset, an interesting feature that could be exploited is the *description* (which contains a description of the wine's taste). However, the exploitation of this feature would need the use of NLP to abstract the meaning of the text, which goes beyond the scope of this work. The other interesting features, that we will use to evaluate the model, are the *province*, the *variety* and the *price* (the *province* and *variety* are one-hot encoded before training the models). We will not use the *country*, since the information is already contained is more precisely contained in the *province*, nor the *regions*, since it has too many distinct values for each wine and therefore does not contain a lot of information. The full reviews dataset will be used for the evaluation, and entries containing a *nan* value for either of the previously given features are deleted from the dataset. The entries for which the *taster name* is unknown are also removed. Finally, as for the merged dataset, we also remove tasters with less than 20 reviews. The full reviews dataset contains 17462 reviews and is reduced to 16160 once entries with *nan* value are removed.
Note that due to computing time issues, we will only perform a 5-fold cross-validation and will not repeat it 5 times as before. It is not a problem since the dataset size is a lot larger than the size of the merged dataset.

Chapter 4

Results

In this chapter, we will describe and discuss the different results we have obtained by following the experimentation protocol described in the previous chapter. Raw data of those results, including the standard deviation for the 5 times repeated 5-fold, can be found in appendix D. We will start by analyzing the results concerning the merged dataset. We will then follow with the comparison of the performances when the model is trained on the merged dataset, and trained only on the reviews dataset.

4.1 Merged dataset models

In this section, we will expose and discuss the results obtained when the model is trained on the merged dataset. Indeed, we will try to figure out how different models trained on the merged dataset perform compared to each other, in order to determinate the best parameters, which are the k value, model type, the Jaro-Winkler threshold and the feature set.

4.1.1 Different models

First, we will take a look at the RMSE for the three model types evaluated in this work : the global, taster-specific and random recommenders. As a reminder, the global taster predicts the score of a wine based on all the similar wines that have been rated, while the taster-specific recommender uses only similar wines that have been rated by the user for which we want to estimate the score. The random recommender, meanwhile, makes prediction by following a continuous uniform distribution between 0 and 1.

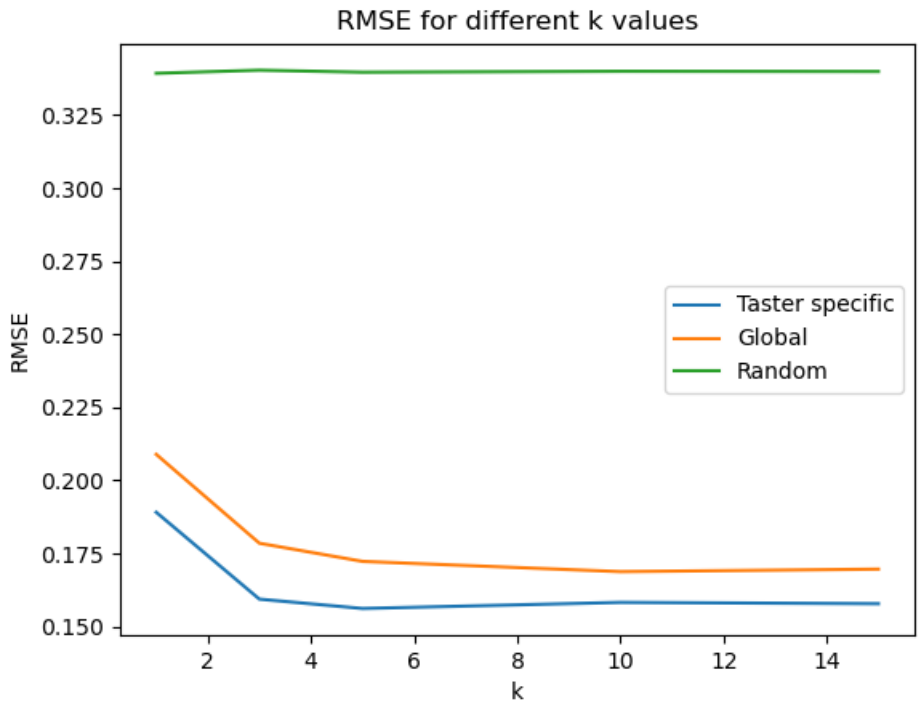

Figure 4.1: RMSE of different models trained on the merged dataset for different k values

Figure 4.1 shows the different RMSE values for each models for different k values (with five iterations per k). As a reminder, k is the number of neighbors (most similar wines) that the model uses to predict the score of a wine. The first thing we can see is that the random recommender's RMSE is constant at about 3.34 across all k values. It makes sense, since this recommender does not use neighbors to estimate wines' scores. Also, for both other models, we can see that the optimal k values are equal or above 5, since the RMSE is higher when k equals 1 and 3. This saturation value around 5 can be explained by the fact that this value is large enough to have enough information about the wine in order to avoid underfitting, but is not too high such that it becomes too general (the more wines we select, the less specific they are in relation to the wine for which we want to predict the rating). Regarding the results for each model, we can see that both the global and taster-specific recommenders have a significantly lower RMSE than the random recommender (by about 49.3% and 54% for k = 5). We can deduce that both those recommenders are able to extract meaningful information from the dataset in order to make predictions. When we look at the global recommender in comparison to the taster-specific recommender, we can see that the second one's RMSE is about

10 % lower for all k values. This constant difference in RMSE means that the taster-specific recommender is able to extract specific information from each taster, in order to better predict the scores.

4.1.2 Jaro-Winkler threshold

We will now take a look at the RMSE values, when we use different Jaro-Winkler threshold values for the merging of the dataset to train the taster-specific model.

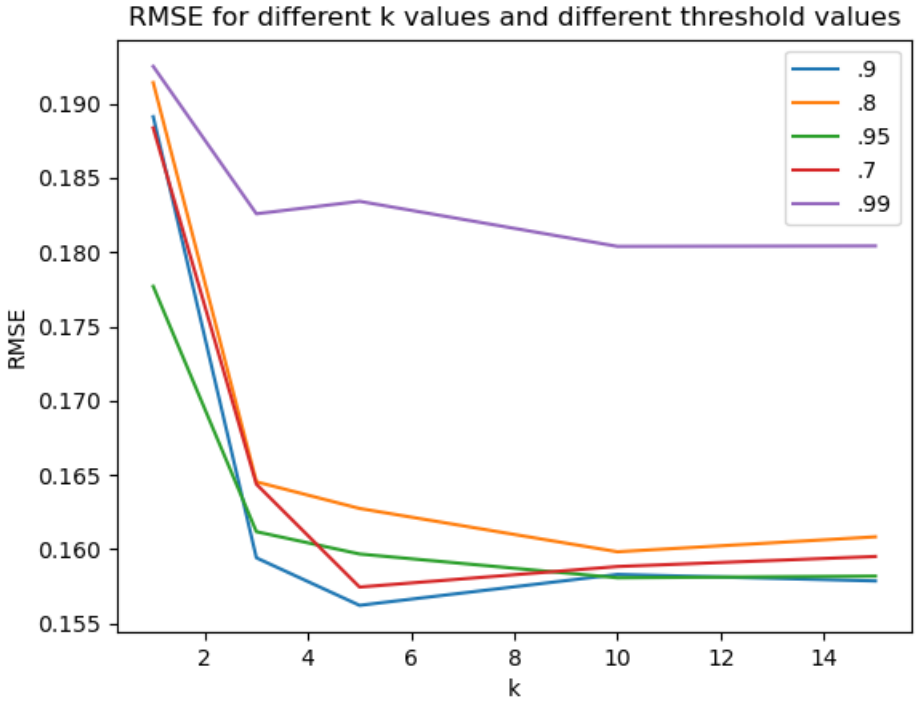

Figure 4.2: RMSE of the taster-specific model for different threshold values for the merged dataset

As we can see on figure 4.2, the RMSE values for all thresholds have similar shapes. Indeed, the RMSE goes down when k increases until it reaches 5 (for thresholds of 0.7 and 0.9) to 10 (for thresholds of 0.8 and 0.95). The only exception occurs when the threshold is 0.99, since the RMSE seems to stabilize for all k values (except when $k = 1$) towards a RMSE of 0.183. We can also see that the RMSE values seem to converge towards the same value (around 0.158) when k increases (except for a threshold of 0.99). It can be explained by the fact that when k increases, the prediction becomes more general (since it uses more wines)

and therefore they become more similar (and some wines may overlap between the different thresholds). The lowest RMSE is obtained with $k = 5$ for a threshold value of 0.9 (which is the threshold value we assessed to be the best previously in this work). For this reason, we will keep a threshold value of 0.9 with $k = 5$ as our reference model.

4.1.3 User's history size

We will now take a look at the variations of the RMSE for the different tasters individually, to observe how it is affected by the number of previous reviews made by the tasters (history).

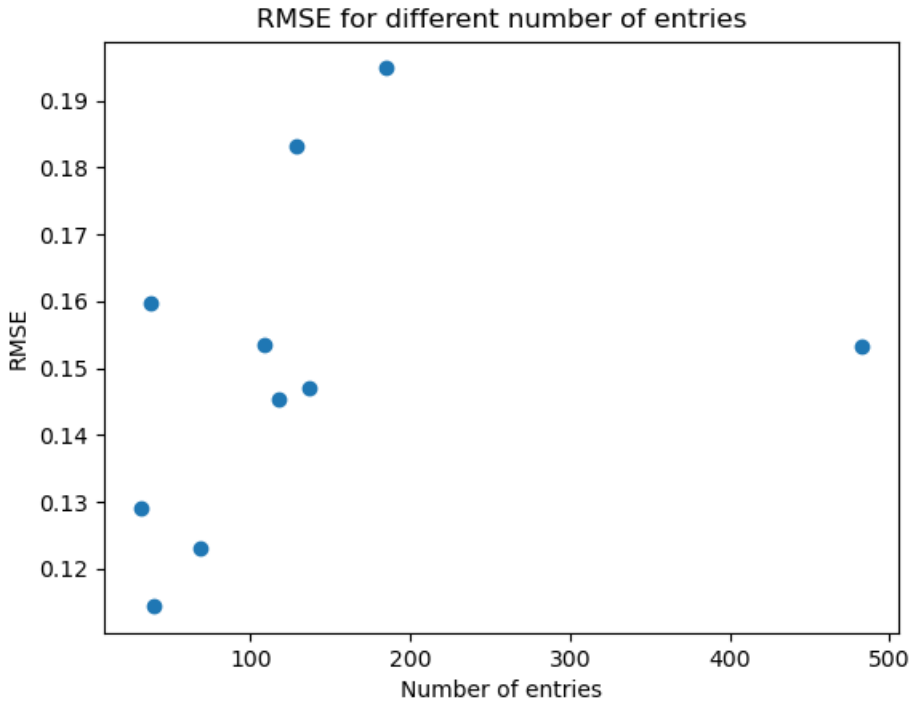

Figure 4.3: RMSE for different user's history sizes

On figure 4.3, we can see that the RMSE is not significantly affected by the size of the taster's history. In fact, the RMSE seems to generally increase when the history size increases, up to about 200 entries. However, we can see that it goes back down to a RMSE of 0.15 when the number of entries goes up to near 500. It does not seem that there is any significant correlation between the RMSE and the history size. This result may be considered counterintuitive, since a larger number

of entries should imply more information for the model to work with, which should lead to better score predictions. A possible explanation for such dispersed data points may be that a history size of less than 200 wines (perhaps even 500, since we do not know how the RMSE behaves above that) is not enough to affect the performances of the model.

4.1.4 Extended feature set

Finally, we will take a look at the variations of the RMSE when we extend the set of taste-related features with the *type* and the *nation* features.

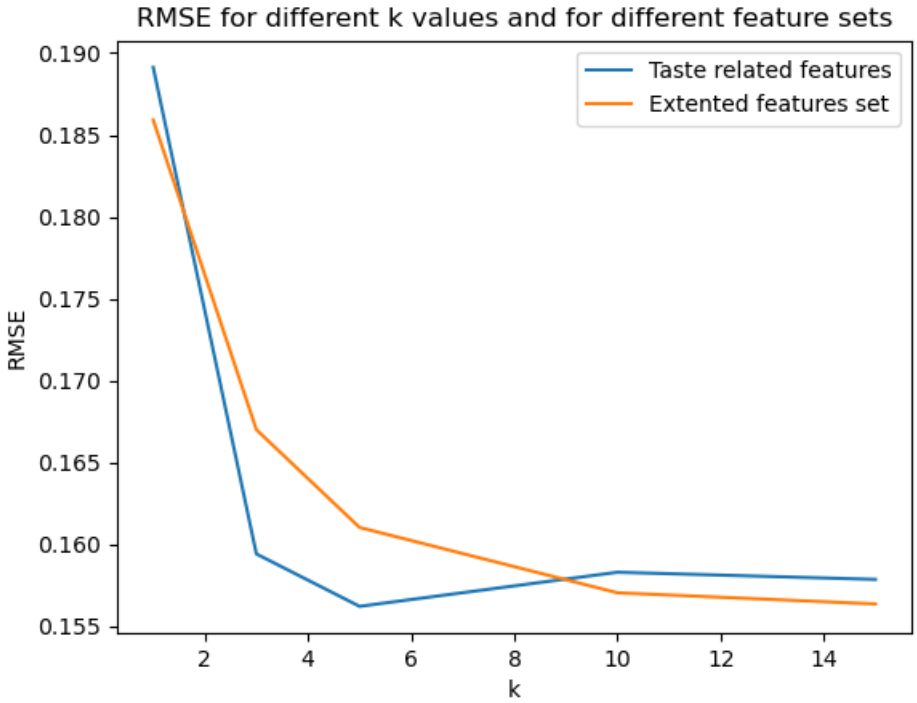

Figure 4.4: RMSE for model trained only on limited and extended feature set, for different k values

As we can observe on figure 4.4, extending the set of features have a mixed effect on the RMSE. Indeed, when k is below 10, the RMSE is lower when the model is trained only on the taste related features. However, the situation reverses when k goes above 10, and even reach almost the same lowest RMSE attained by the first curve when k is equal to 15.
This result implies that adding those two features, the model is better able to

predict the score when the number of neighbors selected is high. In this case, it therefor seems that the model needs more wines (increase of k) to predict the score when it has more information available (increase in number of features). But we should also keep in mind that increasing k even further may lead to overfitting and that adding features increases the complexity of the model. For those reasons, and because the difference between both models is negligible, we will choose to use the model trained with the taste-related features (with $k = 5$) as our reference model for the next step.

4.2 Merged and reviews datasets comparison

Now that we have tested different parameters for our models, we will look at how the RMSE of the best performing model trained on the merged dataset compares to the one only trained on the reviews dataset. As we found in the previous section, the best performing model is the taster specific recommender trained with a threshold of 0.9, a k value equal to 5, and uses only taste related features. We will use the taster specific model trained on the reviews dataset for the comparison.

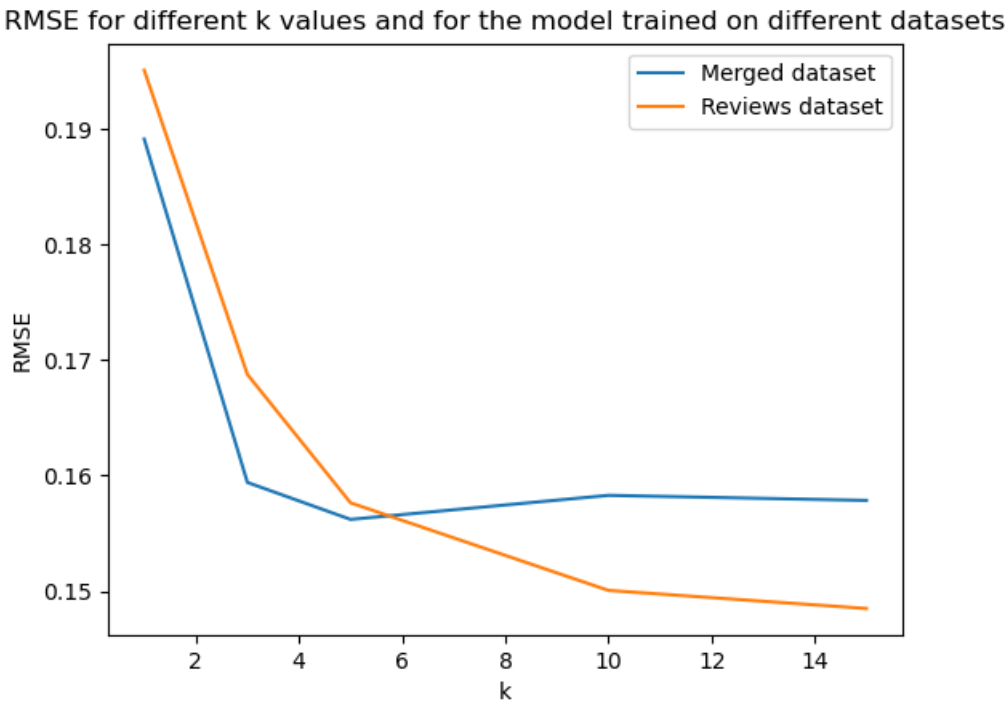

Figure 4.5: RMSE for different k values for a model trained on the merged dataset and a model trained on the reviews dataset

Figure 4.5 shows the RMSE for different k values for both the model trained only on the reviews dataset (in yellow) and the one trained on the merged dataset (in blue). We can see that for k values below 5, the RMSE for the model trained on the merged dataset is lower than the one trained on the reviews dataset. However, the result shifts when k is above 5. Indeed, while the RMSE model trained on the merged dataset is minimized at k = 5 and then goes up, the RMSE for the second model continues to decrease, and goes below the RMSE obtained with the first model. If we take the minimum RMSE values for both models, the second model has a minimum RMSE that is 5 % lower than the first model.
We can conclude that the model trained only on the reviews dataset is able to more accurately predict wine scores than the model trained on the merged dataset, for a k value greater than 5. If we consider only the best (lowest) RMSE for both models, the model trained on the reviews dataset is 5 % more accurate than the other one. This result may be explained by different things. First, it could be the result of a lack of common entries between both datasets by nature, which would lead to a merged dataset that is too small to efficiently train the model. Another possibility is that our merging method is not efficient enough. It could either omit too many

matching entries, or consider too much non-matching as matching, which would decrease the dataset size or increase the number of errors in the dataset, which would impact negatively the accuracy. However, we should keep in mind that both models perform not that far from each other (only a 5% difference), which implies that more advanced merging techniques and/or deeper tweaking of the model parameters may close the performance gap, or even yields better results than the model trained on the reviews dataset.

Chapter 5

Conclusion

In this work, we investigate whether or not it was possible to merge multiple open datasets, in order to enrich ratings with wine description. We also looked at how the accuracy varies for different parameter configurations, when a wine recommender is trained on this new merged dataset. Finally, we evaluated how the accuracy of a wine recommender varies when it is trained on this merged dataset, or when it is trained only on a dataset containing ratings.
First, we managed to merge two open datasets found on Kaggle, containing reviews and wine descriptions. The merging was primarily possible thanks to the use of the Jaro-Winkler distance between the designation feature, specially crafted with other variable, to find matches between both datasets.
Next, we created a KNN recommender system and evaluated the performances of this model when it is trained on the new merged dataset. We assessed the model's accuracy with different parameter configurations, which include different model types (global, taster specific and random), Jaro-Winkler thresholds, history sizes and feature sets. We discovered that the configuration yielding the lowest RMSE with was the taster-specific model trained with the taste related features, with a Jaro-Winkler threshold of 0.9 and a k value of 5.
Finally, we compared the performances of the taster-specific model, when it is trained on the merged dataset, and when it is trained only on the reviews dataset. We found out that the second model performed slightly better for k value higher than 5 than the other model. The merged dataset has therefore not been able to provide a higher accuracy, in comparison to only training the model on one dataset. However, the application of dataset merging in the field of wine recommendation, even though the merged dataset did not give lower RMSE than when the model is trained on one dataset, is an approach that should still be considered. Indeed, the use of advanced merging techniques, with a finer adjustment of the parameters, may close the performance gap or even lead to better results.

This field has potential for a lot of other research. We will follow with some recommendation of research. First, it may be interesting to get some statistical information about the correctness of the potential matches, which may be done using independent surveys. This would be useful to better assess the quality of the merging, and see if it affects significantly or not the results. Also, the use of more advanced features processing and selection techniques may lead to different results. More specifically, using NLP techniques, especially on the reviews dataset, may lead to better score predictions. Next, in this work, we only focused on the accuracy of the models, using the RMSE. However, many other evaluation methods might be used to assess the performances of a recommender system. Some of those include other metrics, such as the F1 score, or user survey to evaluate characteristics like novelty. Finally, we only focused on the merging of two datasets. The use of external resources, such as online wines platforms, may lead to a more efficient merge between datasets, or enrich this merged dataset.

Appendix A

Code link

All the code that has been used throughout this work can be found at this URL : `https://github.com/MaxDill/wine_recommender`.

Appendix B

Datasets vizualisation

In this appendix, we show graphs that we used to visualize both the reviews and the wines datasets.

B.1 Missing values per feature

B.1.1 Wines dataset

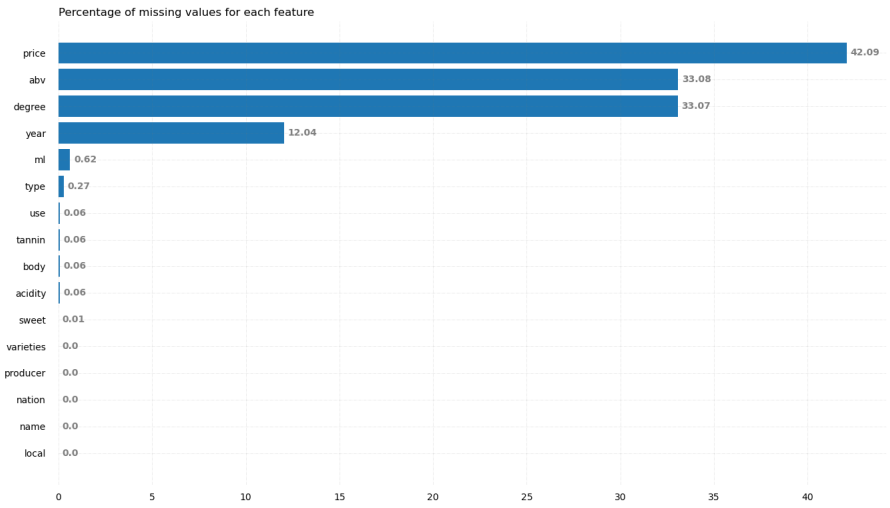

Figure B.1: Percentage of missing values for each features of the wines dataset

B.1.2 Reviews dataset

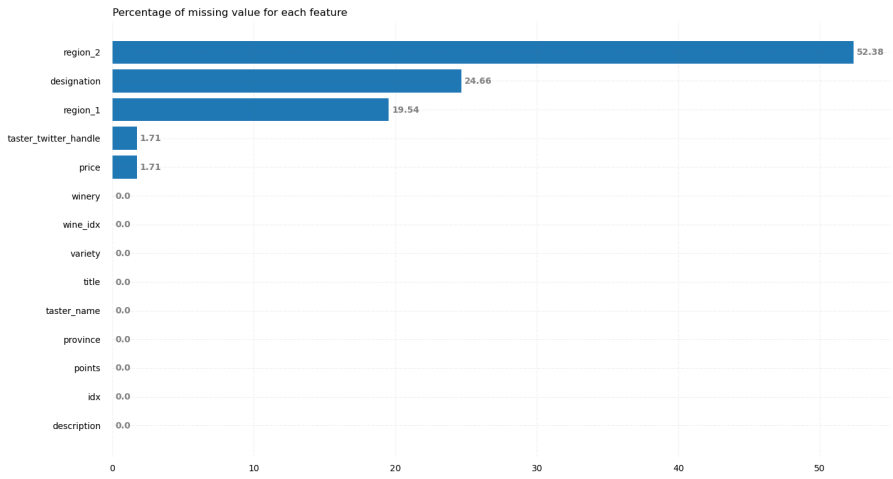

Figure B.2: Percentage of missing values for each features of the reviews dataset

B.2 Distinct values per feature

B.2.1 Wines dataset

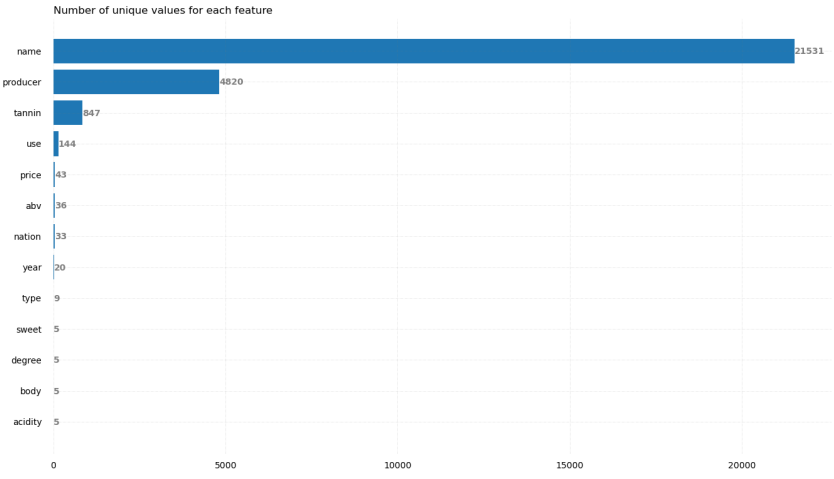

Figure B.3: Number of unique values for each features of the wines dataset

B.2.2 Reviews dataset

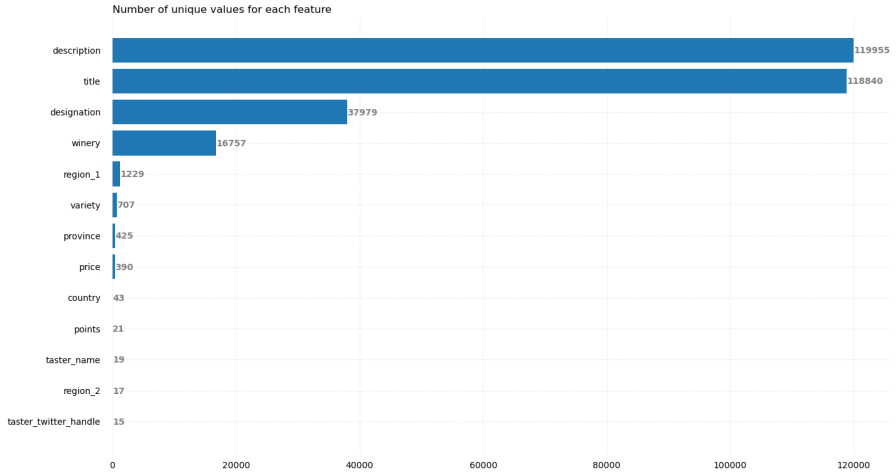

Figure B.4: Number of unique values for each features of the reviews dataset

Appendix C

Threshold selection

In this appendix, we report the results for selecting the Jaro-Winkler threshold value, such as described in 2.4.3. The different threshold values refer as t and the results are 20 randomly selected designations. Note that the template below are of the form "[Winery], [Designation]", (the winery is always the same between the wine and the review), in order to give some context to the reader when reading the results. Also, we should note that it is possible that some wines are drawn from the set multiple times, since multiple reviewers could review the same wine.

C.1 t = 0.1

Wine designation	Review designation
Echeverria, Signo 1	Echeverria, Founder's Selection
Domaine Laroche, Laroche Chablis Grand Cru Les Blanchots	Domaine Laroche, Les Clos Grand Cru
Agustinos, Winemaker Selection	Agustinos, Reserva Privada
Sequoia Grove, Napa Valley	Sequoia Grove, nan
La Gerla, Brunello di Montalcino	La Gerla, nan
Brovia, Rio Sordo Barbaresco	Brovia, Rocche di Castiglione
Domaine Bertagna, Vosne Romanee 1er Cru 'les beaux mont'	Domaine Bertagna, nan
Cantele, Amativo	Cantele, nan
Artesa, Los Carneros	Artesa, nan
J Vineyards & Winery, J Vineyards	J Vineyards & Winery, Bow Tie Vineyard
Chehalem, Three Vinyard	Chehalem, Stoller Vineyards
Kokomo, Sonoma Coast Gopher Hill	Kokomo, Peters Vineyard Winemaker's Reserve
Philippe Colin, Chassagne Montrachet 1er Cru 'Vergers'	Philippe Colin, nan
Neil Ellis, nan	Neil Ellis, nan
La Chablisienne, Chateau Grenouilles Grand Cru	La Chablisienne, Les Preuses Grand Cru
Archery Summit, Premier Cuvee	Archery Summit, Renegade Ridge Estate
Ceretto, Barbaresco	Ceretto, Brunate
Altesino, Brunello di Montalcino Riserva	Altesino, Montosoli
Marchesi Antinori, Antinori Badia a Passignano Riserva Chianti Classico	Marchesi Antinori, Badia a Passignano Riserva
Louis Latour, Vosne Romanee	Louis Latour, Les Damodes

Table C.1: Random sample of matching designations for a Jaro-Winkler threshold of 0.1

C.2 t = 0.5

Wine designation	Review designation
Alta Vista, Premium	Alta Vista, Premium
Domaine Jacques Prieur, Puligny Montrachet 1er Cru les Combettes	Domaine Jacques Prieur, Les Combettes Premier Cru
Leyda, Single Vineyard	Leyda, Reserva
Albert Bichot, Nuis Saint Georges Premier Cru Chateau Gris Monopole	Albert Bichot, Les Chabiots Premier Cru
Olivier Leflaive, Saint Aubin 1er Cru 'Charmois'	Olivier Leflaive, nan
Louis Latour, nan	Louis Latour, nan
Mastrojanni, Brunello di Montalcino Schiena d'Asino	Mastrojanni, nan
Renato Ratti, Barolo Rocche dell'Annunziata	Renato Ratti, Rocche dell'Annunziata
Albino Rocca, Barbaresco Vigneto Loreto	Albino Rocca, Ovello Vigna Loreto
Produttori del Barbaresco, Langhe	Produttori del Barbaresco, Montestefano Riserva
Louis Jadot, Chablis Grand Cru 'Preuses'	Louis Jadot, Les Pucelles Premier Cru
Chakana, Reserve	Chakana, Maipe Reserve
Vietti, Barolo 'Rocche di Castiglione'	Vietti, Rocche di Castiglione
Olivier Leflaive, Puligny Montrachet 1er Cru 'Les Folatieres'	Olivier Leflaive, En Remilly Premier Cru
Marques de Grinon, Pertit Verdot	Marques de Grinon, Estate Bottled
Chehalem, Three Vinyard	Chehalem, Stoller Vineyards
Altesino, Brunello di Montalcino Riserva	Altesino, Montosoli
Artesa, Los Carneros	Artesa, Estate Vineyard
Morgante, Don Antonio	Morgante, Don Antonio
Renato Ratti, Barolo Conca	Renato Ratti, Marcenasco

Table C.2: Random sample of matching designations for a Jaro-Winkler threshold of 0.5

C.3 t = 0.75

Wine designation	Review designation
Balthasar Ress, Hattenheim Risling GG	Balthasar Ress, Hattenheim Schutzenhaus Kabinett
Alta Vista, Premium	Alta Vista, Premium
Trapiche, Oak Cask	Trapiche, Oak Cask
Chateau Ste. Michelle, Ethos Reserve	Chateau Ste. Michelle, Ethos Reserve
Finca Sophenia, Altosur	Finca Sophenia, Altosur
Beyerskloof, nan	Beyerskloof, nan
Santa Carolina, Reserva de Familia	Santa Carolina, Reserva de Familia
Miguel Torres, Santa Digna Gran Reserva	Miguel Torres, Santa Digna Reserve
Chateau Ste. Michelle, Canoe Ridge	Chateau Ste. Michelle, Canoe Ridge Vineyard
Seghesio, Old Vine	Seghesio, Old Vine
Killibinbin, Seduction	Killibinbin, Seduction
Chateau Ste. Michelle, Ethos Reserve	Chateau Ste. Michelle, Ethos Reserve
Casa Castillo, El Molar	Casa Castillo, El Molar
Trimbach, nan	Trimbach, nan
Feudo Solaria, Foglio Cinquanta	Feudo Solaria, Foglio Cinquanta
The Eyrie Vineyards, nan	The Eyrie Vineyards, nan
Campo Viejo, Gran Reserva	Campo Viejo, Gran Reserva
Domaine Jacques Prieur, Clos de Vougeot Grand Cru	Domaine Jacques Prieur, Clos Vougeot
Domaine Laroche, Laroche Chablis Reserve de L'Obedience Grand Cru	Domaine Laroche, La Reserve de l'Obedience Les Clos Grand Cru
Santa Carolina, Reserva de Familia	Santa Carolina, Reserva

Table C.3: Random sample of matching designations for a Jaro-Winkler threshold of 0.75

C.4 t = 0.8

Wine designation	Review designation
Arboleda, nan	Arboleda, nan
Caliterra, Reserva	Caliterra, Reserva
Trapiche, Oak Cask	Trapiche, Oak Cask
Moet & Chandon, Dom Perignon P2	Moet & Chandon, Dom Perignon P2
Emilio Moro, Mallelous de Sanchomartin	Emilio Moro, Malleolus de Sanchomartin
Cono Sur, 20 Barrels	Cono Sur, 20 Barrels
Mount Mary, nan	Mount Mary, nan
Villa Maria, Cellar Selection	Villa Maria, Cellar Selection
Bouchard Finlayson, Crocodile's Lair Kaaimansgat	Bouchard Finlayson, Crocodile's Lair Kaaimansgat
Guicciardini Strozzi, Sodole	Guicciardini Strozzi, Sodole
Mas del Perie, Les Escures	Mas del Perie, Les Escures
Moet & Chandon, Dom Perignon P2	Moet & Chandon, Dom Perignon Brut
Cusumano, Sagana	Cusumano, Sagana Tenuta San Giacomo
Beaulieu Vineyard, Georges du Latour Private Reserve	Beaulieu Vineyard, Georges de Latour Private Reserve
Pio Cesare, Il Bricco Barbaresco	Pio Cesare, Il Bricco
Giant Steps, Sexton Vineyard	Giant Steps, Sexton Vineyard
Campo Viejo, Gran Reserva	Campo Viejo, Gran Reserva
Domaine Jacques Prieur, Clos de Vougeot Grand Cru	Domaine Jacques Prieur, Clos Vougeot
Bodega Noemia de Patagonia, nan	Bodega Noemia de Patagonia, nan
Santa Carolina, Reserva de Familia	Santa Carolina, Reserva de Familia

Table C.4: Random sample of matching designations for a Jaro-Winkler threshold of 0.8

C.5 t = 0.85

Wine designation	Review designation
Alex Gambal, Bourgogne Pinot Noit 'les Deux Papi'	Alex Gambal, Bourgogne Pinot Noir
Valdivieso, nan	Valdivieso, nan
Domaine Jacques Prieur, Echezeaux Grand Cru	Domaine Jacques Prieur, Echezeaux
Georges Vigouroux, Chateau de Merces	Georges Vigouroux, Chateau Pech de Jammes
Fritz Haag, Brauneberger Juffer Sonnenuhr Auslese	Fritz Haag, Brauneberger Juffer Auslese
Morgante, nan	Morgante, nan
Viu Manent, Gran Reserva	Viu Manent, Gran Reserva
Kendall-Jackson, Vintner's Reserve	Kendall-Jackson, Vintner's Reserve
Georges Vigouroux, Chateau de Merces	Georges Vigouroux, Chateau Pech de Jammes
Caliterra, Reserva	Caliterra, Reserva Estate Grown
Chateau Ste. Michelle, Ethos Reserve	Chateau Ste. Michelle, Ethos Reserve
Georges Vigouroux, Chateau de Merces	Georges Vigouroux, Chateau de Haute-Serre Icone Wow
Mazzei, Fonterutoli Chianti Classico	Mazzei, Fonterutoli
Henry of Pelham, Icewine	Henry of Pelham, Icewine
Henschke, Julius Eden Valley	Henschke, Julius
Concha y Toro, Marques de Casa Concha	Concha y Toro, Marques de Casa Concha
Concha y Toro, Casillero del Diablo	Concha y Toro, Casillero del Diablo Reserva
Penfolds, Yattarna	Penfolds, Yattarna
Le Macchiole, Messorio	Le Macchiole, Messorio
Viu Manent, Single Vineyard	Viu Manent, Single Vineyard La Capilla Estate

Table C.5: Random sample of matching designations for a Jaro-Winkler threshold of 0.85

C.6 t = 0.9

Wine designation	Review designation
FEL, nan	FEL, nan
Miguel Torres, Santa Digna Gran Reserva	Miguel Torres, Santa Digna Reserve
Chakana, Reserve	Chakana, Reserve
Terra Andina, Reserva	Terra Andina, Reserva
Fritz Haag, Brauneberge Juffer Spatlese	Fritz Haag, Brauneberger Juffer Spatlese
Finca Sophenia, Altosur	Finca Sophenia, Altosur
Louis Latour, nan	Louis Latour, nan
Arboleda, nan	Arboleda, nan
La Chablisienne, Chateau Grenouilles Grand Cru	La Chablisienne, Chateau Grenouilles Grand Cru
Pratesi, Locorosso	Pratesi, Locorosso
Giant Steps, Applejack Vineyard	Giant Steps, Applejack Vineyard
San Pedro, Epica	San Pedro, Epica
Concha y Toro, Casillero del Diablo	Concha y Toro, Casillero del Diablo Reserve
Concha y Toro, Casillero del Diablo	Concha y Toro, Casillero del Diablo Reserva
Caliterra, Reserva	Caliterra, Reserva
Morgante, nan	Morgante, nan
Emilio Moro, Mallelous de Valderramiro	Emilio Moro, Malleolus de Valderramiro
Henri Bourgeois, Petit Bourgeois Rose	Henri Bourgeois, Petit Bourgeois Rose de
Alpasion, nan	Alpasion, nan
Penfolds, Yattarna	Penfolds, Yattarna

Table C.6: Random sample of matching designations for a Jaro-Winkler threshold of 0.9

C.7 t = 0.95

Wine designation	Review designation
Louis Latour, nan	Louis Latour, nan
Penfolds, Yattarna	Penfolds, Yattarna
Archery Summit, Premier Cuvee	Archery Summit, Premier Cuvee
Rutherford Hill, nan	Rutherford Hill, nan
Marques de la Concordia, Crianza	Marques de la Concordia, Crianza
Jacob's Creek, nan	Jacob's Creek, nan
Louis Latour, nan	Louis Latour, nan
Robert Hall, nan	Robert Hall, nan
Egon Muller, Scharzhofberger Spatlese	Egon Muller, Scharzhofberger Spatlese
Finca Flichman, Misterio	Finca Flichman, Misterio
Columbia Crest, Reserve	Columbia Crest, Reserve
Amayna, nan	Amayna, nan
Kendall-Jackson, Vintner's Reserve	Kendall-Jackson, Vintner's Reserve
Michel Torino, Don David Reserve	Michel Torino, Don David Reserve
Michel Torino, Coleccion	Michel Torino, Coleccion
Penley Estate, Phoenix	Penley Estate, Phoenix
Concha y Toro, Casillero del Diablo	Concha y Toro, Casillero del Diablo
Emilio Moro, nan	Emilio Moro, nan
Landmark, Overlook	Landmark, Overlook
Donnafugata, Lighea	Donnafugata, Lighea

Table C.7: Random sample of matching designations for a Jaro-Winkler threshold of 0.95

Appendix D

Raw data

This appendix contains the raw data of the results obtained in chapter 4.

D.1 Models

D.1.1 Taster specific

k	avg.	std.
1	0.18912689582517203	0.0014523067077875339
3	0.15941375545254052	0.0013027270626956937
5	0.15621290986335434	0.0009652146514093564
10	0.15829555216449895	0.000766458157006716
15	0.15785796122951873	0.0005151330598457043

D.1.2 Global

k	avg.	std.
1	0.20894182181823556	0.003075895477116529
3	0.17853339820327688	0.0018520486819711384
5	0.17232119960303507	0.0007895344077408466
10	0.16881601857684345	0.00026181951615423236
15	0.1696734227130918	0.0003192612791211413

D.1.3 Random

k	avg.	std.
1	0.3392680322827317	0.0007214621206006272
3	0.3403347024868394	0.0023703322127400033
5	0.3396447418992782	0.0014194543568927973
10	0.3399517820433096	0.0014530176335575986
15	0.33988323979274687	0.0011920446698172977

D.2 Jaro-Winkler thresholds

D.2.1 0.7

k	avg.	std.
1	0.1883743348347076	0.0011845494464899633
3	0.1643731203217471	0.00086779551243825
5	0.1574442360758403	0.0005865298090496308
10	0.1588272209073012	0.00021555268069762182
15	0.15950441446929134	0.00040719488621936855

D.2.2 0.8

k	avg.	std.
1	0.19142023463042887	0.003100644573614243
3	0.1645315204293458	0.0014095723850388537
5	0.1627387244994478	0.0003823369357987597
10	0.15982079371450927	0.0007643117042925703
15	0.16082747582161552	0.0010173554557865452

D.2.3 0.9

Same as section D.1.1.

D.2.4 0.95

k	avg.	std.
1	0.1776998785627553	0.0021019444195709016
3	0.16117373260933132	0.001080166526521576
5	0.15967216463244988	0.0009951356665101351
10	0.15807051515849452	0.0005504464832401083
15	0.1581827046701867	0.000251258171390367

D.2.5 0.99

k	avg.	std.
1	0.19251592670276962	0.001921339727132408
3	0.18258517043527187	0.0020593439331325044
5	0.18342574294846437	0.0017935583715740274
10	0.1803901974818608	0.0010939726701274643
15	0.18041894001811323	0.00032586760822430824

D.3 User's history sizes

History size	avg.
483	0.1531696839225115
185	0.19487551532197042
129	0.18319545795366754
118	0.14531222766148558
137	0.14696468006477395
109	0.15338165274787083
69	0.1230695223048074
37	0.15971061067509443
39	0.11432185814365779
31	0.12905976854472506

D.4 Feature sets

D.4.1 Taste related features

Same as section D.1.1.

D.4.2 Extended features set

k	avg.	std.
1	0.18591385169713118	0.0014864039108083878
3	0.16700200433819	0.0016296378660600435
5	0.1610308152470845	0.0008814322729360381
10	0.15704033682168994	0.0007350483041337857
15	0.15636016893785304	0.0005461863163322801

D.5 Reviews dataset results

k	avg.
1	0.19509156839723768
3	0.1687602783082704
5	0.15764858165423817
10	0.15006848606167839
15	0.14850310181802115

References

Aggarwal, C. C. (2016). *Recommender systems: The textbook*. Cham: Springer. doi: 10.1007/978-3-319-29659-3

Boulton, D., & Hammersley, M. (2006). Analysis of unstructured data. *Data collection and analysis*, *2*, 243–259.

Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information systems*, *69*(Supplement 32), 175–186.

Chen, M., & Liu, P. (2017). Performance evaluation of recommender systems. *International Journal of Performability Engineering*, *13*(8), 1246.

Claypool, M., Le, P., Wased, M., & Brown, D. (2001). Implicit interest indicators. In *Proceedings of the 6th international conference on intelligent user interfaces* (pp. 33–40).

Cohen, W. W., Ravikumar, P., Fienberg, S. E., et al. (2003). A comparison of string distance metrics for name-matching tasks. In *Iiweb* (Vol. 3, pp. 73–78).

Cruz, C., Van, C. N., & Gautier, L. (2018). Word embeddings for wine recommender systems using vocabularies of experts and consumers. *Open Journal of Web Technologies (OJWT)*, *5*(1), 23–30.

Deng, D., Li, G., Hao, S., Wang, J., & Feng, J. (2014). Massjoin: A mapreduce-based method for scalable string similarity joins. In *2014 ieee 30th international conference on data engineering* (pp. 340–351).

Feldman, R., Sanger, J., et al. (2007). *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press.

Geluvaraj, B., & Sundaram, M. (2021). An implementation and combining of hybrid and content based and collaborative filtering algorithms for the higher performance of recommended sytems. In *International conference on computing science, communication and security* (pp. 99–111).

Katarya, R., & Saini, R. (2022). Enhancing the wine tasting experience using greedy clustering wine recommender system. *Multimedia Tools and Applications*, *81*(1), 807–840.

Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender systems: issues, challenges, and research opportunities. In *Information science and applications (icisa) 2016* (pp. 1179–1189). Springer.

Lops, P., Gemmis, M. d., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, 73–105.

McCallum, A., & Nigam, K. (1998). A comparison if event models f or naive bayes text classication. In *Proc. aaai-98 workshop on learning for text categorization.*

Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, *1*, 829–838.

Ricci, F., Rokach, L., & Shapira, B. (Eds.). (2015). *Recommender systems handbook* (2nd ed.). New York: Springer. doi: 10.1007/978-1-4899-7637-6

Rosenman, E., Basse, G., Owen, A., & Baiocchi, M. (2020). Combining observational and experimental datasets using shrinkage estimators. *arXiv preprint arXiv:2002.06708*.

Singh, R. H., Maurya, S., Tripathi, T., Narula, T., & Srivastav, G. (2020). Movie recommendation system using cosine similarity and knn. *International Journal of Engineering and Advanced Technology*, *9*(5), 556–559.

Srinivas, K., Gale, A., & Dolby, J. (2018). Merging datasets through deep learning. *arXiv preprint arXiv:1809.01604*.

Stewart, G. W. (1993). On the early history of the singular value decomposition. *SIAM review*, *35*(4), 551–566.

Wei, Y., Wang, X., Li, Q., Nie, L., Li, Y., Li, X., & Chua, T.-S. (2021). Contrastive learning for cold-start recommendation. In *Proceedings of the 29th acm international conference on multimedia* (pp. 5382–5390).

What is open data? (n.d,). *Open Data Handbook*. Retrieved May 1, 2022, from `https://opendatahandbook.org/guide/en/what-is-open-data/`

Zhou, R., Khemmarat, S., & Gao, L. (2010). The impact of youtube recommendation system on video views. In *Proceedings of the 10th acm sigcomm conference on internet measurement* (p. 404–410). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/1879141.1879193` doi: 10.1145/1879141.1879193

Abstract :

Over the past few years, we observed the expansion of the use of recommendation systems in many fields. Even though wine recommendation did not escape this rule, researches for this use-case are limited, due in particular to the lack of comprehensive open datasets on this topic. In this work, we approach this problem by merging two datasets, one containing user reviews and the other containing wine descriptions, in order to evaluate how it affects the performances of wine recommendation systems. To be able to make this evaluation, we designed a wine recommender system, and assessed its accuracy for different parameters. Finally, we looked at how the results are affected, when the model is trained on the merged dataset, compared to when it is trained only on the dataset containing reviews.

Résumé :
Au cours des dernières années, nous avons pu observer une expansion dans l'utilisation de systèmes de recommandation dans de nombreux domaines. Même si le sujet de la recommandation de vin n'a pas échappé à cette règle, les recherches sur le sujet sont limitées, notamment en raison du manque d'ensembles de données ouverts compréhensifs. Dans ce travail, nous abordons ce problème en fusionnant deux jeux de données, l'un contenant des avis d'utilisateurs, et l'autre contenant des descriptions de vins, afin d'évaluer comment cela affecte les performances de systèmes de recommandation de vin. Pour pouvoir faire cette évaluation, nous avons conçu un système de recommandation de vin, et évalué sa précision pour différents paramètres. Enfin, nous avons examiné comment les résultats sont affectés, lorsque le modèle est entrainé sur le jeu de données fusionné, par rapport à lorsqu'il est entrainé uniquement sur le jeu de données contenant des avis.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Louvain School of Management
Place des Doyens, 1 bte L2.01.01, 1348 Louvain-la-Neuve
Boulevard Emile Devreux 6, 6000 Charleroi, Belgique
Chaussée de Binche 151, 7000 Mons, Belgique
www.uclouvain.be/lsm