COMPUTATIONAL TOPOLOGY: TEXT ANALYSIS USING PERSISTENCE DIAGRAMS

ANDREJ DOLENC, ROK IVANŠEK, PETER US
NEŽA MRAMOR KOSTA, GREGOR JERŠE

1. INTRODUCTION

Persistence is a mathematical formalism for following how the shape changes (i.e. for following the homology groups and Betti numbers) as a complex grows or evolves. We can present this information in the form of persistence diagrams where for each homology class we have the time of their life (when it was formed) and death (when it disappears) given as a set of integers $[i, j]$ that are relative to some underlying filtration that follows the forming of some complex K. The visualization of a persistence diagram is then simply a scatter plot with i-s on the x-axis and j-s on the y-axis.

The main idea behind this project was to determine whether or not persistence diagrams built on top of documents from different domains differ significantly enough between the domains that they alone could be used to classify the texts. Because of ease of computation we mainly restricted ourselves to generators up to and including with second dimension. This restriction comes with a benefit of being somewhat intuitively explainable: generators of dimension 0 follow how the complex is interconnected at different points during its creation (counting number of connected components), and generators of dimension 1 how the holes in the complex are created and filled in.

2. DATA USED

2.1. **Description of data.** Our data consisted of text documents in English language from 3 domains: news articles about sports (source), abstracts from scientific papers (source), and movie reviews. Per each domain we had 20 documents, which we later split on train and test sets with 10 documents per set per domain. Each document contained at least 100 words in total, with sports articles averaging exactly 150 words per document, scientific abstracts 172.55 words per document, and movie reviews 381.6 words per document.

2.2. **Preparation of data.** Since we wanted to build topological complexes on this data, we had to somehow convert the textual documents into a set of points in some space of arbitrary dimensions. A common way of doing this involves first preprocessing the words in each text into their base form by applying word stemming and lemmatization algorithms. We also removed all the stop words from the documents and ignored whitespace in the resulting documents. After this process we are left with 60 documents (20 per domain), on which individually we then count various occurrences:

- average word length,
- average sentence length,
- shortest sentence length,
- total number of three most common words among all the words,
- number of words of length ≤ 3, and
- number of words of length ≥ 8.

Date: June 17, 2016.

This gives us 6 counts per each document, which we can normalize to the interval $[0, 1]$, and use as a point in 6-dimensional space representing the document.

We also used another popular method for text preprocessing called term frequency-inverse document frequency (tf-idf for short)[1] to generate a number of additional (more useful) features. The method tf-idf works as follows. For each selected word w we get one feature vector, meaning for each text t in the dataset we get one value, let's call it v_{wt}. This value is calculated by the formula $v_{wt} = tf(w,t)idf(w)$, where $tf(w,t)$ is the number of times a word w appears in text t and $idf(w) = \log \frac{N}{d_w}$, where N is the number of all texts in the dataset and d_w tells us in how many texts the word w appears in. Intuitively this means that words that appear in a lot of texts get weighed less (provide less information). Typically we take n most common words is the whole collection of texts, which gives us n features for each text (n columns in a feature matrix). While we played around with the number of features we got from tf-idf (parameter n), this gave us a bunch of additional values per each document.

3. METHODS

With each document being a single point in space, we can now construct topological complex on top of multiple points and check its persistence diagrams. In order to obtain some meaningful data out of it, we first (randomly) split 20 documents from each domain onto train and test sets with 10 documents per set. We then constructed a complex of dimension ≤ 2 for each of these 6 sets (3 domains, each containing train and test sets), resulting in 6 complexes. After this we used clustering (either single-linkage or max-linkage) on persistence diagrams of these complexes using bottleneck distance as the distance metric. If the persistence diagrams did in fact differ between domains, we would optimally obtain 3 clusters where only persistence diagrams of train and test sets from same domain would be connected.

Our original intention was to use alpha shapes as our complexes, however since we were unable to find an implementation of alpha shapes for points in more than 3 dimensions, we had to resort to first reducing the dimensions of our datasets. We achieved this using principal component analysis (*PCA*), which reduced number of columns in each train and test set down to just 3, enabling us to use 3 dimensional alpha shapes without any modifications. Since we feared that using PCA would negatively affect our result, we also tried building other complexes on whole dataset; i.e. we constructed Vietoris-Rips complex and Čech complex instead of alpha shapes. Since our favorite library for computational topology Dionysus doesn't come with a nice python wrapper for Čech complex, we had to write that ourselves.

We also considered whether checking persistence at only select few time stops during complex's creation would in some way give better results than just checking it for the whole timespan. To do this, we computed the maximum distance R between two points in each domain separately, split the interval $[0, R]$ to 10 pieces and then only checked complexes for those 10 resulting parameters.

4. RESULTS

To visualize persistences of homology groups we used the bar code diagrams and persistence diagrams. The bar code diagram for a certain homology group H is a two dimensional plot that shows us the life spans of all the homology classes. For each homology class γ we plot a line segment that starts in point (i_γ, y_γ) and ends in point (j_γ, y_γ), where i_γ and j_γ are life and death of γ and y_γ is a unique number that represents γ on the y axis of the plot.

[1] You can read more about tf-idf at `https://en.wikipedia.org/wiki/Tf-idf`.

Persistence diagrams show the same information in the form of a scatter plot. For each γ, $pers(\gamma) = [i_\gamma, j_\gamma]$ in some homology group we just plot the point (i_γ, j_γ).

Here are the barcode diagrams and persistence diagrams obtained for each of the three domains abstracts, sports and reviews. To generate this specific plots we used all the feature functions described in 2.2 plus the tf-idf method for the first 50 most common words. We also used the maximal filtration, meaning filtration (in particular parameter r) was not split to some predetermined number of intervals but just left as is and in this case we were looking at the formations of Čech complexes.

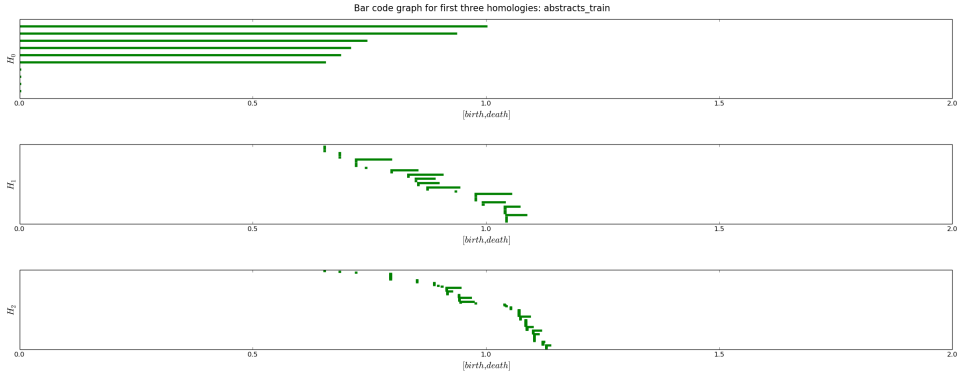

FIGURE 4.1. Bar codes for abstracts_train

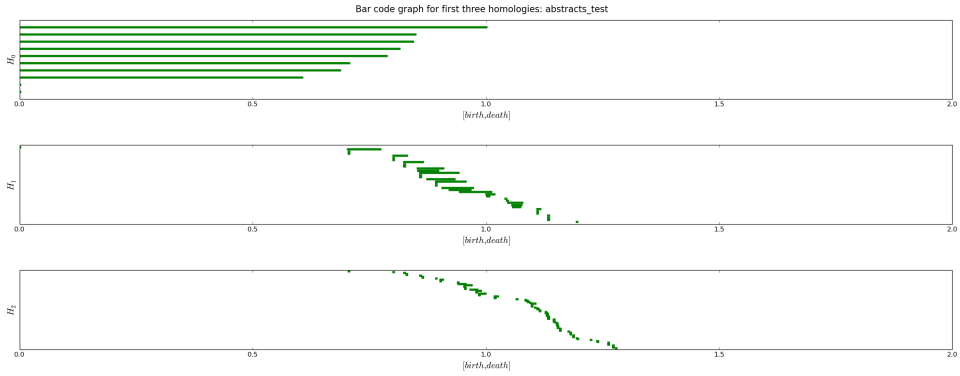

FIGURE 4.2. Bar codes for abstracts_test

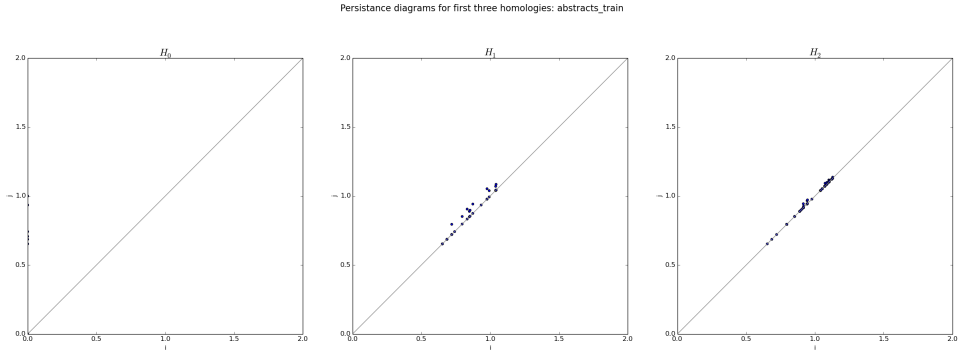

FIGURE 4.3. Persistence diagrams for abstracts_train

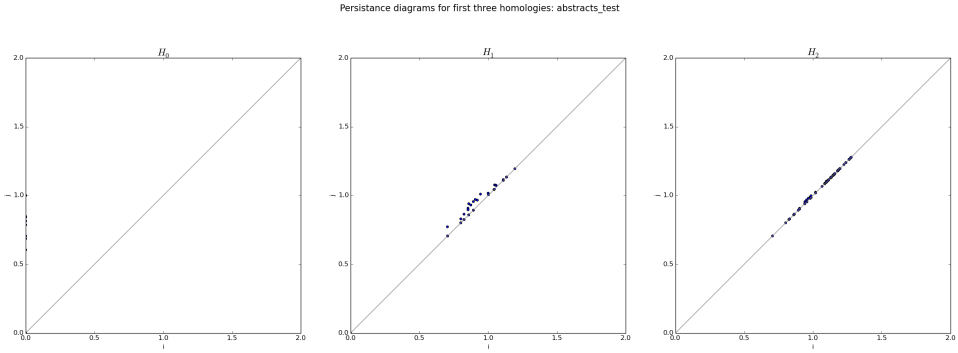

FIGURE 4.4. Persistence diagrams for abstracts_test

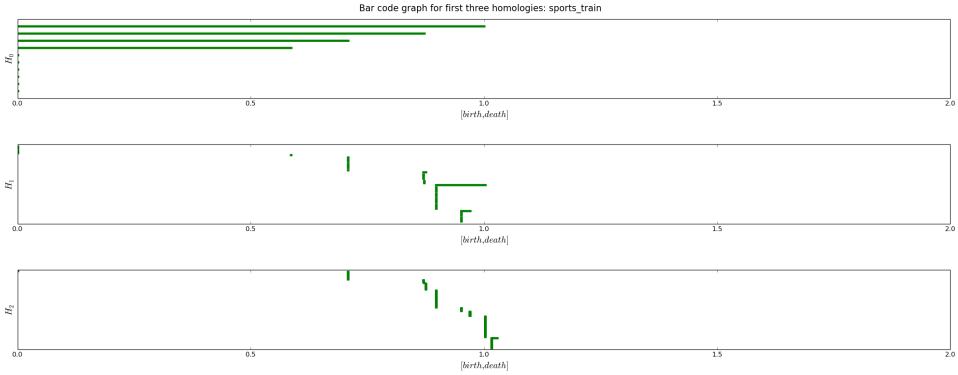

FIGURE 4.5. Bar codes for sports_train

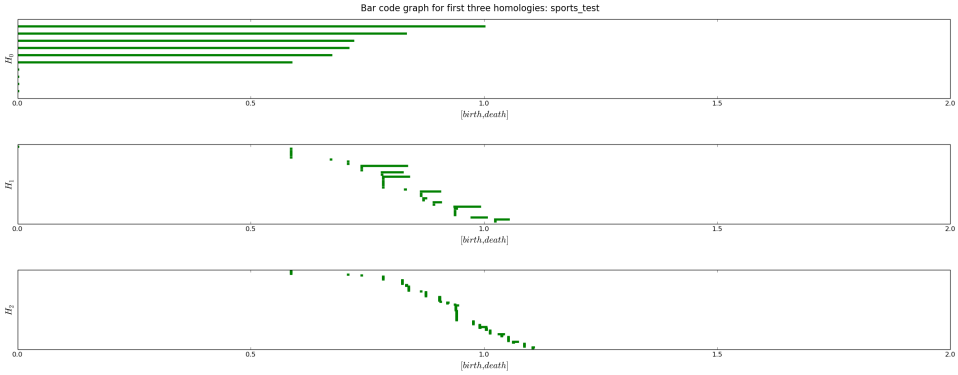

FIGURE 4.6. Bar codes for sports_test

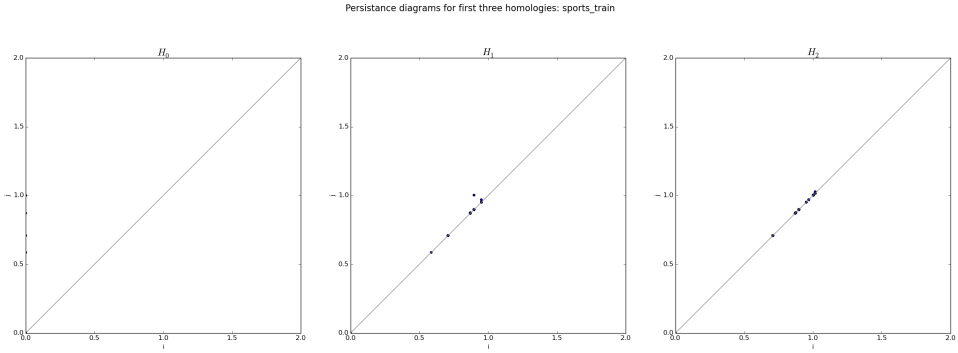

FIGURE 4.7. Persistence diagrams for sports_train

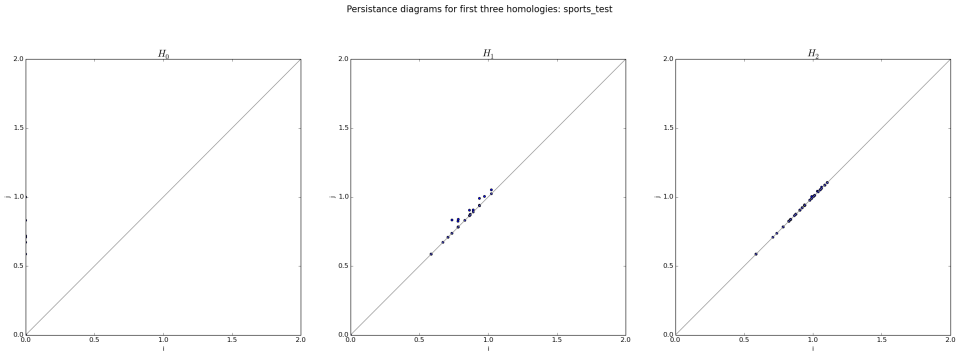

FIGURE 4.8. Persistence diagrams for sports_test

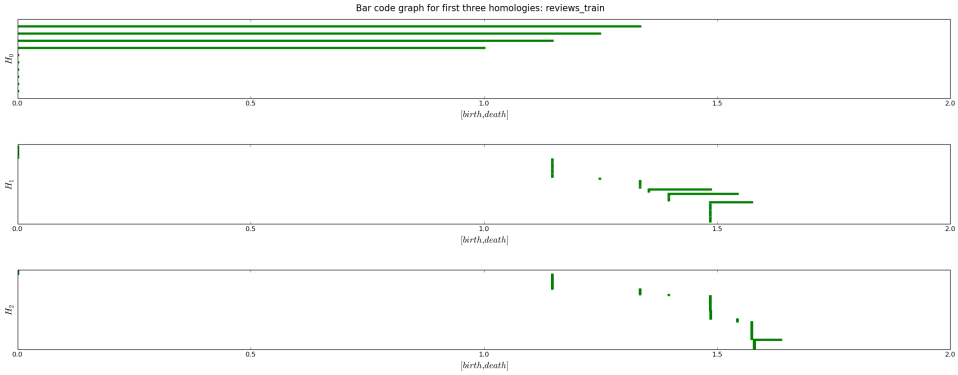

FIGURE 4.9. Bar codes for reviews_train

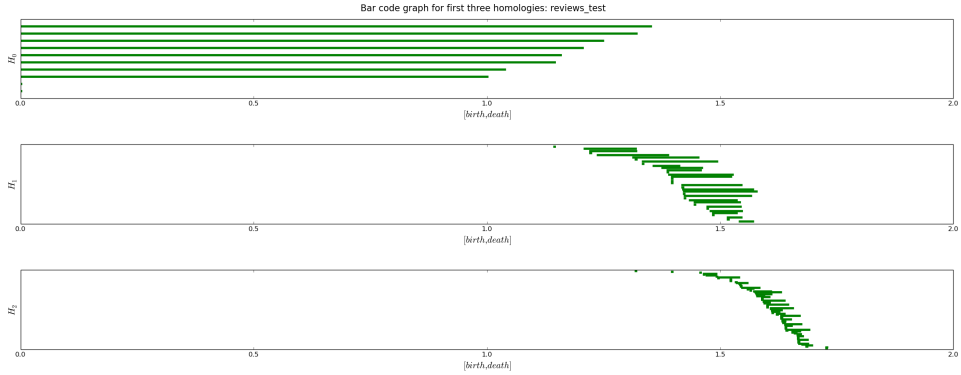

FIGURE 4.10. Bar codes for reviews_test

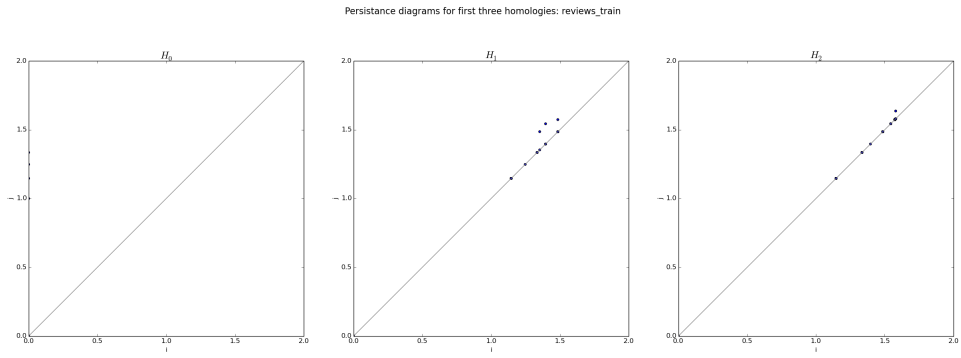

FIGURE 4.11. Persistence diagrams for reviews_train

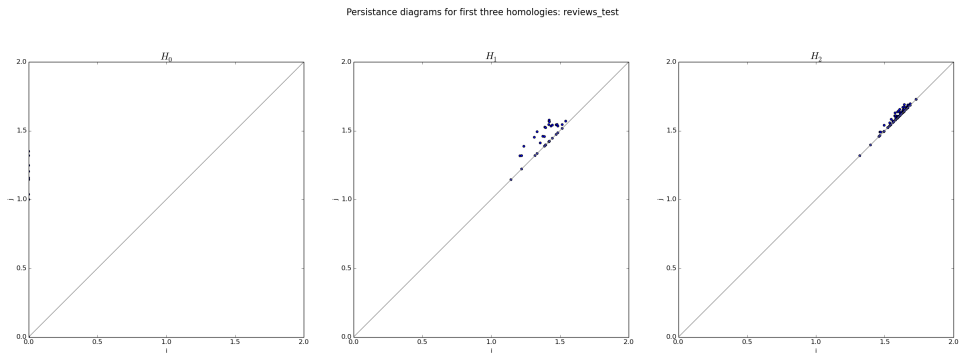

FIGURE 4.12. Persistence diagrams for reviews_test

To get concrete results on how much persistence diagrams differ between texts from different domains, we calculated bottleneck distances between all pairs of diagrams. The bottleneck distance between two diagrams is the cost of the optimal matching between points of the two diagrams. From the calculated diagrams a pairwise distance matrix was constructed, on top of which hierarchical clustering was performed. As already mentioned we performed this on 6 groups of documents where each group had a pair from the same domain. The main idea is that if persistence diagrams separate the documents from different domains well, each two groups of documents from the same domain, would be grouped

"sooner" in the hierarchical clustering than group of documents from different domains. Results can be seen in figure 4.13. We can first notice that abstracts and sports texts get connected sooner than abstracts and sports texts with its pair. This means that bottleneck distance between one group of sports texts and abstracts texts is the smallest distance between all 6 groups and that persistence diagrams between two groups of sports articles differ more than diagrams of different domains (at least by means of bottleneck distance). Although review pair gets connected, the distance between the two pairs is greater than all the distances between other four groups.

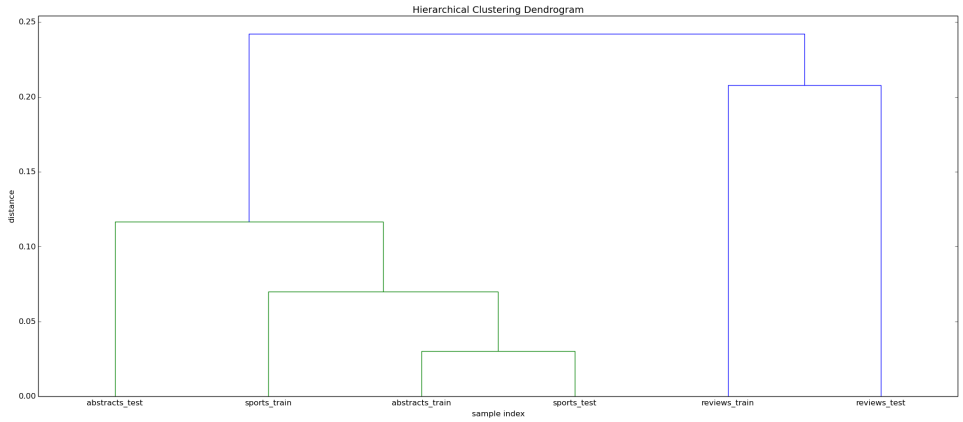

FIGURE 4.13. Hierarchical clustering results on texts dataset.

The results are not promising. One would expect that diagrams from groups of documents from the same domain would differ significantly less, than diagrams of different domains, and that inner-domain bottleneck distances would therefore be much smaller. Instead of using the bottleneck distance as a distance metric between diagrams we also tested the Wasserstein distance with no improvement in the results.

We also tested the hierarchical clustering method on a "toy" dataset where one sample of points were coming from a circle, and the other one form a straight line. The clustering had no trouble distinguishing between the domains and the expected results can be seen in 4.14. The inner-group bottleneck distances in both groups are much smaller than the distance between groups from different domains, which confirms the intuition of our test, and that the results would be expected of persistence diagrams differed enough.

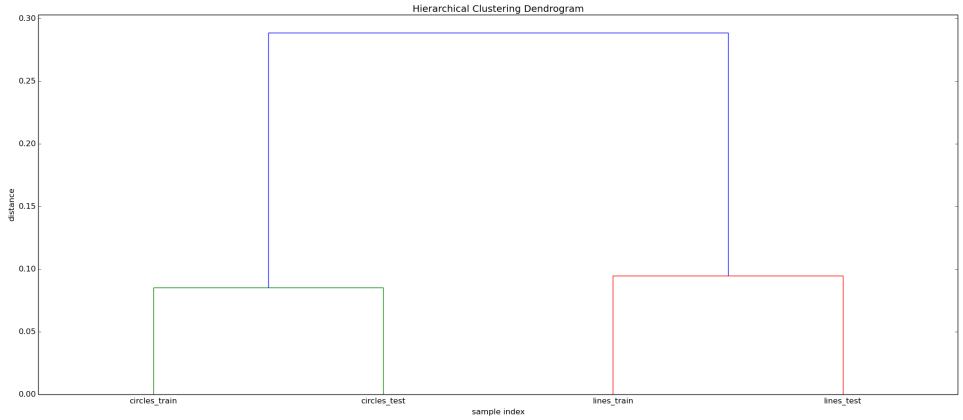

FIGURE 4.14. Hierarchical clustering results on a toy "circles and lines" dataset.

5. SUMMARY

Persistence diagrams built on top of documents from different domains did not differ enough to correctly classify documents based on bottleneck distances only. However it should be noticed that bottleneck distances between diagrams is only one possible way of using persistence diagrams for a predictive task. From diagrams, a various number of other numerical features can be extracted (e.g. number of homology generators of a specific dimension, average living length of a specific dimension etc.) so further exploration in this direction could provide promising results. Also the number of samples used in this project was relatively small (60), so testing the methods on a bigger corpora could provide different results as well.

We also show that this method works well on a toy example where homology of points in each group differ significantly. Therefore we believe that there are possible other applications (outside of text classification domains) that could benefit from the analysis of persistence diagrams.

We notice the main benefit of using persistence diagrams as a predictive task that it relies on inner-class data structure, instead on finding a linearly separable representation of the data, as is the case in many other clustering or classification models. Therefore we see it as an interesting tool for data analysis where other simple linear models would fail. The other important benefit of using this method is that it can provide us additional numerical features that can be used to further improve an existing model.

6. WORKLOAD

The work on this project was even more collaborative then was the case in the first project, meaning sometimes all three of us would work and contribute on the same issue. It is therefore hard to say who did what and give credit where credit is due. Nonetheless here is the distribution of the workload.

- A collected movie reviews dataset, implemented preprocessor that loaded text documents, combined them with pre-preprocessor, feature selection and tf-idf and constructed X and y matrices, implemented interfaces to Čech complex and Vietoris-Rips complex from Dionysus, wrote code that took outputs from all three complex building methods and constructed persistence diagrams from them, and wrote the glue that ran all the steps required for our analysis of the data;
- P collected the abstracts dataset, implemented the texts preprocessors which removed stopped words and applied stemming and lemmatization, implemented various methods for extracting text features, wrote code for hierarchical clustering on top of the bottleneck distances of persistence diagrams, performed clustering tests on the main dataset and implemented clustering on the toy dataset
- R collected the sports dataset, implemented the tf-idf method for extracting additional features, designed the drawing of barcode plots and persistence diagrams, performed benchmark clustering methods on the features matrix, chose the graphs for the presentation.

Writing the report was again a joined effort.