Textual representations and semantic embeddings: an application for sentiment/essay analysis

MARIE Clément, SAMAHA Elio, XIA Tianxiang

August 2, 2023

Contents

1 Introduction - Motivation

We were given several datasets from the website *https://www.trictrac.net/*, which sells a variety of board games. This website collects data for every article, such as reviews and grades posted by the users. Our task was to treat and analyze this data to gain valuable insights from it. One purpose was to classify these reviews and predict the sentiment (positive or negative).

This is a real-world application as the company could automatically gain insight on products by correlating words with grades. Utilizing word vector representation and dimensionality reduction helps establish meaningful associations between words and the sentiment expressed in reviews. By exploring the represented space, the company could identify trends in customer preferences, potentially increasing sales.

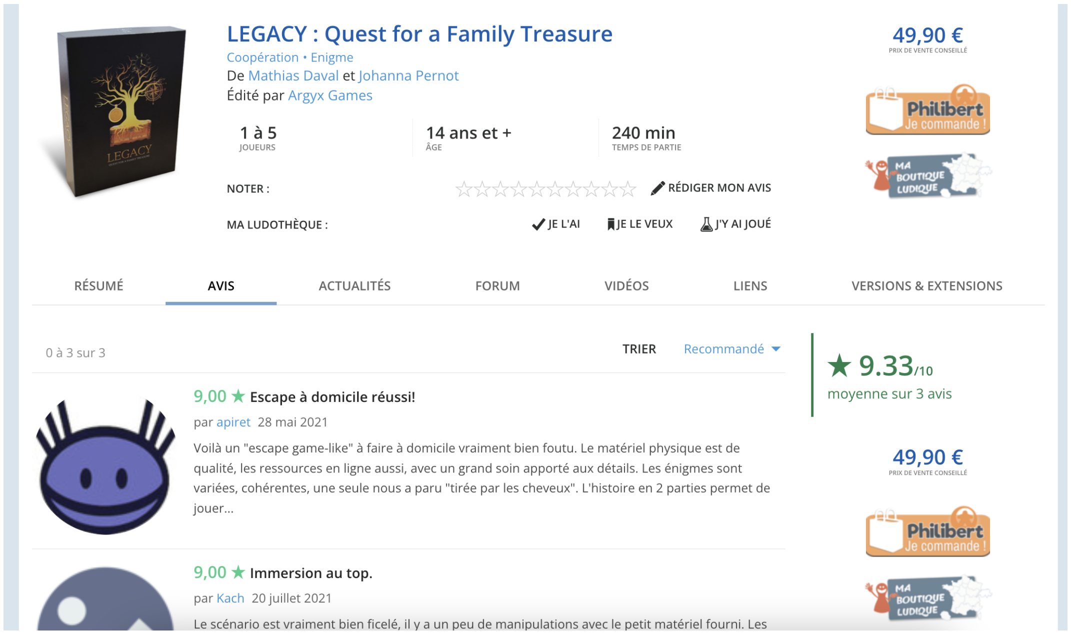

LEGACY : Quest for a Family Treasure
Coopération • Enigme
De Mathias Daval et Johanna Pernot
Édité par Argyx Games
1 à 5 JOUEURS | 14 ans et + ÂGE | 240 min TEMPS DE PARTIE
NOTER : RÉDIGER MON AVIS
MA LUDOTHÈQUE : JE L'AI | JE LE VEUX | J'Y AI JOUÉ
49,90 € PRIX DE VENTE CONSEILLÉ
RÉSUMÉ | AVIS | ACTUALITÉS | FORUM | VIDÉOS | LIENS | VERSIONS & EXTENSIONS
0 à 3 sur 3 TRIER Recommandé
★ 9.33/10 moyenne sur 3 avis
9,00 ★ Escape à domicile réussi!
par apiret 28 mai 2021
Voilà un "escape game-like" à faire à domicile vraiment bien foutu. Le matériel physique est de qualité, les ressources en ligne aussi, avec un grand soin apporté aux détails. Les énigmes sont variées, cohérentes, une seule nous a paru "tirée par les cheveux". L'histoire en 2 parties permet de jouer...
9,00 ★ Immersion au top.
par Kach 20 juillet 2021
Le scénario est vraiment bien ficelé, il y a un peu de manipulations avec le petit matériel fourni. Les

Figure 1: Tric Trac webpage : on the left of a star we have a grade, and below is the comment

The features X represents the *comment* column, and the target variable y is *grade*. In this report, we aim to uncover valuable insights from these reviews and effectively capture and quantify the sentiments conveyed. We will discuss our methodology, data processing approaches, and the potential of our findings.

2 Descriptive statistics of the *grades* feature

We are starting by looking at the grades data since it will be the criterion to classify reviews as positive or negative.

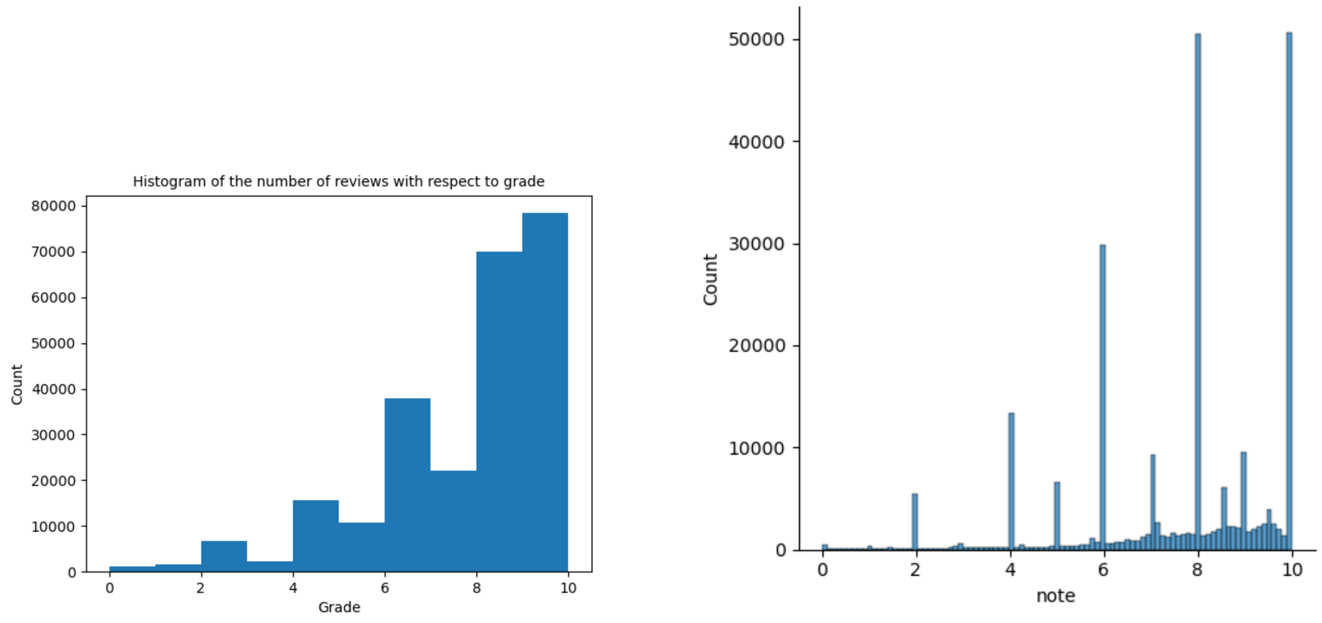


Figure 2: Grade distribution

On the histogram on the right plot, we can notice that the majority of the grades are actually integers. Removing float numbers (or rounding to closest integer) could possibly simplify the model. On the left plot, we can notice that the majority of reviews are actually positive.

It is cautious to already say it could introduce some challenges. Indeed, since grades are quite imbalanced, the model can become biased towards positive reviews. The model may tend to predict positive reviews more frequently, leading to lower accuracy for negative ones and an overall imbalance in predictions. Also, since there are fewer samples from negative reviews, the model may not have enough representative examples to learn the patterns and characteristics of the negative class.

3 Text documents to vectors

Text documents, in their raw form, consist of unstructured data that cannot be directly processed by machine learning algorithms. To effectively analyze and derive insights from text, it is necessary to transform the textual information into a numerical representation known as vectors because they provide a structured and quantifiable representation of textual information.

Therefore, since the features of the dataset represent the *comment* feature, we will tackle ways to treat the data.

3.1 Preprocessing

Raw text data might contain unwanted or unimportant text due to which our results might not give efficient accuracy and might make it hard to understand and analyze. So, proper pre-processing must be done on raw data.

We had to remove punctuation, URLs, numbers, accents, and stopwords. Stopwords are commonly used words in a language that are considered to have little or no significance in determining the meaning of a text. ‘not’ is not a stopword, because it might indicate the opposite sentiment.

We also applied Stemming which reduces words to their root forms. It minimizes the confusion around words that have similar meanings and lowers the complexity of the input space. However, it comes at the cost of throwing information away.

For example:

```
J'avais peur que les extensions s'éssouflent à terme...
->
['peur', 'extens', 'éssouflent', 'term', ...]
```

3.2 One Hot Encoding

One Hot Encoding is a vector representation of a word in the vocabulary, i.e. the unique list of all the words appearing in the documents. If the vocabulary size is n, each word in the vocabulary is therefore represented as a vector of size n. It takes binary values: 1 for the corresponding word and 0 otherwise.

On the implementation: We can represent the set of sentences as a tensor of shape (a, b, c) ie a matrices (number of sentences) of shape (b,c) where b represents the number of words in the sentence and c the vocabulary size.

Advantages:

- Intuitive and easy to implement

Inconvenients:

- Increase in dimensionality: a large vocabulary size implies a large number of columns, taking more memory size, which results in an increase in computational cost. + The matrix is sparse.
- Every vector sits in the orthogonal vector space so vectors are perpendicular to each other and are considered independent to each other, which is rarely the case.
- High chance of multi-collinearity due to dummy variables, which might affect the performance of the model (cf. Dummy Variable Trap)

3.3 Bag of Words

The Bag of Words (BoW) technique is a commonly used method for representing text data in natural language processing. It treats each document or sentence as a collection of words without considering the order or grammar, focusing only on the frequency of words. BoW represents text data as a sparse matrix, where each row corresponds to a document or sentence, and each column represents a unique word in the vocabulary.

Implementation Details In the BoW representation, the set of sentences is transformed into a matrix, where each row corresponds to a document or sentence, and each column represents a unique word in the vocabulary. The value in each cell of the matrix indicates the frequency or occurrence of the corresponding word in the document.

Advantages

- **Intuitive and easy to implement**: The BoW technique provides a straightforward representation of text data, where words are treated as independent features, and their frequencies capture some information about the documents.

Disadvantages

- **Increase in dimensionality**: One drawback of the BoW technique is the increase in dimensionality. As the vocabulary size grows, the number of columns in the matrix increases, occupying more memory space and resulting in higher computational costs. Additionally, the resulting matrix is sparse, containing mostly zeros.
- **Loss of word order and grammar**: By disregarding the order and grammar of words, the BoW technique may lose important linguistic information present in the text data.
- **Lack of semantic meaning**: BoW representation treats each word independently, ignoring the semantic relationships between words, which can limit its ability to capture the meaning and context of the text.

3.4 tf-idf

The BoW (Bag of Words) model assumes that the importance of a term is directly proportional to the number of its appearance in the document, this can easily be misleading when the most common words are ‘stopwords’. (But it really depends on the algorithm that we are going to use afterwards, if we do perceptron, it wouldn’t matter.)

But the BoW gives vectors with integer features, which may be favored by some algorithms, like Naive Bayes below. This being said, the tf-idf embedding gives vectors with features in $\mathbb{Q}$, which can be dilated to integers.

tf-idf (term frequency-inverse document frequency) first considers the whole of a corpus (for the terminology, we call the set of comments a corpus, each comment a document, every word a term), it assumes a term too frequent in the corpus has little information. Then it considers the importance of a term in a document as the frequency of the term in this document times a scalar representing its information in the corpus. The idf is actually an estimation of Shannon information of a term.

$$\begin{aligned}
\text{tfidf}(\text{term}) &= \text{tf}(\text{term}) \times \text{idf}(\text{term}) \\
\text{tf}(\text{term}) &= \frac{\#\text{ of times term appears in document}}{\#\text{ of terms in document}} \\
\text{idf}(\text{term}) &= \ln\left(\frac{\#\text{ of documents}}{\#\text{ of documents in corpus with term}}\right)
\end{aligned}$$

Other possible choices:

- A term can be many (successive) words (n-gram) (e.g. to take into account negations before a word)
- The exact formulae can be changed while the same idea remains

On the implementation `for` is slow in python, do use functions in numpy instead; we use sparse matrices as data structure to gain memory and boost speed sometimes.

Observations

- Preprocessing is important to reduce the dimension

4 Classification

4.1 Naive Bayes

The Naive Bayes algorithm does not require text representations such as one-hot encoding. It instead utilizes count vectorization (i.e. bag of words) to represent documents. Count vectorization preserves the word frequencies, allowing the algorithm to consider the importance of words in expressing sentiment.

Naive Bayes is a probabilistic classification algorithm. It is called naive because it assumes that each input variable is independent.

We are looking for:

$$\max_y \mathbb{P}(Y = y \mid X = (x_1, ..., x_n))$$

using the Naive Bayes formula: $\mathbb{P}(Y \mid X) = \frac{\mathbb{P}(X|Y)\cdot\mathbb{P}(Y)}{\mathbb{P}(X)} = \frac{\mathbb{P}(Y)\prod_i \mathbb{P}(X_i|Y)}{\mathbb{P}(X)}$ by independence, where $\mathbb{P}(X \mid Y)$ is the likelihood, $\mathbb{P}(X)$ is the evidence, $\mathbb{P}(Y \mid X)$ is the posterior and $\mathbb{P}(Y)$ is the prior.

More precisely, we used Multinomial Naive Bayes for our task because it is suitable for discrete and count-based features, such as word frequencies in text data. The count vectorization of words aligns well with the assumptions of this model. It allows the model to estimate the probability of sentiment classes based on the frequency distribution of words.

Advantages

- Even though the independence assumption is rarely true, the model is still effective
- Handles high dimensional data well

Inconvenients

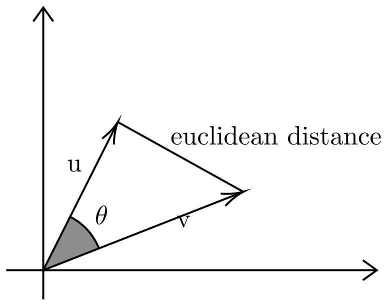
- Estimated probability is often inaccurate because of the naive assumption

4.2 Distances

Before introducing the k-nearest neighbors algorithm, it is important to note first that the distances to measure the similarities between documents we use for the algorithms are vital.

We can use Euclidean distance, or we can use the distance concerning only the angle between 2 vectors (so the 'difference of lengths' of documents is ignored).

$$\text{(cosine similarity)}S(u,v) := \cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|}$$
$$\text{(cosine distance)}D(u,v) := 1 - S(u,v)$$



We see that cosine distance is better in nlp : no curse of dimensionality (the Euclidean distance tends to be the same for every pair of vectors when the dimension grows).

4.3 K-Nearest neighbours

Algorithm One predicts the information associated with a vector by the majority of information associated with its neighbors (within k nearest). We use tf-idf embedding here. By simple experience and reasoning (the length doesn't affect the positivity) as well, we chose cosine distance.

For comments of grades 4 - 7, they are more or less neutral, we can't say if it is definitely positive or negative as human-being, so less chance for our more or less naïve algorithm. For this reason, we chose to do our algorithm on comments of extremities.

We add a parameter to our algorithm: **balance**. balance is a float ¿= 1 meaning that we trim the list of data of one label of more quantity to size = balance×(number of data with another label). This is to solve the problem of disproportional labeled data.

On the implementation

- We favor the nearer information when there is a tie.
- Cosine distance works much better than Euclidean distance.
- When the data is disproportionally labeled, we need to balance the data to ensure performance for the prediction of each label.
- We sometimes use implemented functions to improve efficacity after having understood the method and implemented it ourselves.

Positive

- Easy to implement

Negative

- Supervised
- $\Theta(n)$ for each prediction (n the size of training data), which is very slow when we want to use big training data to ensure better prediction.

5 Evaluation of models - Results

5.1 Model evaluation

First of all, they split the labeled dataset into training and testing subsets. The model is trained on the training data and then evaluated on the testing data to measure its performance, in order to assess its capacity to generalize predictions to unknown data.

Then, we used cross-validation for each model. It divides the data into parts, trains the model on some parts, and tests it on others. This process is repeated multiple times to get a reliable performance estimate. Cross-validation helps us understand how well the model works on unseen data and allows us to find the best settings for the model.

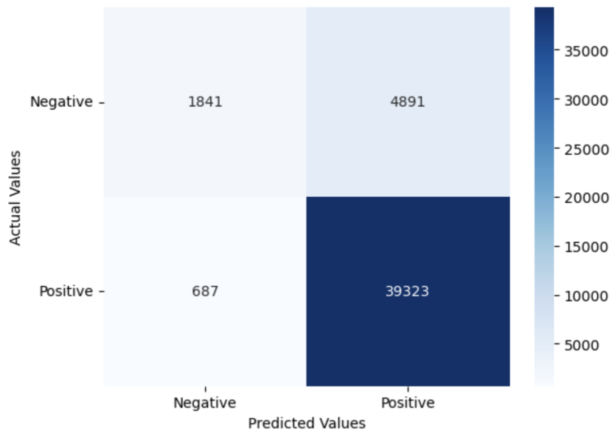
As mentioned earlier, the positive and negative classes are somehow imbalanced. Data stratification would be here inappropriate because the minority class will get underrepresented since it has fewer samples. Therefore, given the size of the dataset, an option would be to undersample the positive class: we train models with data that has the same proportion of positive and negative classes.

Finally, the metrics we used are standard classification metrics such as precision, recall, and F1 score. Precision measures the rate of true predictions. Recall measures the ability of the model to identify positive/negative comments correctly. F1-score is a metric that combines precision and recall scores (the harmonic mean of the two). All the evaluation is done on the test dataset.

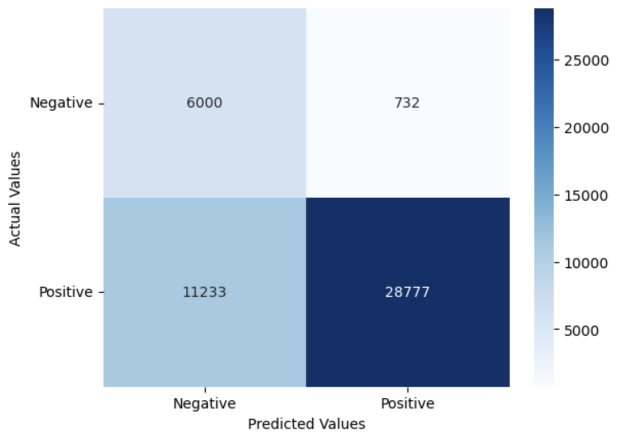
On implementation We enable to specify a random seed to ensure reproductivity.

5.2 Naive Bayes

Figure 3: Confusion matrices and associated metrics



(a) No undersampling



(b) Undersampling

	F1 Pos.	F1 Neg.	Accuracy
Undersampling	0.83	0.50	0.74
No undersampling	0.93	0.40	0.88

(c) Scores

We notice that without undersampling, the F1 score is higher for the positive class than for the negative class. The model is biased towards predicting positive sentiment, leading to a higher number of false positives. However, with undersampling, the F1-score is lower for the positive class but it more balanced with the negative class. As accuracy is not really relevant in the case of class imbalance, it is probably more appropriate to keep the second model with undersampling.

We are diving into the training phase with looking at learning curves. This is giving us an idea on how well the model generalizes to testing data.

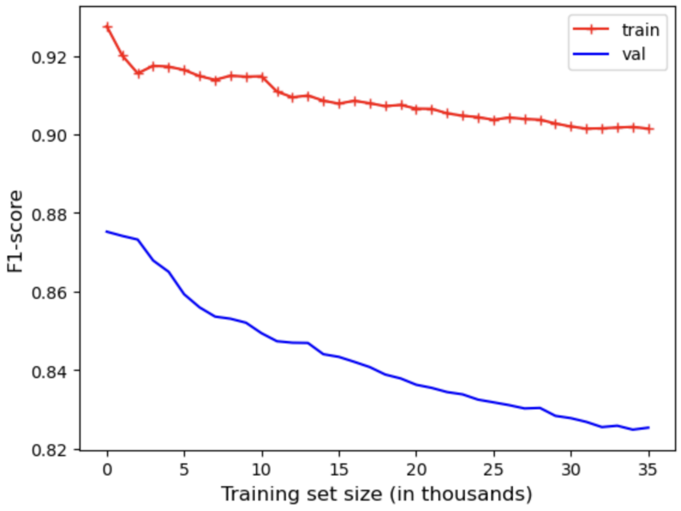


Figure 4: Learning curves for Multinomial Naive Bayes with undersampling (with training size in thousands)

We can see that the training F1-score is a bit larger than the testing F1-score, which was expected since the model training is done on the training set. We can see that the validation F1-score keeps on decreasing until the end of the training. The model is possibly overfitting. Possibly, training it on a bigger training set could improve performance.

5.3 K-Nearest neighbors

To begin with, we deliberately balanced the data, 50% of positive comments (5000) and 50% of negative comments (5000).

By adjusting the k, we obtained such result shown in Figure 5.

That was some good results. But in reality, our data is extremely disproportional, if we apply KNN directly, the algorithm would predict positive almost every time. (Although a positive comment has less chance to enter the neighborhood of a negative comment, there are too many of them.) Indeed, we tried and get 0.26 as accuracy for negative comments and 0.94 for positive comments.

The accuracy in total (on the data set) was good but it was not what we are after. We would like $(r_p + r_n)/2$ to be big (so in the previous case it was 0.6). $r_{p/n}$ is the accuracy on the positive/negative comments.

To solve this problem, we use our **balance** parameter introduced before. (recall that balance=1 means total balance, the bigger the less balance)

As the Figure 6 shows, the balance value would better stay at 1 (otherwise it damages significantly the accuracy on the negatives) and we increase k to remedy the disadvantage of positive prediction.

Where does this disadvantage come from, while the number of positives and negatives are equal in the training data (after trimming)? A conjecture is that the number of positive comments in the training data is significantly less after trimming so some vocabularies are not even present. In this case, we can only judge by more basic words and rely on a bigger survey, thus increasing k helps.

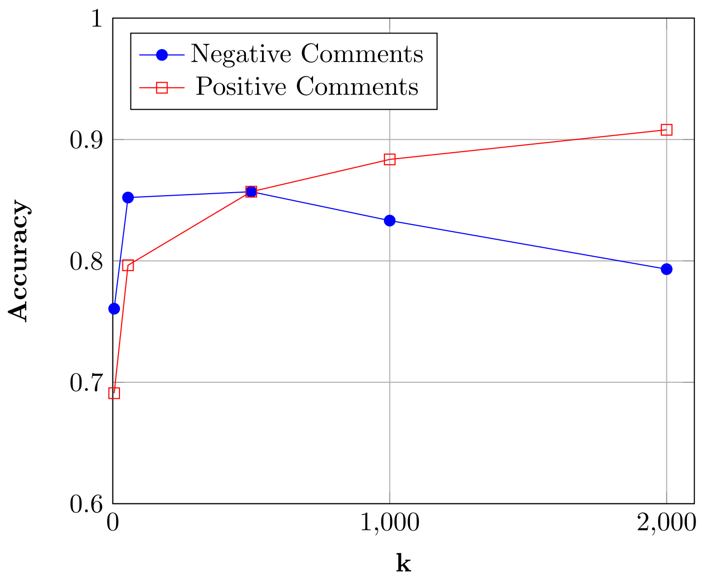
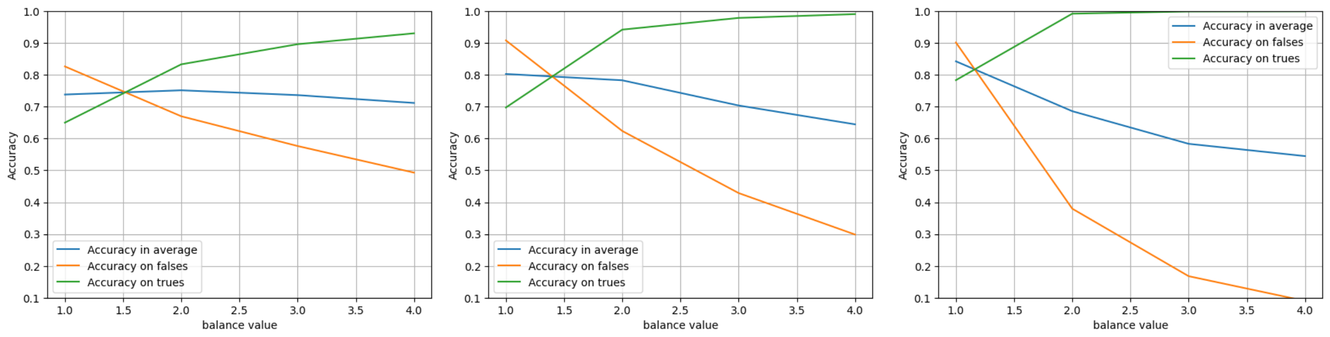


Figure 5: k nearest neighbor on balanced data



(a) k=5 (b) k=50 (c) k=500

Figure 6: Positive comments are labeled as true and negative comments are labeled as false

Now we show a case in reality, our assumption/scenario is that :

The given data on the games is disproportional. Most comments are positive. So positive comments do not give much information. The manager then wants to detect then inspect all the negative comments among all the comments.

Our KNN (with parameters balance=1, k=500 as explained in 4.3) does this job. So as shown in Table 1, we almost covered all negative comments, the payoff is that the precision of class false is low, meaning that the manager needs to see as much as 5 comments to read a true negative comment.

Class	Precision	Recall	F1-Score	Support
False (negative comments)	0.23	0.91	0.37	1789
True (positive comments)	0.99	0.76	0.86	22520
Accuracy = 0.77				

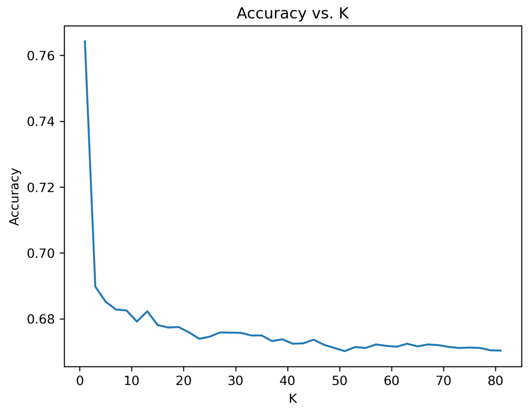
Table 1: Classification report with undersampling; note that we have a smaller support here for KNN than naive bayes, this is because of the efficacity problem.

5.4 KNN with other representations and distances

In our KNN above, we used tf-idf representation and cosine distance for reasons already mentioned. Here we make some experimental justifications by trying out other different options.

	BoW	**tf-idf**	**One-hot**
Euclidean	0.7642 (1)	0.7425 (113)	0.6864 (5)
Cosine	0.7725 (1)	0.7771 (1)	0.6864 (5)

(a) The table consists of three columns representing the word representation methods: One Hot Encoding, Bag of Words (BoW), and tf-idf. The rows represent the distance measurement techniques: Euclidean distance and cosine distance. The table showcases the accuracy values achieved by applying KNN (k=5) with each word representation and distance combination. It can be observed that KNN with tf-idf and cosine distance yields the highest accuracy of 0.7104, followed closely by KNN with BoW and cosine distance. On the other hand, KNN with tf-idf and Euclidean distance exhibits the lowest accuracy of 0.484.



(b) Variation of accuracy with k (KNN, euclidean distance, BoW)

Figure 7: KNN with other options; here we chose to only classify a comment as positive if the game received a rating strictly higher than 7.

Indeed that we see tf-idf combined with cosine distance works the best. Interestingly, as shown on the right in Figure 7, the result is the best when $k = 1$. This is reasonable as it can be interpretated as : euclidean distance only have meaning when the distance is very small. (Every pair of points tend to have the same distance when the dimension increases, curse of dimensionality.)

6 Another type of representation, persistent homology

The problems with the methods above are: 1. Not considering the order/structure of sentences in a document; 2. having parameter (e.g. k in KNN, estimated distribution in Naive Bayes) to determine which can not be optimal uniformly for every prediction. For the sentiment prediction, it is fine. So we change our task where we actually need to solve these problems.

In this section, we introduce a method for analyzing a text without parameters. In fact, we consider all the thresholds ranging from 0 to infinity. Moreover, we want to be able to find a meaningful correspondence between the computation and the structure of the text. We use persistent homology.

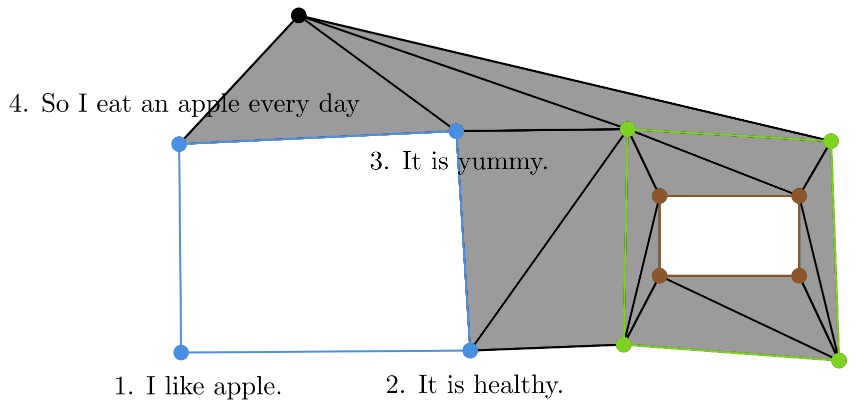
This is a project report instead of a course note, so the importance is not on the details of the algorithm but rather the ideas and experiments. But the algorithm and the theory in themselves are interesting and not easy to be found clearly on the internet, so for the clearness of the report we include concisely the necessary concepts and key algorithms there 8.

6.1 Essay grading

Inherently, persistent homology is useful when it comes to 3d modeling. But in data analysis in general, it is also useful (e.g. analyze the performance of a basketball team and get an intuitive view of the team structure).

Here we use the persistent homology to analyze discursive essays and grading (specifically the richness of their argument structures).

Roughly speaking, a better discursive essay should have a richer writing structure, (a proposition should be discussed in different angles, the last paragraph echoes to the beginning etc.). For example, in Figure 8, on the right, the essay has a statement, and then some arguments, finally a conclusion. On the left, each point represents a sentence. 1 - 2 - 3 - 4 are linked by time order, 1 - 4 are linked because they are similar (within a certain range of distance). This hole means a good argument. There are also filled triangles (2-simplices), those sentences form clusters, so they are filled instead of forming holes.



$\cdots$I like apple. It is healthy. It is yummy. So I eat an apple every day. $\cdots$

Figure 8: There are 2 holes in this simplicial complex, one represented by the blue cycle and one by brown. Note that the brown cycle and the green cycle are homology equivalent, so they count as one.

If we only have one such graph, we will have to determine the similarity threshold to link. However, different thresholds can give different insights. Structure changes with scale, e.g. Figure 9. So our approach will consider all the scales (whence "persistent" homology).

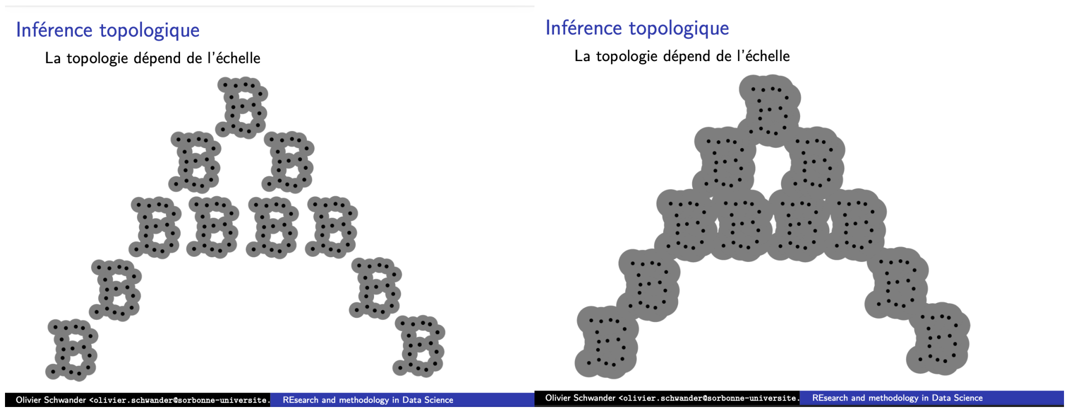

Inférence topologique
La topologie dépend de l'échelle
Olivier Schwander <olivier.schwander@sorbonne-universite.
REsearch and methodology in Data Science
Inférence topologique
La topologie dépend de l'échelle
Olivier Schwander <olivier.schwander@sorbonne-universite.
REsearch and methodology in Data Science

Figure 9: Structure changes with scale

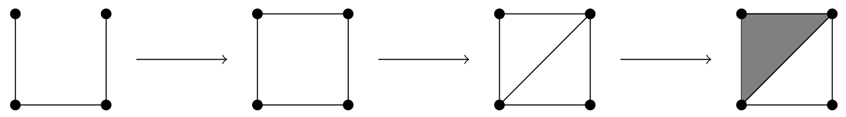


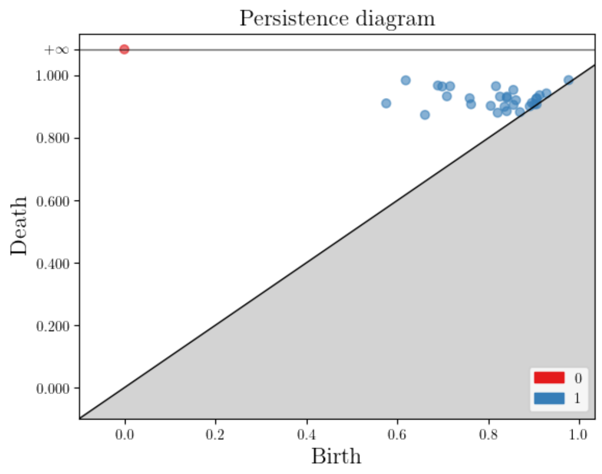
Figure 10: We continue linking and filling (e.g. triangles) when the d increases; note that we may have multiple linkings and fillings to do within a same d but we discretize these for the algorithm

The algorithm is the following (for the exact version 8.0.2) :

We vary d from 0 to 1, for every d we create a graph like Figure 8 (points representing sentences are linked first in order in the essay, then we link sentences whose differences are lower than d, we fill triangles, tetrahedrons, etc. as shown in Figure 10) What we will get is $(V_d)_{0 \leq d \leq 1}$, a filtration of graphes one included in another while d increases. Then there are holes that appear and disappear, we count the number of total appearances of holes.

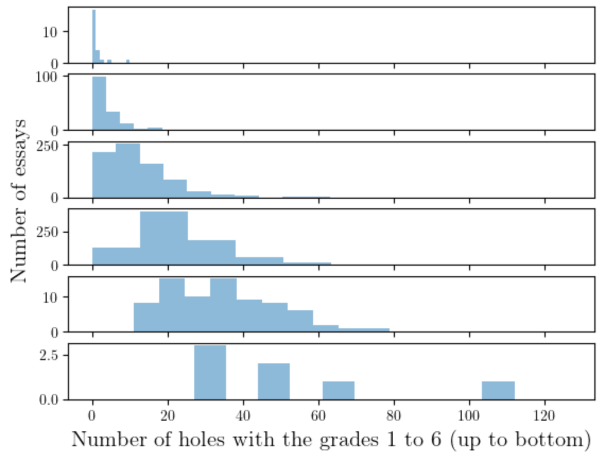
We hope to show that this number of holes reflects the quality of arguments of an essay.

Experiment on a real set of data data source (more precisely, the essay set 2, which is discoursive) : https://www.kaggle.com/datasets/thevirusx3/automated-essay-scoring-dataset/code?select=training_set_rel3.tsv



In @DATE1's world, there are many things found offensive. Everyone has their own opinion on what is offensive and what is not. Many parents are becoming upset because they think their children are viewing things that they should not. Other people are upset because they think the libraries are offending their culture or way of life. This is even taken to the extreme where people want censhorship on libraries to avoid this, which is wrong. Some people are becoming concerned about the materials in libraries...(∼450 words)

Figure 11: The persistence diagram of an essay of grade 4/6 : A blue point (x, y) in this diagram means that there is a hole appears at $d = x$ and disappears at $d = y$. One red point means there is only one connected component the whole time, because we link all the sentences by order in the essay.



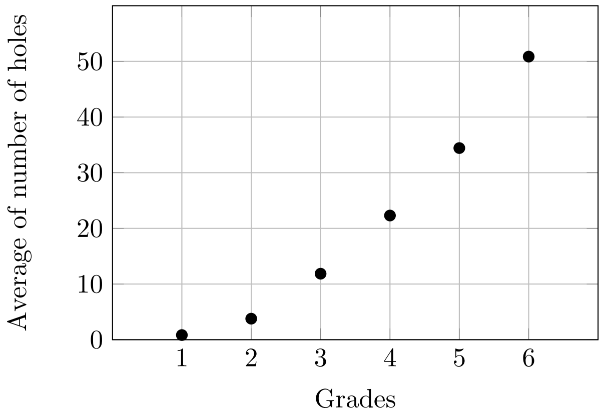


Figure 12: Number of holes increases as grade increases

Those are some discursive essays on the topic of offense/censorship graded 1-6. We first inspect an essay with a grade 4/6 as an example Figure 11. Then we show the average number of holes for each grade (1-6) as well as the average number of holes for each grade (Figure 12). We find that number of holes increases as the grade increases.

But there are other factors other than the richness of structure of the essay that may affect the number of holes, like the number of sentences (if we have more points in the set we will have more chances to form holes) or the number of words.

h

Number of Sentences 30-35	
Grade	Average of Number of Bar-codes
3	30.75
4	32.0
5	33.06
6	28.33

Number of Sentences 40-45	
Grade	Average of Number of Bar-codes
3	40.0
4	45.96
5	49.45
6	52.0

Number of Sentences 50-55	
Grade	Average of Number of Bar-codes
3	63.0
4	50.0
5	79.0

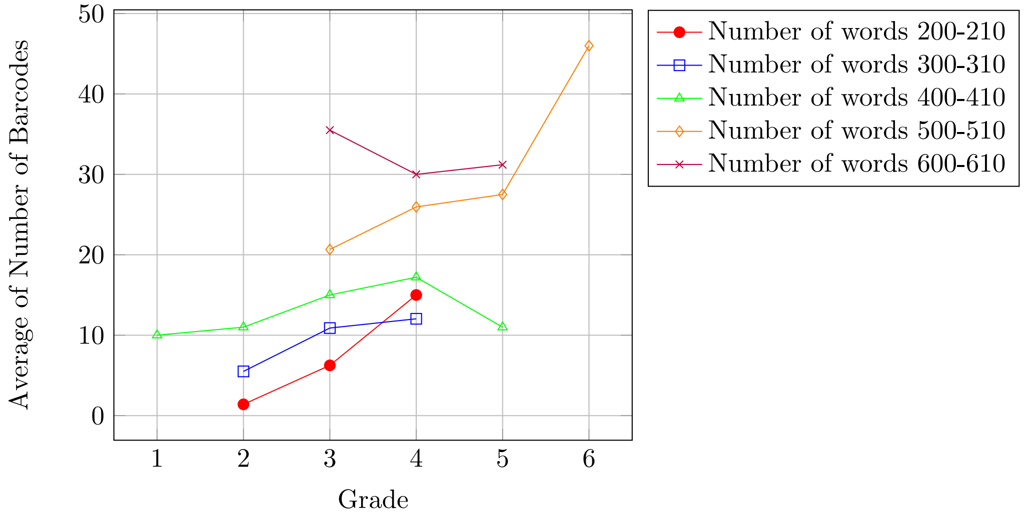


Figure 14: Grades and averages of number of holes within fixed sentence/word number ranges

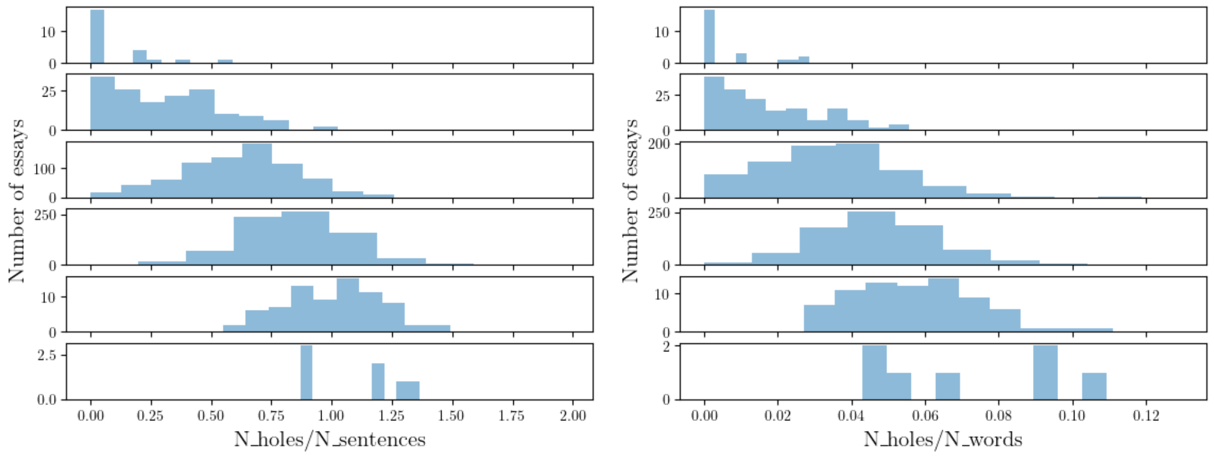


Figure 13: The average of holes each sentence/word increases as grade increases (from 1 to 6, upper to lower histogram)

First, let's try to eliminate the effect of number of sentences/number of words by dividing it (Figure 13). So as we expected, it still shows a simultaneous increase in the number of holes and grades. But even now we may suspect that maybe the number of holes is just the number of sentences squared. To show that this is not the case, we fix the number of sentences/the number of words in a range and inspect the number of holes.

Seeing Figure 14, we indeed observe and confirm the correlation of increase that we conjectured. However, the correlation is not absolute as we can constate. This is because we don't have enough data (for almost all contradictory data we have selected only one essay of that grade in that range) and other factors than the argument do affect the grade too (holes only show the structure of an essay, for which we chose to do the tests on discursive essays). Typically, the wording of essays of grade 3 but with 30-40 holes is bad and repetitive, which is probably the reason why they are graded not high.

In conclusion, while in Figure 13 we have enough essays as examples with the default that the method(division) is not rigorously convincing, and in Figure 14 we don't have enough essays for each range though the method is solid, with them combined, we are confident that the number of holes of dim 1 is a good measure of the quality of argument structure of an essay. While we changed our task for applying this topological representation of text, this task can not be achieved by KNN or Naive Bayes either, because they would both totally ignore the structure of an essay.

Further We may provision studying the identification of the type of arguments (or in the previous case, type of comments) by exploiting persistent homology structure, trying higher dimensional persistent homology, etc. But our journey ends here.

7 Insights - Analysis

7.1 Challenges encountered

The most challenging task was to process data since the dataset is very large. We realized how efficient libraries are since their computation time is low compared to when we implement it all by hand. The implementations are optimized, and the computations are designed for performance. Moreover, libraries such as the ones from sklearn take advantage of parallel computing (executed on multi-core processors).

Another challenge encountered is that it was not obvious at first how to apply preprocessing. Indeed, we had to think about how much we wanted to remove stopwords or common words. Removing these words might simplify the model, but we might also lose information with it. Another example is the use of negative contractions. It was not obvious if they could have a significant impact or not.

A common obstacle with NLP is that sentences are far from perfect, even after preprocessing is applied. Sentiment classifiers can also struggle when encountering words or phrases that were not present in the training data, so that's why using word embeddings are useful to generalize to unknown terms.

7.2 Lessons learned

The importance of preprocessing is the first lesson we could remember from this project. Indeed, without preprocessing, making a mathematical representation of words does not make any sense since the word embedding space will have a very large dimension, thus making predictions almost impossible.

Then, handling class imbalance. It is easy to classify reviews as positive when the majority of reviews in the training data are positive. Undersampling was a good solution to this problem (with the tradeoff that we lose some precision in prediction).

7.3 Limitations and areas of improvement

Sentiments are subjective, and annotating large datasets with sentiment labels can be challenging due to varying interpretations. Consequently, different customers might assign different sentiment scores, leading to inconsistencies in the training and evaluation data. Additionally, sentiment analysis can be dependent on the context, making it challenging to capture the nuances and sarcasm present in the text.

Another challenge is the domain adaptation problem because our model is trained on one domain and may struggle to generalize well to other domains due to differences in language usage and sentiment expressions.

We could evaluate the ability of our model to generalize to reviews from other fields (such as movie reviews) and see if it can transfer its knowledge pretty well or not.

Finally, the models we used to represent text data were not sequential for the most part, instead, they focused on the words themselves. Using persistent homology is a way to analyze the structure in a human-force way. Another option would be to use Deep Learning models such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, or transformer-based architectures (e.g., BERT, GPT). These models can capture complex linguistic patterns and contextual information.

7.4 Outro

Throughout the project, we learned how text representation is important for computers to process natural language and different representations adapt to different tasks. Moreover, we improved our ability to write a good report and present a project. All thanks to Professor Olivier Schwander.

8 Annexe

This content is a synthesis of this course[1] and other information on the internet.

Motivation Given a point set, find the underlying (homological) structures of the data. First, let's consider a topological space, how do we determine the homology structure of the space in a computational way.

Definition 1 (p-simplex ($p \in \mathbb{N}$))**.** *Let V be some finite set (the vertices). A p-simplex σ is the convex hull of $p+1$ points $x_0, \ldots, x_p \in V$. We denote $\sigma = \mathrm{conv}\{x_0, \ldots, x_p\}$ a subset of V.*

RM In practice, we simplify a topological space to its homology equivalence (thin triangulation) without losing the information we want (connected components, holes).

Definition 2 (Face)**.** *A face of σ is $\mathrm{conv}(S)$ where $S \subset \{x_0, \ldots, x_p\}$*

Definition 3 (Simplicial complex)**.** *A simplicial complex X is a finite collection of simplices s.t.*

$$\forall \sigma \in X, \forall \tau \subset \sigma, \tau \in X$$

Definition 4 (Chain space)**.** *The vectorial space of k-chains of a simplicial complex X over a field $\mathbb{K}$ is defined by:*

$$C_k(X) := \left\{ \sum_{i=1}^{|X_k|} \alpha_i \sigma_i : \alpha_i \in \mathbb{K}, \sigma_i \in X_k \right\} \simeq \mathbb{K}^{|X_k|}$$

where $X_k = \{A \subset X : |A| = k+1\}$ the set of k-simplexes in X.

RM Let's work with $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$ so that we don't consider orientation.

Definition 5 (p-chain)**.** *A p-chain is a subset of p-simplices in a simplicial complex X.*

Definition 6 (Boundary operator)**.**

$$\begin{aligned} \partial_k : C_k(X) &\to C_{k-1}(X) \\ \sigma = \{v_0, \ldots, v_k\} &\mapsto \sum_{j=0}^{k} (-1)^j \underbrace{\{v_0, \ldots, v_k\} \setminus \{v_j\}}_{\in X_{k-1}} \\ (\lambda\sigma + \sigma') &\mapsto \lambda \partial_k \sigma + \partial_k \sigma' \textit{ (linear extension)} \end{aligned}$$

Lemma 1.

$$\forall k, \partial_k \partial_{k+1} = 0$$

Definition 7 (k-th homology group)**.**
- *k-cycles : $Z_k := \mathrm{Ker}(\partial_k)$*
- *k-boundaries : $B_k := \mathrm{Im}(\partial_{k+1})$*

The k-th homology group of X for the field $\mathbb{K}$ is the following quotient of vector spaces:

$$H_k(X) := \frac{Z_k}{B_k}$$

(well defined because of the Lemma 1) The k-th Betti number $\beta_k = \mathrm{rank}(H_k)$.

RM The computational method is by Gauss elimination, we omit it here because we later introduce a method to calculate barcodes which is more powerful and uses Gauss elimination too.

Now, we are able to give the homology structure of a given topological space (simplicial complex). Let's go back to the beginning, initially, we are only given a point set, we may link the points in the set to form a simplicial complex. The way in which we link the points change our view of the underlying structure. The scale matters Figure 9.

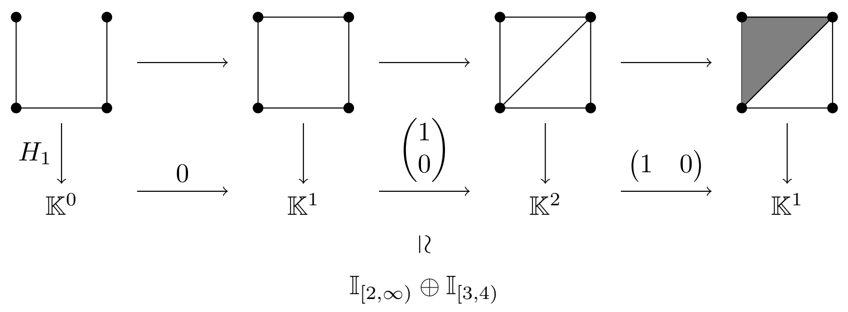


Figure 15: Filtration gives persistence module gives decomposition gives barcodes

Definition 8 (Filtration). *A filtration over T is a family $\mathcal{F} = (F_t)_{t\in T}$ of increasing topological spaces:*

$$\forall t, t' \in T, t \leqslant t' \Rightarrow F_t \subset F_{t'}$$

Definition 9 (Persistence module). *Let $\mathbb{K}$ be a given field. A persistence module over $T \subset \mathbb{R}$ is a family $\mathbb{V} = (V_t)_{t\in T}$ of $\mathbb{K}$-vector spaces endowed with linear application $v_t^{t'} : V_t \to V_{t'}$ induced by a filtration by being applied H_k (e.g. Figure 15).*

$\mathbb{I}_{[l,r)}$ *(the interval can be open/close at 2 ends) is an interval module with $T = [l, r)$, $(V_t)_t = (\mathbb{K})_t$, $(v_t^{t'}) = (\text{id})$.*

Theoreme 1 (Decomposition theorem). *A persistence module $\mathbb{V}$ can be decomposed as a direct sum of interval module in the following case (sufficient, not necessary):*

- *If T is finite. [Gabriel, 72]*
- *When all the vector spaces V_t are finite-dimensional. [Crowley-Boevey, 2012]*

Furthermore, when it exists, the decomposition is unique (up to isomorphism and ordering of terms). See an example at Figure 15

RM Intuitively, the way to decompose is beginning from where a cycle occurs and following the linear map until it diminishes and concluding an interval module; the multiset of pairs of extremities of the intervals obtained by decomposition is called the **barcodes** or the **persistence diagram**; to compute the decomposition, we do Gauss elimination (explained in the following).

8.0.1 Algorithm to calculate the persistence diagram

Input A simplicial filtration, that is a filtration over a simplicial complex K which verifies:

- $T = \{1, \ldots, m\}$ (finite, so we have a decomposable filtration).
- $K_1 = \{\sigma_1\}, K_m = K$
- $\forall t \in T, K_t$ is a simplicial complex, which is a sub-complex of K_{t+1}.
- We only add one simplex at each step, that is $K_{t+1} \backslash K_t = \{\sigma_{t+1}\}$.

Proof of validity. Each $c_i = 0$ means the birth of a cycle (one plus for the dimension of the kernel). Each $c_i \neq 0$ trivialize a cycle (one plus for the dimension of the image). It remains to show that the pairing is right.

Algorithm 1 Compute the barcodes corresponding to a simplicial filtration

```
M := the matrix of boundray operator $\partial =: (c_0, \dots, c_m)$
$\mathrm{low}(j) := \begin{cases} \max\{i \mid M_{ij} \neq 0\} \\ 0 \text{ if } M_{ij} = 0 \text{ for all } i \end{cases}$
for $j = 1 \dots m$ do
    while $\exists i < j$ s.t. $\mathrm{low}(i) = \mathrm{low}(j) \neq 0$ do
        $c_j \leftarrow c_j + c_i$ (Still working with $\mathbb{K} = \mathbb{Z}/2\mathbb{Z}$)
    end while
end for
Barcodes are $\{(i, \max\{j \mid \mathrm{low}(j) = i\}) \mid c_i = 0\}$ (1. This is a multiset 2. max return $\infty$ if empty 3. the set where we take max is of size at most 1)
```

Consider a $c_j \neq 0$, $c_j = e_{l_1} + \cdots + e_{l_k}$ $(0 \leq l_1 < \cdots < l_k < j)$. Note that $\partial_d(\sigma_{l_1} + \cdots + \sigma_{l_k}) = 0$ (Lemma 1). So $c_{l_k} = 0$. A posteriori, we can say that $(\sigma_{l_1} + \cdots + \sigma_{l_k})$ this cycle is created at l_k as in a base of $\mathrm{Ker}\, \partial_d$ (we construct such base a posteriori in this way).

So the pairing is right.

□

8.0.2 Similarity Filtration with Time Skeleton (SIFTS) [2]

As in KNN we choose an embedding and a distance. Then we construct a filtration of Rips Complexes.

Definition 10 (Rips complex)**.** *Given a symmetric matrix $M \in M_n(R^+)$, we define a filtration of Rips complexes*

$$(V_d := \{A \subset [\![1, n]\!] : \forall i, j \in A, M_{i,j} \leq d\})_{d \geq 0}$$

RM A simplex appears when the last appearance of the its faces happens.

Given an essay, we split it to sentences in order. We embed each sentence by tf-idf. Then we construct our matrix M by setting $M_{i,j} = 1 - \cos(\theta_{i,j})$ (the cosine distance) where $\theta_{i,j}$ is the angle between the vectors embedded by the ith and jth sentences. Then we set $M_{i,i+1} = 0$ (whence "time skeleton", the reason is to preserve the order of the essay). We obtain then a filtration of Rips complexes. In the filtration, what all this means is that, at index 0, we will have a skeleton from the first sentence to the last sentence, then we add simplexes with closer points then those with further points.

Then we decompose persistence module induced by the filtration in dim 1, the number of interval modules is what we called earlier the number of holes. Note that if we just count the number of holes at each index for the filtration, it would be trivial and not useful, we have to be able to identity the holes throughout the filtration so that we don't count a hole twice.

8.0.3 Stability theorem

Definition 11 (Homological critical value)**.** *Let X be a topological space and $f : X \to \mathbb{R}$ a function. A homological critical value of f is a real number b for which there exists an integer k such that $\forall \epsilon > 0$, $H_k(f^{-1}(-\infty, b - \epsilon]) \to H_k(f^{-1}(-\infty, b])$ is not an isomorphism.*

Definition 12 (Tame)**.** *A function $f : X \to \mathbb{R}$ is tame if it has a finite number of homological critical values and the homological groups $H_k(f^{-1}(-\infty, a])$ are finite-dimensional for all $k \in \mathbb{Z}$ and $a \in \mathbb{R}$.*

RM So that the first condition of the decomposition theorem is satisfied and we have a finite number of barcodes.

Theoreme 2 (Stability Theorem)**.** *For any two tame functions $f, g : X \to \mathbb{R}$*

$$d_\infty(\mathrm{Dg}\ f, \mathrm{Dg}\ g) \leq \|f - g\|_\infty$$

where

$$d_\infty(\mathrm{Dg}\ f, \mathrm{Dg}\ g) = \inf_{m : \mathrm{Dg}(f) \sqcup \Delta \tilde{\to} \mathrm{Dg}(g) \sqcup \Delta} \ \sup_{t \in \mathrm{Dg}(f) \sqcup \Delta} \|t - m(t)\|_\infty$$

and Dg *denotes the persistence diagram.*

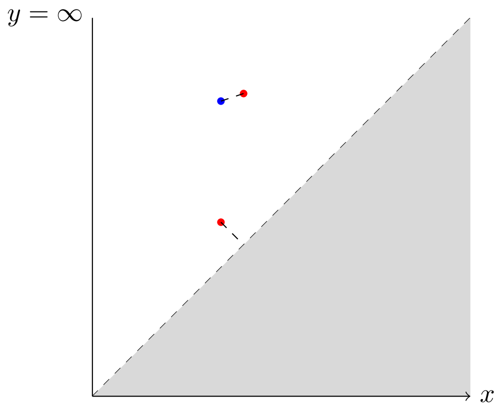


Figure 16: Barcodes matching on a persistence diagram

Proof. Let $f, g : X \to \mathbb{R}$ be 2 tame functions. We denote $\varepsilon = \|f - g\|_\infty$. We pose $F_t = f^{-1}((-\infty, t]), G_t = g^{-1}((-\infty, t])$ $(t \in \mathbb{R})$.

Key observation: $\{F_t\}_{t\in\mathbb{R}}$ and $\{G_t\}_{t\in\mathbb{R}}$ are ε-interleaved w.r.t. inclusion:

$$\forall t \in \mathbb{R}, G_{t-\varepsilon} \subseteq F_t \subseteq G_{t+\varepsilon}$$

We denote $F^{2\varepsilon} = \{F_{2n\varepsilon}\}_{n\in\mathbb{Z}}$, $G^{2\varepsilon} = \left\{G_{2(n+1)\varepsilon}\right\}_{n\in\mathbb{Z}}$ 2 discret filtrations.

$$\cdots \subseteq F_0 \subseteq G_\varepsilon \subseteq F_{2\varepsilon} \subseteq \cdots \subseteq G_{(2n-1)\varepsilon} \subseteq F_{2n\varepsilon} \subseteq G_{(2n+1)\varepsilon} \subseteq \cdots$$

We denote another discret filtration $\{H_{n\varepsilon}\}_{n\in\mathbb{Z}}$, where $H_{n\varepsilon} = \begin{cases} F_{n\varepsilon} \text{ if } n \text{ is even} \\ G_{n\varepsilon} \text{ if } n \text{ is odd} \end{cases}$

$$d_\infty(\mathrm{Dg}\ f, \mathrm{Dg}\ g) \leq d_\infty(\mathrm{Dg}\ f, \mathrm{Dg}\ F) + d_\infty(\mathrm{Dg}\ F, \mathrm{Dg}\ H) + d_\infty(\mathrm{Dg}\ H, \mathrm{Dg}\ G) + d_\infty(\mathrm{Dg}\ G, \mathrm{Dg}\ g) \leq (2+1+1+2)\varepsilon \leq 6\varepsilon$$

(We can optimise it to ε, but difficult)

□

References

[1] *INF556 – Topological Data Analysis.* Topological Data Analysis. URL: https://www.enseignement.polytechnique.fr/informatique/INF556/.

[2] Xiaojin Zhu. "Persistent homology: An introduction and a new text representation for natural language processing". In: 2013.