UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

SUPERVISED PROJECT REPORT

PoC - Deduplication of the disk catalog

Author : Salif SAWADOGO
Specialization : MASERATI - Major Data Science
Academic Year : 2021 - 2022

Reviewer : Anne-Pierre De PEYRONNET

Supervisor: M. Erwan Bargan
Consultant and educator

Septembre 2022

Confidentiality Notice

For reasons of confidentiality, the numbers and real data in this report have voluntarily been suppressed or modified.

Acknowledgments

I would like to express my deepest thanks and gratitude to my supervisor **Mr. Erwan BARGAN** for his availability, valuable pieces of advice and for involving me in the diverse activities that have to do with the work process. His ability to reconcile statistical learning methods, programming and business, and his data communication - storytelling skill have deeply inspired me. It is a great privilege and an honor to work under his supervision.

My gratitude goes also to **Mrs. Nathalie Zalensky** manager of CDandLP for accepting me in their company without major constraint on the use of their company's data.

Furthermore, I would like to thank **Mr. Anne-Pierre De PEYRONNET** for having accepted to review this report. On this occasion, my gratitude goes also to the faculty and administrative staff of MASERATI for the efforts they make day and night to train us.

Abstract

The main objective of this study is to implement a new recommendation system for artists and titles by removing duplicate items and suggesting the best neighboring items by given raw inputs.

Data processing and cleaning was a challenge that followed us throughout the project because good data leads to a better solution. From many experiments, we concluded that using simple textual distances to find nearest neighbors is not optimal and requires a lot of computational resources. We tested multiprocessing to perform parallel computations and reduce the execution time, but without success.

We therefore designed a solution based on unsupervised nearest neighbors from scratch. A few experiments allowed us to design a country-based and a word-based blocking strategy to minimize the amount of data to be used for matching. The word-based strategy is the one we chose. Nevertheless, we keep our country-based solution for other uses because it can be useful for other projects. To capitalize our work and the solution for others, we used Docstring and Sphinx.

Key Words: Unsupervised learning, text distances, nearest neighbors , recommendation, Sphinx.

Contents

Introduction

The advent of digital technology has resulted in an increasing amount of data generated by users and individuals. This collected data, comparable to crude oil, once processed and analyzed can be used as a source of growth, marketing or decision support. Hence, thanks to the data collected by Discogs, a website and collaborative database of music recordings, CDandLP decided to clean up its internal database and match it to Discogs 's data.

Indeed, many elements may be incorrectly filled in or written with additional information. This results in an imperfect match with the CDandLP data. Therefore, to enable efficient search and recommendation using only artists and titles, I have been working on the proof of concept (PoC) to illustrate the feasibility and business opportunity of the project. It has been demonstrated that a recommendation system can help a business increase average order value, drive traffic, engage customers, and increase the number of items per order.

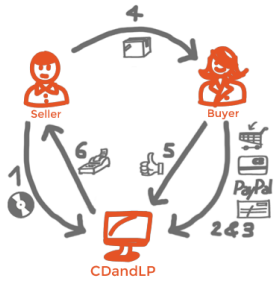
For more coherence in this report, we will first present the context of this project and the expected results, then we will support the theoretical elements necessary to understand the project. Finally, we will highlight the solution created and the different experiments carried out.

1 Presentation of the business CDandLP and the project's context

In the first part of this report, I will present the context in which I carried out my internship. To do so, I will introduce you CDandLP, then I'll make focus on our project.

1.1 CDandLP

To join international buyers and sellers of new and used records and CDs, two vinyl record enthusiasts decided to launch a website in 2001, at the dawn of the Internet era. CDandLP was created. It swiftly rose to prominence as one of the first music-specific markets.



Due to the enthusiasm of the collectors, the marketplace for private and professional sellers continued to expand. CDandLP is a marketplace. Its missions are the following:

— To put in relation the buyers and the salesmen,

— To take care of the collection of each transaction,

— Ensure that buyers have received a follow-up and delivery of your order.

Indeed, CDandLP does not keep any sensitive data and uses Secure Socket Layers (SSL) to protect all payments. It ensures that the transaction will proceed smoothly for each party: They conduct the appropriate checks on buyers' profiles to ensure sellers' payments, and we postpone paying sellers until buyers have acknowledged receipt of the order. They guarantee a satisfactory shopping experience and As a trusted third party, guarantee that buyers receive their order and sellers get their money back.

1.2 How data are collected?

As shown in the orange dotted box (figure 1), to sell or buy an item on CDandLP, you will have to use the search engine by mentioning the key words which are the name of the artist, the title, the pressing or the label. Once you have filled in the information, a list of items will be recommended.

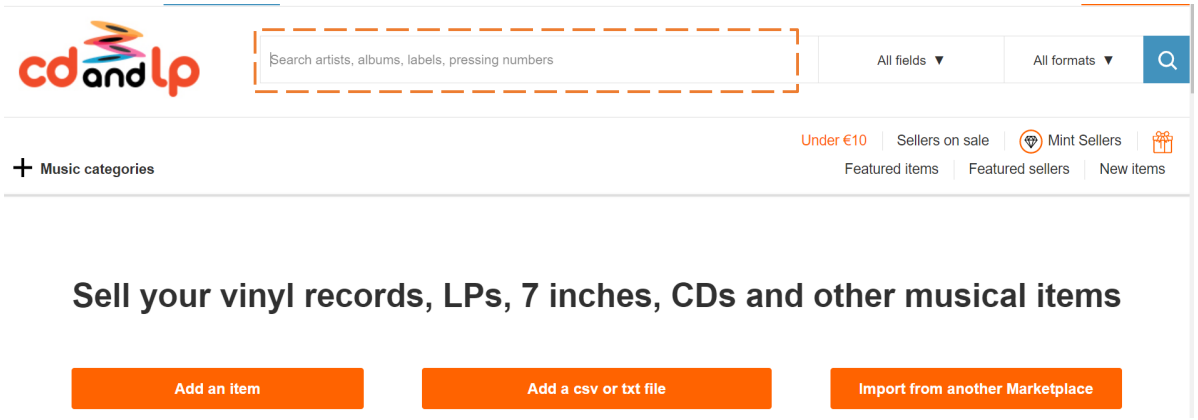


Figure 1 – Selling on cdandlp - webpage

1.3 Objectives

A **Proof Of Concept** is usually called POC. It is an experiment that seeks to evaluate a solution or idea's worth, relevance, and/or viability before it is put into practice. It is frequently used in data science to test a use case.

Our PoC is called **DEDUPLICATION OF THE DISK CATALOG**.

Without specific goals, data science projects usually end up in the trash. This is why we defined our main objectives with this work. These are the following.

— Define the optimal strategy for deduplication and suggest the most similar elements
— Build the foundation for technical scalability (technology and advanced analytic algorithms) for the upcoming machine learning project.

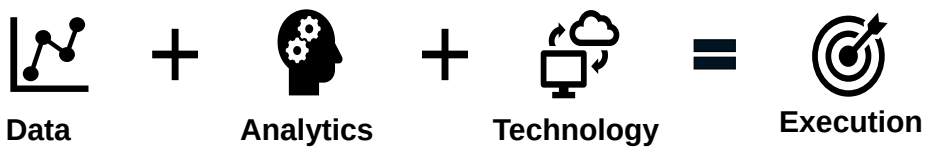


Figure 2 – Ingredients used for missions

Offering the best items to users is very important for a very digital business. To understand what we are talking about, we can refer to the figure 3 (below). For example, we performed a search with the name of the artist **Bob Marley**. We got several suggestions, as shown in the orange dotted box on the right side of figure 3.

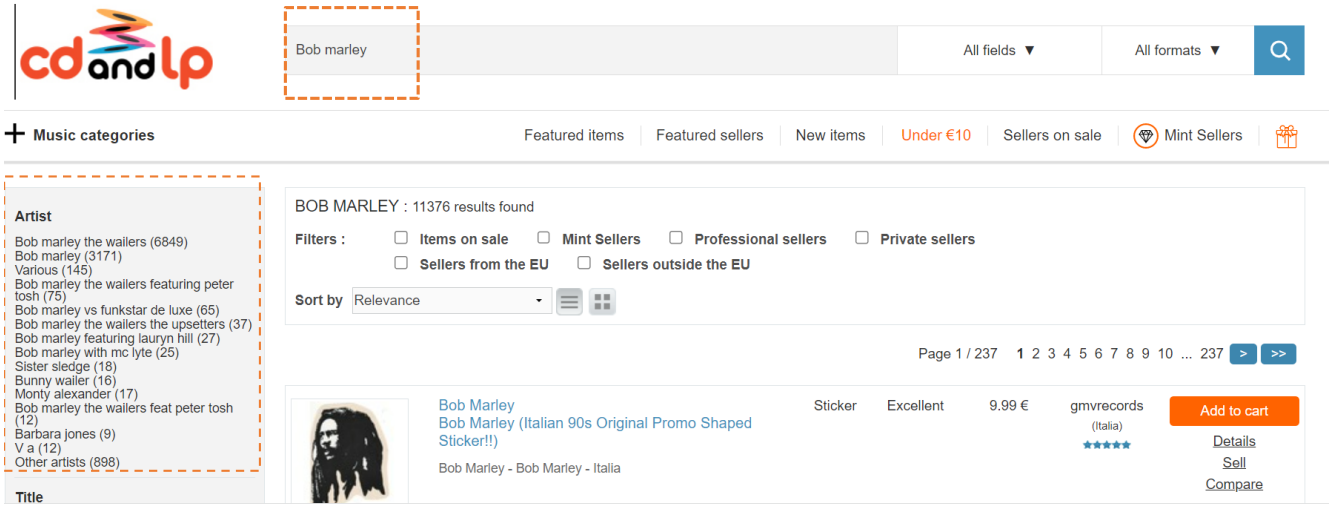


Figure 3 – Exemple of recommandation - CDandLP webpage

Our project is the beginning of the sophistication of the recommendation used. For this, we will use advanced techniques to prevent **BOB MARLEY** and **BO MARLE** from being very different for the algorithm. Using text mining techniques, we will first standardize the textual data and then measure the degree of similarity.

In order to take advantage of **cloud computing** and very powerful computing capacity, we have decided to use **Google Colab**. Cloud computing is the provision of computing services (including servers, storage, databases, networking, software, analytics, artificial intelligence) via the internet in order to provide faster innovation, flexible resources and economies of scale. For this project, we use a free account.

1.4 Roadmap

To carry out this study, planning was a crucial point to be well defined. The roadmap of the project is divided in several phases which sometimes overlapped (figure 4) .

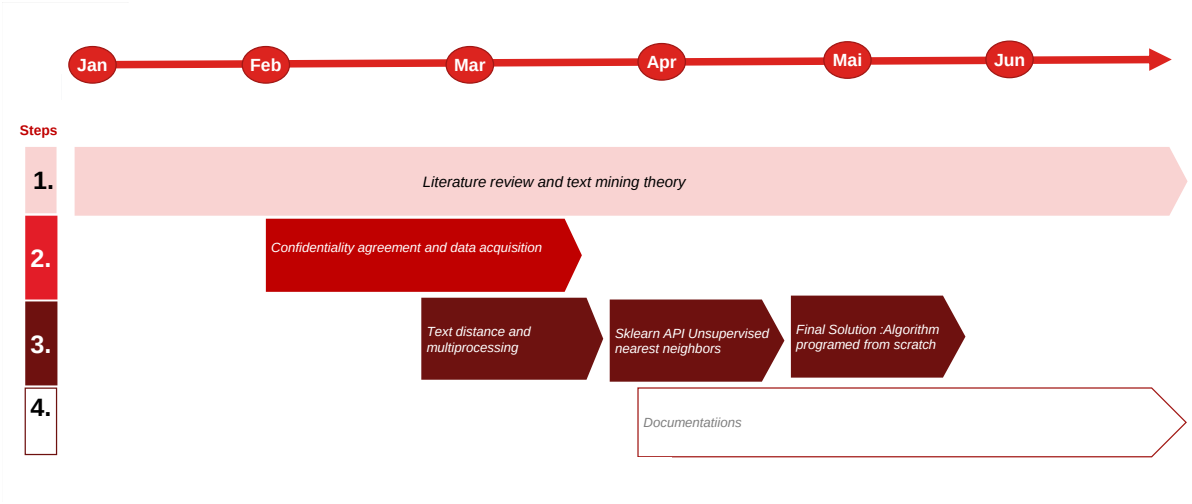


Figure 4 – Our PoC Roadmap

- Phase 1 : Literature review.

- Phase 2: Data acquisition and confidentiality.

- Phase 3: Experimentation and mplementation of the solution.

- Phase 4: Project documentation.

As a scholarship student without the status of student worker, I realize this project between January and the end of May because apart from the days of course sessions, the other days were entirely devoted to the realization of this PoC.

2 Theoretical elements

This paragraph is primarily intended to introduce measures of similarity and distance of texts. Second, it summarizes the pre-processing steps of the textual data. Finally, we'll present a machine learning algorithm used: unsupervised nearest neighbors.

2.1 Textual data similarity measurement

Distance measure and similarity measure are sides of the same coin. They determine the degree of similarity between two strings. First of all, why did we say that similarity and distance go together? Indeed, the smaller the distance between two textual data, measured by a distance measure, the better the agreement. To this distance, we associate a similarity measure. It is between zero and one, one indicating a perfect agreement of the chains and zero no agreement. To formalize the different measures, we will use the following notations.

— s1 is string 1 and s2 is string 2

— n is the length of the shortest chain of string

— m is the length of the longest chain of string

— i is the letter at position 1 in the string 1

— j is the letter at position j in the string 2 There is a multitude of similarity and distance measures in the literature. We will limit ourselves to measures that are based on a comparison of characters.

2.1.1 Levenshtein distance

It is the most used measure. It is defined as the smallest number of update operations required to transform s1 into s2. That is why this method is considered as an edition method. The edits concern insertions, deletions, and substitutions of characters. In its basic form, each editing operation used is equalized to a unit cost. The Levenshtein distance is equal to:

$$lev_{s1,s2}(i,j) = \begin{cases} max(i,j) & \text{if} \quad \min(i,\, j)=0 \\ min \begin{cases} = lev_{s1,s2}(i-1, j+1) \\ lev_{s1,s2}(i, j-1+1) \\ lev_{s1,s2}(i-1, j-1) + [s1_i s2_j] \end{cases} & \text{else} \end{cases} \tag{1}$$

.

- Deletion of the letters N and I respectively $\rightarrow count = 1 + 1 + = 2$
- Respective insertions of I, N and S $\rightarrow count = 1 + 1 + 1 = 3$

This distance is between max (| s1|, | s2|) and abs (| s1|-| s2|). The first one is the maximum distance. So in the example of the two strings used, the maximal cost is 6 and the minimal cost 1.

Mathematically, it is shown that the Levenshtein distance is a real metric, as it satisfies the triangle inequality . Furthermore, the triangle inequality can be used to filter when determining the best match of a string dictionary, by selecting strings as "pivots". To transform the Levenshtein distance metric into a similarity measure, the distance must be divided by the upper bound on the Levenshtein distance, which is the length of the longest string2 and subtracted by

2.1.2 Damerau-Levenshtein Distance

This distance is a modified version of the Levenshtein distance. Indeed, it allows in addition to Levenshtein editions, transpositions of adjacent characters.

2.1.3 Hamming Distance

The Hamming distance between two strings of the same length is the number of positions in which the corresponding symbols differ. In other words, it measures the minimum number of substitutions needed to convert one string into another, or the minimum number of possible errors when converting one string into another. In a more general context, the Hamming distance is one of several string metrics

for measuring the edit distance between two sequences. It is named after the American mathematician **Richard Hamming**. Below is its pseudo algorithm.

Function hamm(s1, s2):

counter= 0

for k in range 0 and n :

If the k^{th} letter of s1 is different from k^{th} letter of s2 :

Increment counter by 1

End if

End for

Return counter

End

For example, let's be "JAPON" and "SAVON" two character strings. In this case, the Hamming distance between "JAPAN" and "SAVON" is equal to 2 and the similarity is equal to 0.6.

2.1.4 Jaro Distance

The Jaro distance is mainly applied to corresponding person names and short addresses. It is not a true distance metric, because it does not obey the triangle inequality. In this metric, strings are more similar if they have matching characters in a specific range, based on the length of the longest string. There is a penalty if the matching characters are transposed (i.e., not in the same order).

The similarity measure is calculated as follows :

$$\text{simJARO}_{s1,s2}(i,j) = \begin{cases} 0 & \text{if} \quad \text{c} = 0 \\ \frac{1}{3}\frac{1}{3}\left(\frac{c}{|s1|} + \frac{t}{|s2|} + \frac{c-t}{c}\right) & \text{else} \end{cases} \quad (2).$$

Where c is the number of matching characters and t is the number of matching characters that is transposed. Two characters are defined as matching if they are equal and positioned in the position interval of 0 to $\frac{max(|s1|,|s2|)}{2} - 1$ from each other

2.1.5 Jaro-Winkler Distance

The Jaro-Winkler distance is nothing but an extension of the previous distance. Indeed, the similarity associated with it is equal to the similarity associated with the Jaro distance, plus a score if the prefixes of the chains are equal. Winkler in 1999 assumed that errors tend to occur less often at the beginning of the chains [1].

The new robust distance measure should be written :

$$SimJAROWinkler = SimJARO + (\ell * \rho * (1 - SimJARO)) \tag{3}$$

Where ℓ is the number of characters equal to the beginning of the strings with a maximum often set to four characters, ρ is a parameter to be regularized. By default, it is set to 0.1.

2.1.6 N-grammes

We call n-grams substrings of length n. Here the method is based on a comparison of the substrings. We then try to know if each N-gram of s1 exists in s2. The similarity is equal to :

$$SimNgram = \frac{1}{N-n+1} \sum_{I=1}^{N-n+1} h\left(i\right) \tag{4}$$

— $h\left(i\right) = 1$ if the n-element subsequence starting from position i in s1 appears in s2, else $h\left(i\right) = 0$;

— N-n+1 = number of subsequences with n elements in s1.

Example : let us fix n=2 for s1= "JAPON" -and s2 = "SAVON".

The substrings are written respectively: JA, AP, PO, ON, vs SA, AV, VO, ON

$SimNgram = \frac{1}{N-n+1} \sum_{I=1}^{N-n+1} h\left(i\right) = \frac{1}{4} * (0+0+0+1) =$ **0,25**

2.1.7 Others measures

The Jaccard Similarity and the overlap coefficient are based on the intersection of the characters common to the two strings s1 and s2. The first reports the intersection to the union of the characters in s1 and s2 while the second reports only by the minimum of the size of s1 and the size of s2.

$Simjaccard = \frac{|s1 \cap s2|}{|s1 \cup s2|}$

$SimOverlap = \frac{}{|s1 \cap s2|} min(|s1|, |s2|)$

We also define a similarity based on the average size of the two strings. In this case, we talk about the Sørensen-Dice index, or the Dice coefficient.

2.2 Text standardization in Natural Language Processing

Before performing any similarity measure computation, pre-processing is done in order to have good performances. The preprocessing step is well known in the field of Natural Language Processing (NLP). This field is one of the most active research areas in data science today. It is a combination of statistical learning and linguistics. The study of similarity measures in a NLP context is the penultimate step. The last step is the actual modeling. As an example, from which similarity score can we declare two strings s1 and s2 equivalent? The first step which is the pre-processing can be divided into six successive steps which will be discussed in the following lines.

— Language detection

It is obvious that a direct comparison of two strings from different languages is not meaningful. It is then necessary to check the language used before applying the analyses. This is why this step is considered to be the first one. Language detection is possible thanks to open-source libraries. The best known is the **Google Compact Language Detector cld3**.

— Removal of punctuation and special characters

— Tokenization

We are now at the third step. Tokenization is the process of breaking down text into words, punctuation marks, or numeric digits. More roughly, it can be defined as breaking down sentences into words. In this sense I can break down the sentence "I am a student" into words as follows "I" "am" "a" "student".

— Removing stopwords or empty words

Stopwords are words in any language that do not add much meaning to a sentence. They are called stopwords because their absence does not profoundly change the meaning of the text. They are usually the most common short words, such as "the", "is", "to", "who" and "about".

— Stemming

Stemming follows the stopword deletion step. It allows a word to be reduced to its "root" form. The aim of stemming is to group together numerous variants of a word as a single word. For example, when we apply stemming to "student" or "students" or "studying", we will find a single radical. Stemming techniques are numerous in the literature, however one of the most well-known and commonly used is the Snowball Stemmer [2].

— ASCII folding

This is a filter that consists of converting alphabetic, numeric and symbolic characters that are not in the basic Latin Unicode block (the first 127 ASCII characters) into their ASCII equivalent, where applicable. Example: "à" becomes "a". This step completes the textual preprocessing.

Once the data is extracted and ready to be used, statistical modeling takes place.

2.3 Unsupervised K-nearest neighbor (KNN)

The K-nearest neighbor (KNN) algorithm uses "feature similarity" to find the most similar new data points, which further means that the new data point will be assigned a value based on its correspondence with the points in the training set.

Unsupervised Nearest Neighbors has 3 different algorithms **BallTree** ,**KDTree**, and **brute-force**. These algorithms have been implemented on Sklearn API [3].

2.3.1 Brute Force

This is the unoptimized version of the unsupervised algorithm. Fast nearest neighbor computation is an active research area in machine learning. The most basic implementation of neighbor search involves brute-force computation of distances between all pairs of points in the dataset: for N samples in D dimensions, this approach has a scale of $\mathcal{O}(DN^2)$. Efficient brute-force neighbor searches can be very competitive for small data samples. However, as the number of samples N increases, the brute force approach quickly becomes infeasible

2.3.2 KD Tree

To address the computational wasteful aspects of the brute constrain approach, a assortment of tree-based information structures have been designed. In general, these structures attempt to reduce the required number of distance calculations by efficiently encoding the aggregate distance information for the sample. The essential thought is that on the off chance that point A is exceptionally far off from point B , and point B is exceptionally near to point C , at that point we know that focuses A and C are exceptionally removed, without having to expressly calculate their remove . In this way, the computational taken a toll of a closest neighbor look can be diminished to $\mathcal{O}(DN\log(N)$ [4] or improved . This is often a critical change over brute-force for huge dataset.

Although the KD tree approach is exceptionally quick for dataset with low features (D<20) , it gets to be wasteful for high value of D : **curse of dimension**.

2.3.3 Ball trees

Faced with the inefficiency of KD trees in higher dimensions, ball tree data was developed. However, it is an algorithm that is very dependent on the data structure. It does not work on sparse data. Ball trees recursively divide data into nodes defined by a centroid C and a radius r ; so that each node point lies in the hyper-sphere defined by r and C. Hence it is a very fast algorithm.

2.4 Text Vectorization

In simple terms, this step consists of the transformation of word lists into number vectors. Contrary to the traditional approach which uses similarity and distance measures, we will use here methods adapted to large volumes.

This is the moment when we prepare the text for the algorithm.

Among a large number of vectorization methods, we will limit ourselves here to one-hot, frequency and TF-IDF encodings. They are the easiest to understand and are the most used in the first NLP applications to be developed. More solutions are presented

These three methods are in fact limited to **Bags of Words** (BoW) or, if one decides to tokenize by n -grams, by **Bag of n -grams** (BoN). We make in all the cases the assumption that the tokens or n -grams are distributed in an independent way. The presence of a token or n -gram, denoted B , does not influence the probability of appearance in the document of another one, denoted A , i.e. P(A|B)=P(A) . This implies that documents containing the same tokens (or n -grams) in opposite directions will be considered as identical.

2.4.1 one-hot encoding and term frequency

The one-hot encoding creates vectors of 0 and 1 for each document in vectors whose dimension is determined by the vocabulary of the corpus: a 0 is associated with the absence of a token in a document that is present in the corpus and a 1 indicates its presence also in the document.

The figure 5 illustrates this method by assuming 3 vectorized documents (each composed of a single sentence)

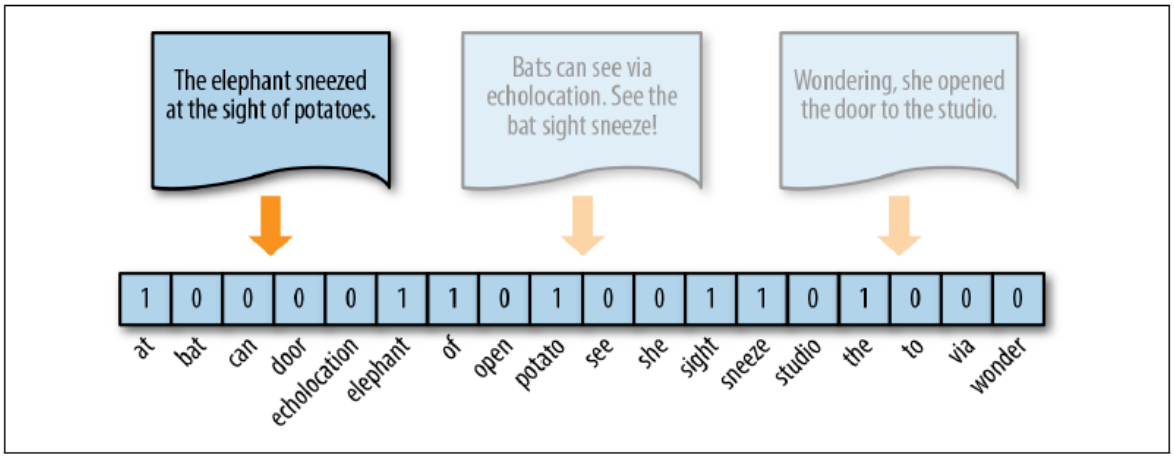


Figure 5 – Example of one hot encoding

When the values 1 of the one-hot encoding are replaced by the number of appearances of the token in the document: this is called term frequency

2.4.2 Term frequency - inverse document frequency (TF-IDF)

TF-IDF vectorization gives each word a weight according to its originality and importance in the document.

More precisely, this real value is calculated for a token as the product of the **term frequency (TF)** by the **inverted document frequency (IDF)**.

In a document, for a token, TF is equal to the number of times it appears in this document and IDF is calculated by the formula :

$$IDF(\text{token}) = log\left(\frac{m}{DF(\text{token})}\right),$$

where m is the total number of documents in the corpus and $DF(\text{token})$ is the number of documents that contain the token at least once, that is :

$$TF\text{-}IDF(\text{token}) = TF(\text{token}) + log(m) - log(DF(\text{token})).$$

— $tf(t, d)$: The more a word is found in a document, the more it influences the meaning of the document, which will be useful to compare it to corpora.

— $idf(t, D)$:The less a word is used in the corpus, the more important it will be to evaluate the similarity between two documents where it is found.

The figure 6 below illustrates this method with 3 documents in English.

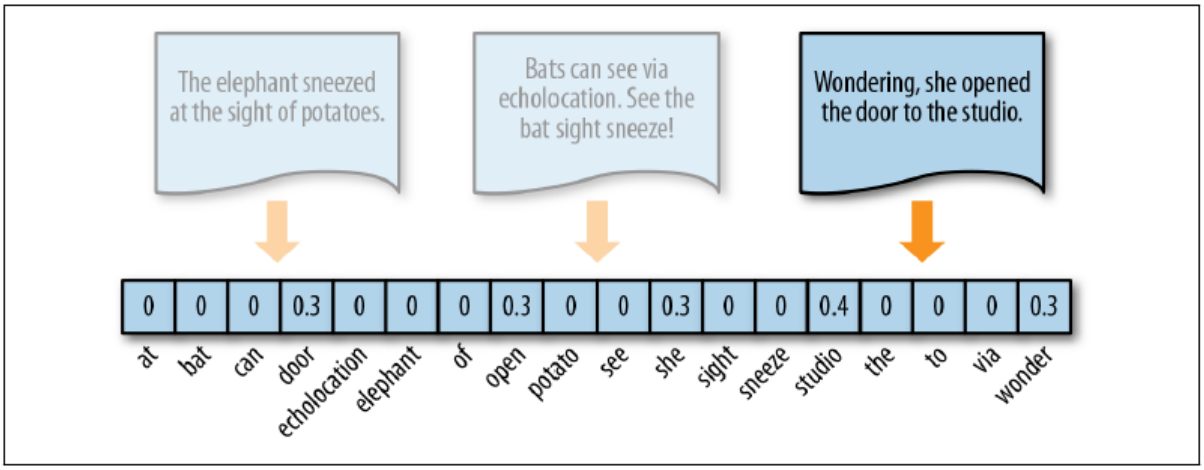


Figure 6 – Example of TF-IDF vectorization

3 Main results and Solution

3.1 Experimentations

3.1.1 Potential quality constraints and scope

The company provides us with two databases. The first one is the discogs database with 9 023024 rows and the last one is the cdandlp database with 1 000 000 rows. Our scope is the following:

— on articles/items for which we do not know the correspondence on the discogs

— on articles where a sql query can be used to perfectly match discogs and cdandlp data

— go with rows with at least assigned value of *(n_ref artist, title*

Only 9 % of languages capture 70 % of titles in our data.We used pycld3 for the language predicti with 67% realibility.

Table 1 – Language distibution in our data

Language	**percentage**
English	33.1
French	20.3
Italian	3.1
Latin	3.1
Spanish	.5
German	2.4)
Catalan	2.2)
Norvegian	1.9)

We tested three library for text distance :**fuzzy**, **TextDistance** , **python-Levenshtein**. Python-Levenshtein is a very fast distance and similarity computation library based on Cpython.

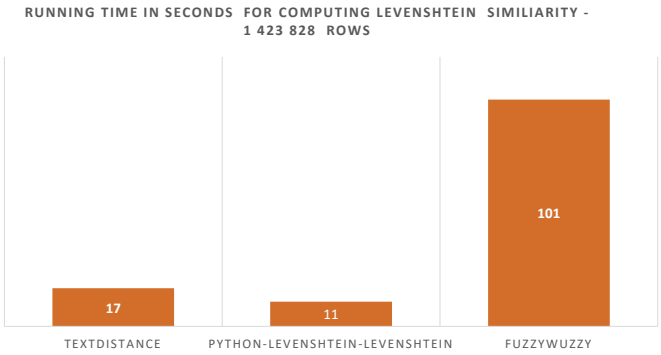


Figure 7 – Running time in second - text distance library

3.1.2 Parallelization

Pandarallel is a straightforward and efficient tool for parallelizing Pandas operations across all available CPUs. We decided to test it for text standarization via function **Pandas.apply** (For more details , see section 2.2.)

Pandas.apply allows users to pass a function and apply it to every value in the Pandas series.

On Linux and macOS, there are no special requirements to install pandaparralle. However, on Windows, pandarallel will only work if the Python session is run from the Windows Subsystem for Linux (WSL).

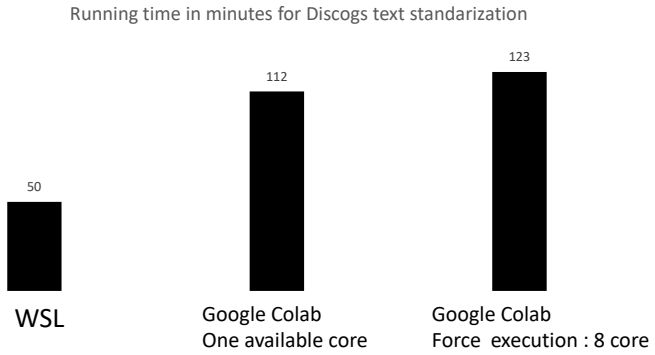


Figure 8 – Running time in second - Pandaparallel

Reassuringly, the parallelization is better on WSL on its own despite the limited RAM resource of 8 much lower than google colab. The figure 8 below illustrates our discovery. Especially since the company server cdandlp is under linux. We have to code the apply operations with Pandaparallel because during the production, the execution time will be significantly reduced.

3.1.3 Sklearn API for unsupervised learning

We use brute-force because it require parse matrix while balltree and kdtree need dense matrix (even these algorithms are the fastest) This algorithm is a successive sequence of several treatments.

1. Vectorise train dataset by bigram,
2. Choose an input as query (Arstist + Title or one of both)

3. Standardize the input,
4. Load the trained Vector (from baseline)
5. Fit input to the trained vector
6. Compute distance Manhattan
7. Find the 20 nearest text

To make it simple, let's take the example : input Arstit = **bob marley** and title = **uprising**. The closest matches suggested by the algorithm are showed in the figure 9. The first row in green was the most similar.

query = 'bob marley uprising'

master_id	text	text_CLEAN	distance
875634	Bob Marley Uprising Live!	bob marley uprising liv	0,40
0,66044	Bob Marley & The Wailers Uprising	bob marley the wailer uprising	0,93
0	Bob Marley Bob Marley 3	bob marley bob marley 3	1,06
09,1402700,1265582,863615	Bob Marley Bob Marley	bob marley bob marley	1,06
1112183	Bob Marley Bob Marley 2	bob marley bob marley 2	1,06
0,891730,79398	Bob Marley Marley	bob marley marley	1,12
1112183	Bob Marley Marley 2	bob marley marley 2	1,12
0	Bob Marley Uprising Tour 1980	bob marley uprising tour 1980	1,16
0	Bob Marley Bob Marley Vol 2	bob marley bob marley vol 2	1,24
0	Bob Marley Bob Marley Vol. 1	bob marley bob marley vol 1	1,24
0	Bob Marley Bob Marley – Vol. 1	bob marley bob marley - vol 1	1,24
0	Bob Marley Bob Marley Live	bob marley bob marley liv	1,25
1064836	Bob Marley Bob Marley Collection	bob marley bob marley collect	1,28
0	Bob Marley Only ! Bob Marley	bob marley only bob marley	1,30
0	Bob Marley Bob Marley And Friends	bob marley bob marley friend	1,30
0	Bob Marley Bob Marley & Friends	bob marley bob marley friend	1,30
0	Bob Marley Exitos de Bob Marley	bob marley exitos bob marley	1,31
488432	Bob Marley Forever Bob Marley	bob marley forev bob marley	1,31
0	Bob Marley Legends - Bob Marley	bob marley legend bob marley	1,31
0	Bob Marley Essential Bob Marley	bob marley essential bob marley	1,32

Figure 9 – The 20 nearest neighbors of "Bob Marley uprising" - Sklearn

With this experimentation we learn that cosine distance does not perform much better than Euclidean distance for sparse data with a large number of points, but it is at least faster, which is a concern since the data sets I work with are large. Theoretically, to get away from the curse of the dimension, the Manhattan distance is the least bad among the most used distances.Manhattan is faster than the cosine distance (3 times according to our results)

Among our first experiments, the running time of the algorithm on all cdandlp requests was 21h 48min. More specifically, it takes 0.78 second per request

3.1.4 An algorithm Knn from scratch

In order to have a perfect control of our algorithm and to be able to optimize our calculations, we have experimented to code a knn alogorithm starting from nothing.

1. Vectorise train dataset by bigram via tf-idf,
2. Choose an input as query (Arstist + Title or one of both)
3. Standardize the input,
4. Load the trained Vector and **Blocking strategy on X_train dimension**
5. Fit input to the trained vector

6. Compute **cosine similarity**
7. Find the 20 nearest items
8. **compute knn for the overall dataset by using *map* and *partial* functions from functools library**

We use here a similarity measure and not distance to discriminate the items. The average running time for the overall dataset is 4 days 7 hours. More specifically, it takes 3.75 seconds per request.

The figure 10 below illustrates the results obtained for a **Bob Marley Is this Love** query. The first two lines have the highest possible similarity measure : 100

query = 'bob marley Is this Love'

master_id	text	text_CLEAN	similarity
0	Bob Marley Is This Love?	bob marley is this lov	100
0	Bob Marley Is This Love	bob marley is this lov	100
0,168601	Bob Marley & The Wailers Is This Love	bob marley the wailer is this lov	82,92
168601	Bob Marley & The Wailers Is This Love (Live)	ob marley the wailer is this lov li	79,69
0	Bob Marley Bob Marley- One Love	bob marley bob marley one lov	79,20
1440538	Bob Marley vs. Mr. K Is It Love	bob marley mr k is it lov	76,76
1112183	Bob Marley Bob Marley 2	bob marley bob marley 2	76,63
0	Bob Marley Bob Marley 3	bob marley bob marley 3	76,63
402700,1265582,86	Bob Marley Bob Marley	bob marley bob marley	76,63
0	Bob Marley Love Songs	bob marley lov song	75,77
5097,1339248,290	Bob Marley One Love	bob marley one lov	75,43
0	Bob Marley Bob Marley – Vol. 1	bob marley bob marley - vol 1	75,03
0	Bob Marley Bob Marley Vol. 1	bob marley bob marley vol 1	75,03
0	Bob Marley Bob Marley Vol 2	bob marley bob marley vol 2	75,03
0	Bob Marley Bob Marley Live	bob marley bob marley liv	74,74
0,935628	Bob Marley Love Life	bob marley lov lif	74,31
1354349	ley The Bob Marley Collection: Bob Marley - V	the bob marley collect bob marl	74,30
401463	ley The Bob Marley Collection: Bob Marley - V	the bob marley collect bob marl	74,30
401469	ley The Bob Marley Collection: Bob Marley - V	the bob marley collect bob marl	74,30
0	Marley I Love Bob Marley (Classic Mixes) (Volu	rley i lov bob marley classic mix v	74,12

Figure 10 – The 20 nearest neighbors of "Bob Marley Is this Love" - From scratch

3.1.5 Blocking strategies by country

	index	country
0	Us	1838701
1	Uk	897272
2	Germany	674060
3	France	423732
4	Italy	350008
...	...	...
189903	Uk ¤ Canada ¤ Us ¤ Australia ¤ New Zealand ¤ G...	1
189904	Germany ¤ Us ¤ Scandinavia ¤ Canada ¤ Italy ¤ ...	1
189905	Us ¤ Uk ¤ Canada ¤ Netherlands ¤ New Zealand ¤...	1

Figure 11 – How is the feature country recorded in our cdandlp table?

The implementation of the algorithm from scratch gives us the opportunity to experiment with possible improvements: reduce the two data tables before comparing them. We tested two blocking strategies. The first is blocking by **country**, which amounts to comparing cdandlp items from country x and looking for their correspondence in the discogs database belonging to country y. the results were mixed even if the approach was engineers, nevertheless we kept it for later studies. If the solution did not work, we suspect the quality of the data used. The expression **garbage in garbage out** would make sense here. Indeed, we have recoded the country variable based on our common sense of the available data. As shown in the figure 11(table) below, it is certain that countries such as UK France Italy have been repeated several times once for certain items. Nevertheless, the last items seem to come from several countries. In the blocking strategy, we have decided that

item 189903 comes from United Kingdom, USA, Canada, Australia, New Zealand... However, nothing guarantees that. This way of proceeding could wrongly eliminate or include items that do not belong to a country to register there.

When we have a France strategy in fig 12, and we are looking for the 4 closest neighbors of Patrick Moraz (the first line being the query), we have obtained neighbors who have nothing to do with Patrick Moraz knowing that they exist elsewhere.

	id_query	master_id	artist	title	similarity	rank	levenshtein	jaro_winkler
0	117183052	117183052	Patrick Moraz	Future Memories : Patrick Moraz Live On Tv	-1.0	0	1	1
1	117183052	1306989	Patrick Fiori	4 Mots	65.678958	1	0.485714	0.692696
2	117183052	0	Tripis	Crazy Memories	65.678958	2	0.405797	0.622004
3	117183052	0	Patrick Forgas	Monks	63.564173	3	0.463768	0.828023
4	117183052	0	Patrick Fiori - Patrick Bruel	Corsica	63.157959	4	0.581395	0.822409

Figure 12 – Blocking strategy by country - outputs

3.1.6 Blocking strategy by words

Everything here starts with the creation of a data dictionary. This one is created as follow :

```
baseline['text_2'] = baseline['text_CLEAN'].str.title()
# stop_words = ['and', 'et']
# dict_words = {}


for id_row,text in enumerate(baseline['text_2']):
    words = text.split()
    words = [ w for w in words if w not in stop_words ]
    words = set(words)
    for word in words:
        if word not in dict_words:
            dict_words[word] = [ id_row ]
        else:
            liste_temp = dict_words[word]
            liste_temp.append(id_row)
            dict_words[word] = liste_temp

 with open(REP_INTERMED + 'dict_words.pkl', 'wb') as f:
        pickle.dump(dict_words, f)

with open(REP_INTERMED + 'dict_words.pkl', 'rb') as f:
        dict_words = pickle.load(f)
```

Figure 13 – Blocking strategy by words - dictionary

The idea of process in this code is to browse discogs dataset and create a dictionary of words. So let's take a given item *Bob Marley is this love?*.With this process, we are able to find the indexes of the different items that **contain at least one word** of *Bob Marley is this love?*

To leverage this solution to optimize the running time: - we iterate each row of cdandlp data and we retrieve the discogs subset of potential candidates suggested from the baseline word dictionary.

- The matching will be between each cdandlp query and the discogs subset. The rest of the procedure remains identical.

As the table in the figure 14 below shows, the results are very reassuring and are better than the blocking strategy 12.

	id_query	master_id	artist	title	similarity	rank	levenshtein	jaro_winkler
0	117183052	117183052	Patrick Moraz	Future Memories : Patrick Moraz Live On Tv	-1.0	0	1	1
1	117183052	18058,0	Patrick Moraz	Future Memories	88.396	1	0.675325	0.901961
2	117183052	18025	Patrick Moraz	Patrick Moraz	84.407222	2	0.692308	0.868845
3	117183052	18044	Patrick Moraz	Future Memories Ii	82.686887	3	0.725	0.913725
4	117183052	0	Patrick Moraz	Patrick Moraz 1,2	82.591801	4	0.707317	0.852546

Figure 14 – Blocking strategy by words - outputs

3.2 Final solution

The previous experiments allowed us to optimize our algorithm from scratch because we used word blocking and multiproccessing to browse the set of queries. Pandaparralel was chosen for the standardization of the textual data. We have retained a solution of 16h 06 min for a complete execution of the knn and the calculation of the other textual distances. It beats by far all previous approaches. We computed six other textual distances (discussed in section 2.1) to prepare further machine learning studies. Our model uses eight major functions. They range from data processing to results reporting.

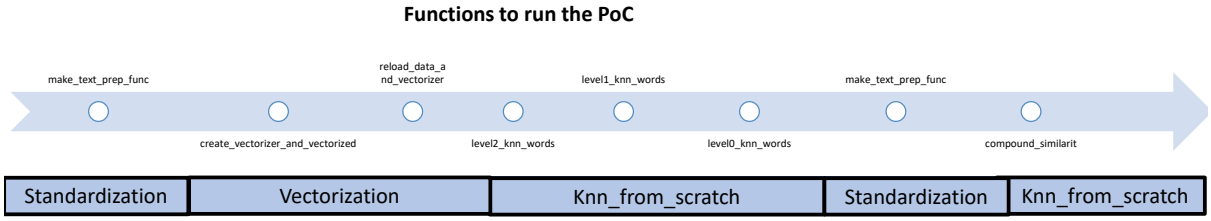


Figure 15 – Main functions - Final solution

The whole process runs on four notebooks. We have summarized in the following figure 16 The first step is the data processing and formatting. The second step is done by the notebook 02_standardization.ipynb which does all the textual preprocessing. Then we vectorize our baseline (discogs data) to use it as a train set. In the last notebook, we use the cdandlp database and the vectorized data to establish our 20 nearest neighbors via our functions. At the end of this step we generate pkl files to store the partitions of the results that we combine in a single file **resultat_knn_all.csv**

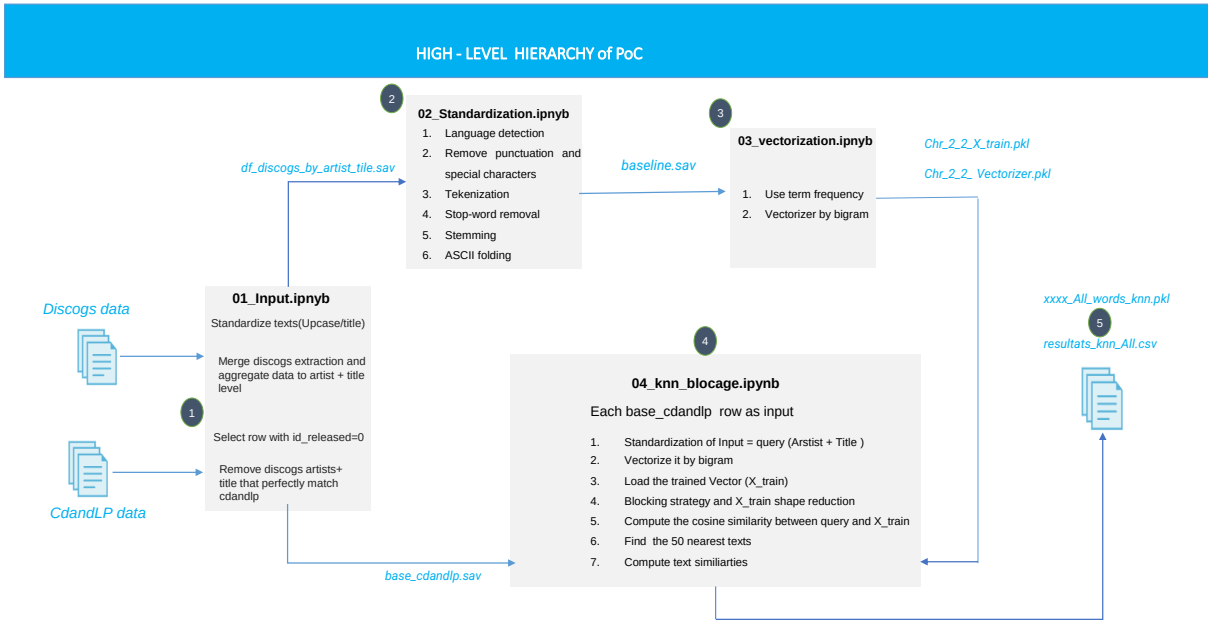


Figure 16 – Architecture - Final solution

4 Documentation du projet

Considering the complexity and the experiments carried out during this project, an appropriate documentation that allows a grip of the code used is necessary. From the docstrings of the functions, we used Sphinx as the main documentation tool of the project and deployed it as a web page.

4.1 Docstrings

```
def customize_output(i, results, df_baseline, sample) :

  """
   A function to make the output more appealing and informative.


    Arguments:
        i : index for row identification
        results : unsupervised knn api results (list of distances and indexes)
        df_baseline : discogs data
        sample  :  sample of cdandlp datasets, size = 1000

     Return :
            pd.DataFrame
    """
```

Figure 17 – Example of a function docstring

4.2 Sphinx Documentation

Sphinx is a tool that transforms the docstrings of your code into nice readable HTML files. Below is the documentation homepage.The first section is a reminder of the order of the functions and their hierarchy.

We can see hyperlinks that refer to categories of functions in the section. There are three categories. On of them is *Text Standardization*.

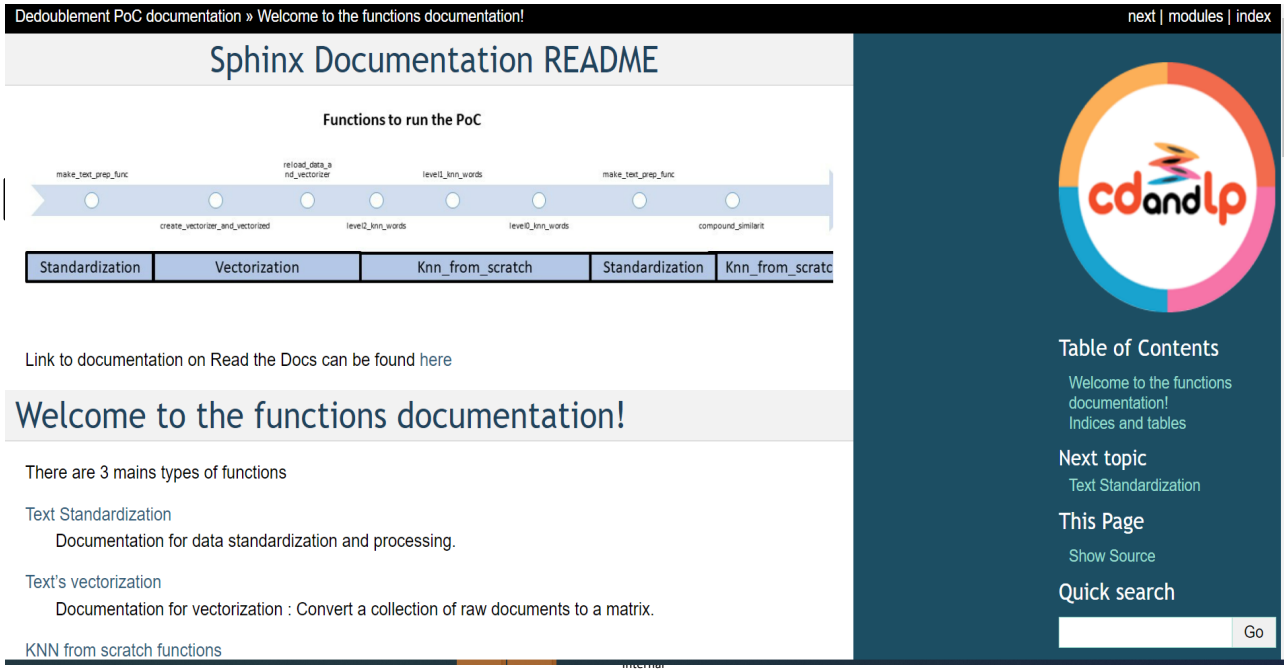


Figure 18 – Landing page - Sphinx Documentation

After clicking on one of the hypertext links **Text Standardization**, we get a page similar to figure 18. hyperlinks [**source**] on the far left of the description of each function refers to the source code of the function. Specialists for more details, can refer to it.

Text's vectorization ¶

Functions:

`create_vectorizer_and_vectorized_db`(...)	This function returns the vectorized reference base + the vectorize itself.
`reload_data_and_vectorizer`([analyzer, ...])	This function to import the pickles (vectorizer and the matrix Sparse of the documents).
`text_to_vect`(source_query, vectorizer)	A function to Standarize a string and vectorized it.

`vectorization.`**`create_vectorizer_and_vectorized_db`**`(`*`df_items, analyzer, ngram_range`*`)`

This function returns the vectorized reference base + the vectorize itself. [source]

Parameters:
- **df_items** (*DataFrame*) – Baseline Discogs.
- **analyzer** (*str*) – {'word', 'char', 'char_wb'}
- **ngram_range** (*tuple*) – Range of n-values for different n-grams to be extracted

Returns: Save/write to pickle the method of vectorization and the X_train

Return type: (matrix, vectorizer)

© Copyright 2022, Erwan BARGAIN, Salif SAWADOGO. Created using Sphinx 4.2.0.

Figure 19 – Example of a function description

4.3 Livrables

We provided technical documentation and algorithms coded in Python. To allow CDandLP to leverage our experiences and solution, we provide the deliverable as a folder for later use.

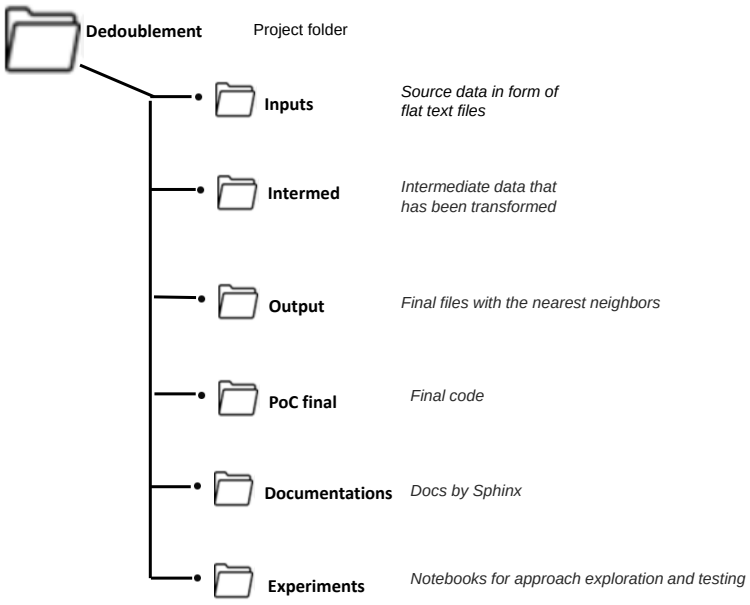


Figure 20 – Organization of livrables

It contains the Python Notebooks Scipts for the final solution and an expiremitentations section. To ensure reproducibility, we add requirement files. The input folders contain raw data provided by company, intermed contains the intermediate tables and files.

Conclusion

At the end of our study, we managed to propose a recommendation solution composed of the artist + the title. For a given article, we manage to provide the thirty most similar articles in descending order. Several tests have been performed to improve the execution time. This includes the use of machine learning algorithms, parallel computing tricks and optimization of distance measurements.

This project allowed us to deepen our knowledge in natural language processing or text mining. The limitation of computer resources also pushed me to refine my programming method by privileging faster and less expensive approaches in RAM. Moreover, I learned to organize myself well because I had to present my results regularly to my tutor. Not to mention that the literature review in English and the writing of the present in English pushed me to make my English level more professional.

In order to make it easier for the other collaborators of the company to take charge of this project. An important place was given to the documentation of the project. This project will allow CDandLP to carry out other very important use cases that will allow it to increase its sales and marketing performance.

List of Figures

List of Tables

References

[1] William E. Winkler. « The State of Record Linkage and Current Research Problems ». In: *U. S. Bureau of the Census* (1999).

[2] Martin Porter. *The Porter Stemming Algorithm*. [Page consulted on 16-April-2022]. 2006. URL: https://tartarus.org/martin/PorterStemmer/.

[3] F. Pedregosa et al. « Scikit-learn: Machine Learning in Python ». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Jon Louis Bentley. « Multidimensional binary search trees used for associative ». In: *Communications of ACM* (1975).