NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

THESIS

Clusterix: A visual analytics approach to clustering

Ilias Koutsakis

Supervisors: **Panagiotis Stamatopoulos,** Assistant Professor NKUA
Eamonn Maguire, CERN
Gilles Louppe, New York University

ATHENS

JULY 2016

THESIS

Clusterix: A visual analytics approach to clustering

Ilias Koutsakis
A.M.: 1115200800026

SUPERVISORS: **Panagiotis Stamatopoulos,** Assistant Professor NKUA
Eamonn Maguire, CERN
Gilles Louppe, New York University

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Clusterix: Συσταδοποίηση δεδομένων με οπτικοποίηση

Ilias Koutsakis

Επιβλέποντες: **Panagiotis Stamatopoulos,** Assistant Professor NKUA
Eamonn Maguire, CERN
Gilles Louppe, New York University

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Clusterix: Συσταδοποίηση δεδομένων με οπτικοποίηση

Ilias Koutsakis
Α.Μ.: 1115200800026

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Panagiotis Stamatopoulos,** Assistant Professor NKUA
Eamonn Maguire, CERN
Gilles Louppe, New York University

ΠΕΡΙΛΗΨΗ

Στο παρόν έγγραφο παρουσιάζω μια διαδικτυακή εφαρμογή ονόματι Clusterix, που στόχο έχει να υποστηρίξει τους χρήστες σε εργασίες συσταδοποίησης δεδομένων, έχοντας τους χρήστες-αναλυτές στο επίκεντρο της εργασιακής ροής. Το Clusterix, δίνει την ευκολία στον χρήστη να:

1. Φορτώσει και να εξετάσει αρχεία csv,
2. να επιλέξει συγκεκριμένες στήλες που θα χρησιμοποιηθούν απο τους αλγόριθμους συσταδοποίησης και να επιλέξει βάρη για τις στήλες αυτές,
3. να επιλέξει και να τρέξει έναν ή περισσότερους αλγόριθμους (K-Means, Ιεραρχική συσταδοποίηση) με διαφορετικές παραμέτρους,
4. να εξετάσει και να ανηλεπιδράσει με τα αποτελέσματα σε ένα διαδικτυακό περιβάλλον, και
5. να αλλάξει τις παραμέτρους της εισόδου για να διορθώσει τα αποτελέσματα της συσταδοποίησης.

Αυτή η επαναληπτική μέθοδος, με την χρήση οπτικοποίησης δεδομένων, επιτρέπει στους χρήστες να αντιληφθούν γρήγορα τον καλύτερο αλγόριθμο συσταδοποίησης και παραμέτρους για τα δεδομένα τους, και να διορθώσουν τυχόν λάθη στην έξοδο. Το Clusterix έχει χρησιμοποιηθεί για συσταδοποίηση και ανάλυση ετερογενών πηγών δεδομένων, και συγκεκριμένα στην συσταδοποίηση ακαδημαικών συγγραφέων και συσχετισμών τους με πανεπιστήμια, με στόχο τη δημιουργία ενός reccomendation system για το InspireHEP, την μεγαλύτερη ηλεκτρονική βιβλιοθήκη για High Energy Physics στον κόσμο, που αναπτύσεται στο CERN.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Οπτικοποίηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: συσταδοποίηση, μηχανική μάθηση, οπτική ανάλυση, ιεραρχική συσταδοποίηση, k-means

ABSTRACT

In this thesis I present a web-based, visual analytics tool called Clusterix to support clustering tasks by users, by having analysts at the center of the workflow. Clusterix provides the facilities to:

1. Load and preview CSV data;
2. select columns to be used by the clustering algorithm and modify weights;
3. select and run one or more clustering algorithms (K-Means, Hierarchical Clustering) with varying parameters;
4. view and interact with the results in a browser environment;
5. modify the parameters or input data to correct the clustering output.

Such an iterative, visual analytics approach allows users to quickly determine the best clustering algorithm and parameters for their data, and to correct any errors in the clustering output. Clusterix has been applied to the clustering of heterogeneous data sets, in particular to the clustering of author affiliations in publications, for a recommendation system on InspireHEP, the largest High Energy Physics library in the world, based at CERN.

SUBJECT AREA: Visualization

KEYWORDS: clustering, machine learning, visual analytics, hierarchical clustering, k-means

Dedicated to my mother, who stood by my side all these years.

ACKNOWLEDGEMENTS

I would like to thank my supervisor at CERN, Jan Age Lavik, who helped me throughout my year working beside him, and allowed me to work on this very personal project and complete my degree, as well as Gilles Louppe who helped jump start and provided his insights during the implementation of this project. But first and foremost, I would like to thank Eamonn Maguire, who not only had the original project idea, but guided me through the implementation for 7 months, providing his knowledge, brilliance and endless patience. He was not only a colleague, but also a teacher. Last but not least, I would like to thank professor Panagiotis Stamatopoulos, who accepted to be my thesis supervisor, helping me through this final push to obtain my bachelor degree.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PROLOGUE

Clusterix was implemented during my year working as a technical student at CERN, on InspireHEP. InspireHEP is the largest High Energy Physics database in the world, and is run by people of different backgrounds, including developers, physicists and curators/librarians. As explained in detail in the introduction, the main purpose of Clusterix is to provide a visual way to create a recommendation system for the needs of InspireHEP, more specifically to address the problem of automated author affiliation s extraction. We hope that the tool will be further developed and provide a useful service to users who want an intuitive way to explore datasets. The code that was developed for Clusterix, as well as installation instructions can be found at https://github.com/Lilykos/clusterix.

1. INTRODUCTION

We are living in an information age, where a constant stream of data, news, and ideas is flowing in every direction. The amount of information is unprecedented, and an important challenge of computer science right now is finding ways to understand these data.

The growth of all the data-driven fields of computer science is astonishing. There are new fields and job titles created for the sole purpose of making sense of data. Even more impressively, there is a re-branding and renewed enthusiasm for older concepts that are now more readily usable due to increased computer power (e.g. neural networks). Global interest on artificial intelligence and machine learning has only risen in recent years, with each new company and algorithm offering increased insights in to your data to improve a vast number of subject areas including the biological sciences, physics, business, and politics.

Such subject areas aim to use machine learning or data analytics in their respective fields in order to make sense of their data, that could possibly increase profit, throughput to customers, or help scientific breakthroughs in medical fields. Some well known examples of this are:

- Atlassian is using behavioral data to promote product growth in the company.
- Stanford has created a project trying to connect data science with modern medical research.
- The data produced from CERN’s Large Hadron Collider is approximately 15 petabyte per year, so Big Data methods and grid computing are being used to analyze them.

Of course, this means that more and more people are needed to work on these data, find ways to analyze them and extract meaningful results. Unfortunately, the tools of the trade are still in their infancy, as the planescape is constantly shifting, and new need emerge.

1.1 Motivation

InspireHEP is the largest High Energy Physics online digital library in the worlds, and fully hosted and implemented at CERN. It is a full and comprehensive library, hosting thousands of papers and publications, as well as a full set of metadata, e.g. citations, authors, collaborations, revisions, etc.

Trying to create a comprehensive catalogue, along with the right metadata from many different sources is no easy task. One of the main problems that arose during my time working on InspireHEP was the problem of extracting canonical affiliations for the authors. Depending on the source, the affiliation of an author could be quite different. For example:

- School of Physics, University of Melbourne, 3010, Victoria, Australia
- School of Physics, University of Melbourne, VIC, Melbourne, Australia

- School of Physics, University of Melbourne, Victoria, Melbourne, Australia
- School of Physics, University of Melbourne, VIC, Parkville, Australia
- School of Physics, University of Melbourne, Victoria, Parkville, Australia

So, although similar, all the previous affiliations are considered different. For a human, it is trivial to understand the context of the affiliation, and try to search for the right one online or in other documents, and this is exactly what is happening right now. However, as the project scales, it is becoming increasingly time-consuming to manually search for every single affiliation that seems incorrect.

Although a database of canonical names for a considerable amount of institutions and universities are available at CERN, searching through it is cumbersome and slow. The solution proposed was to build a self-learning recommendation system, that would match the current affiliation with the canonical one from the database instantly, as an extension module of InspireHEP.

The problem of fine-tuning this system remained though. The potential number of features that should be used is formidable. University name, laboratory, country, city, all need to be taken into consideration to create a successful recommendation system. So the real question was, how could we make sure that the recommendation system would work correctly? The solution lied in the field of visual analytics.

1.2 The Solution: Clusterix

Visual analytics is a field that combines data science and information visualization. By transforming scientific data into comprehensible visualizations, visual analytics take advantage of the natural tendency of the human eye to recognize patterns, and reach conclusions in a more intuitive way. It is much easier for a human to interact with data that are coupled together in obvious formations, and understand why this happens, instead of figuring out results from plain text and numbers. Generally, visual analytics give control back to the user in guiding how an algorithm is working

Considering the above, we decided that the best course of action would be to create a general-use tool, which would integrate unsupervised machine learning, visual analytics, and a user-friendly interface for the user to experiment. The tool was named Clusterix, and it tries to achieve three basic goals:

- Provide an intuitive interface with all the required options and parameters that clustering algorithms use, to the user.
- Visualize the data and provide interactivity, so that the user can easily understand why certain data items are grouped together and what makes them similar.
- Support an iterative approach that would allow a large amount of experiments to be conducted in a short time, making decision making faster and well documented.

Clusterix was created in order to make it easier to understand which clustering methods should be used for the affiliations recommendation system, as the large amount of features, most of them in text form, make this endeavor non-trivial. A prototype has been implemented, and is usable for generic use and exploration of data.

1.3 Thesis Structure

This thesis consists of the following parts:

- Clustering description, distance methods and categorization of clustering algorithms.
- The theory of cluster analysis, different categorizations of clustering and distance measures.
- Presentation and description of the cornerstone clustering algorithms: K-Means and Hierarchical Clustering.
- Text clustering, Natural Language processing methods for feature extraction.
- Implementation of Clusterix, technologies used and the basic workflow of the tool itself.
- Case study and description of Clusterix user workflow.
- Discussion on future work.

2. CLUSTERING FUNDAMENTALS

The first thing that needs to be addressed, is the basic building block of Clusterix: clustering itself. In order to approach clustering in a concise manner, the term should be clarified and defined first. According to Manning et. al. (2009) clustering algorithms are those algorithms that create sets of data called clusters. Clusters involve similar content. Each cluster is generally a subset of the total data, with more similarities between the members of the group, than those members in other clusters. So clusters are differentiated by content and context.

In the Figure 1 below, there are three distinct clusters. The data is distributed into different groups according to their characteristics and their content, and plotted in a way that visually makes sense to a user. In this specific example none of the data items is part of two clusters, so they are evenly distributed.

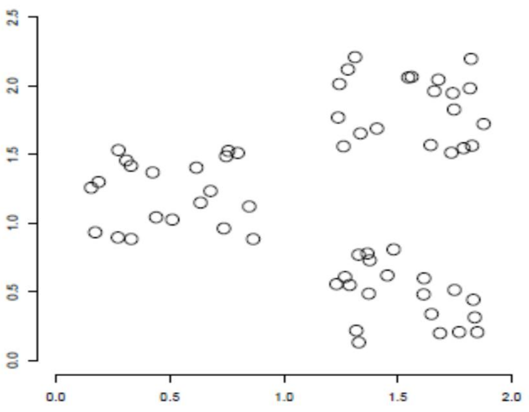


Figure 1: "Example of cluster structure"

2.1 Distance Measures

Clustering is used in unsupervised learning, during which the human factor does not engage in the learning process. There is no teacher or supervisor who has registered and categorized the training content. Instead, the algorithm tries to find out the relations between the data items simply by comparing distances between them.

It makes sense then, that the key element in clustering algorithms is the distance measure. Different distance measures can create different clusters, which makes it the most important measure to customize the exported result of clustering.

Clustering algorithms in general, are used to decide the clusters that each data item will be assigned to, by minimizing the distances between the centroid/center point of a cluster and the data points around it. As expected, the calculation of the distance is of grave importance in the use of clustering algorithms. Hence, the are some techniques for distance measurement such as "Cityblock", "Euclidean", "Cosine" and "Correlation" (Bora & Gupta, 2014).

Clusterix provides options for using most of the important distance measures, as it is one of the most crucial options for a clustering algorithm to choose. As will be explained later, the use of different distances makes more sense with different types of data sets.

Following, are the most important distance measures, that are regularly used by analysts and data scientists.

2.1.1 City Block/Manhattan distance

In the City Block or Manhattan distance we assume two points in the *xy*-plane. The City block distance is calculated as the distance in *x* plus the distance in *y*, is similar to the way we move in a city (like Manhattan) where we have to move around the buildings instead of going straight through. The city block distance of two points *a* and *b* with *k* dimensions is defined as:

$$\sum_{j=1}^{k} |a_j - b_j|$$

2.1.2 Euclidean distance

The shortest distance between the two points is along the hypotenuse, which is the Euclidean distance. The advantage of Euclidean distance is that the distance between any two objects is not affected by the addition of new objects to the analysis, as it is computed from raw data, not standardized. Yet, any differences in scale among the detentions can remarkably influence the distances. The Euclidean distance between two points, *a* and *b*, with *k* dimensions is calculated as:

$$\sqrt{\sum_{j=1}^{k}(a_j - b_j)^2}$$

2.1.3 Cosine distance

The Cosine distance between two points is one minus the cosine of the included angle between points, treated as vectors. Given an $m \times n$ data matrix X, which is treated as m $(1 \times n)$ row vectors $x_1, x_2, \ldots, x_m$, the cosine distances between the vector x_s and x_t are defined as:

$$d_{st} = 1 - \frac{X_s X_t'}{\sqrt{(X_s X_s')(X_t X_t')}}$$

2.1.4 Correlation distance

The Correlation distance is a measure of dependence between random vectors. Given an $m \times n$ data matrix X, which is treated as m $(1 \times n)$ row vectors $x_1, x_2, \dots, x_m$, the correlation distances between the vector x_s and x_t are defined as:

$$d_{st} = 1 - \frac{(X_s - \bar{X}_s)(X_t - \bar{X}_t)'}{\sqrt{(X_s - X_s)(X_s - X_s)'}\sqrt{(X_t - X_t)(X_t - X_t)'}}$$

2.2 Differentiating clustering methods

2.2.1 Flat and Hierarchical Clustering

There is a categorization about the kinds of clustering algorithms and there are different kinds of clustering according to Tan et. al. (2006) and later on Manning et. al. (2009), which are called flat and hierarchical.

In flat clustering there are flat results of clusters or subsets with clusters of one kind. The purpose of using this algorithm, is to create clusters that are internally coherent, meaning the content is explicitly differentiated between the clusters. In other words, in flat clusters we observe flat grouping of elements into groups in which none of the elements overlap another.

On the other hand, there is the hierarchical clustering which can create a hierarchy of clusters. Furthermore, we acknowledge sub clusters to main clusters, so sub groups merge into bigger groups. Different authors use another terminology to define hierarchical and flat clustering. According to Tan et. al. (2006) they use the terminology nested or un-nested to define hierarchical or partitional clustering.

2.2.2 Soft and Hard Clustering

There is another categorization of clustering in case of overlapping elements. With hard clustering, clusters do not overlap, which means that each element goes to one and only cluster and can't go to another or be part of more than one cluster. However, in the soft clustering method the clusters may overlap and there is the opportunity that one element can be part of two clusters at the same time.

3. CLUSTERING ALGORITHMS: K-MEANS AND HIERARCHICAL CLUSTERING

Clusterix is using two of the most basic and well-known clustering algorithms: K-Means and Hierarchical Clustering. Following is a primer on these algorithms and their inner workings.

3.1 K-Means

In order to understand the K-Means algorithm, it is important to analyze how the algorithm works. Initially, according to Tan et. al. (2006) the algorithm selects some K centroids, which constitute the chosen number of clusters. Subsequently, every other element is set near the closest centroid and every collection of centroids becomes a cluster. Later on, every centroid of each cluster has moved a little according to all of the elements of the cluster. The main process is completed when there are no more moves and changes of the elements into the cluster or the centroids become a fixed point or when no element changes clusters.

Approaching the algorithm further, we can analyze its steps. In the visual representations of the algorithm, as seen in Figure 2 all the centroids are illustrated by the symbol $+$ and every element that is part of the same cluster has the same marker shape.

The main algorithm is the following. First of all, the algorithm selects initially a point as an initial centroid. Afterwards, a loop starts in which each point is attached to the closest centroid with other common points and the centroid moves accordingly. The whole process continues until all the centroids stop changing, reaching convergence.

3.1.1 Convergence

Before ending the process of the algorithm the K-means algorithm results in no other centroid changes. The biggest movement in centroids should occur at the initial steps of the algorithm. A convergence to a result occurs when the algorithm observes less than 1$\%$ changes in the elements moving into different clusters.

3.1.2 Assigning points to the closest centroid

Euclidean (L_2) distance is used to point elements into Euclidean space. For the elements of a document it is preferred to use the cosine. Instead of the Euclidean distance, the Manhattan (L_1) distance can also used for Euclidean, ie inherent numeric data, while the Jaccard measure can be used for documents.

The whole process of the algorithm meets no significant difficulty as the algorithm calculates the common parts of the points with each centroid. On the other hand, the algorithm can be sped up when the data is in a low-dimensional Euclidean space, because by reducing the amount of dimensions, the number of necessary calculations is also reduced. Another

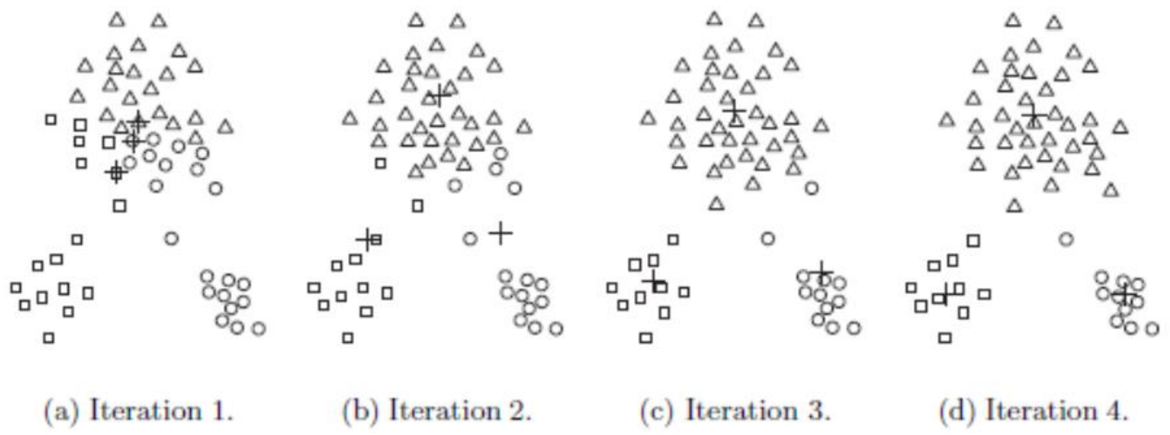


Figure 2: "Representation of the K-Means algorithm"

way to quicken the procedure is to bisect the K-means by compressing the number of calculations.

3.1.3 Centroids and Objective Functions

The main target of clustering is to reduce the distance between the elements and the centroids of the clusters. This is the objective function of clustering which relies on the proximity of the elements to each other or to the centroids.

3.2 Data in Euclidean space

Consider data whose proximity measure is Euclidean distance. In order to measure data in Euclidean distance we can use the squared error (SSE) known as scatter and we calculate the error of each data point. The whole process is to calculate the total sum of squared errors. After calculating the squared errors, i.e., the Euclidean distances, the calculation of the total sum of them follows. The result we prefer is the one with the smallest squared error. This means that the centroid of this specific clustering represents the points in their cluster more than any other cluster. In *SSE* the *dist* is the standard Euclidean (L_2) distance between two objects in Euclidean space as represented below:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(C_i, x)^2$$

Table 1: Symbols used in the equations throughout this chapter and their meaning.

X	An object
C_i	The i^{th} cluster
c_i	The centroid of cluster C_i
C	The centroid of all points
m_i	The number of objects in the i^{th} cluster.
M	The number of objects in the dataset
K	The number of clusters

According to the theory we can show that the centroid that minimizes the *SSE* of the cluster is the mean. Using the table above the centroid of the i^{th} cluster is defined as following:

$$c_i = \frac{1}{m} \sum_{x \in C_i} x$$

For example, the centroid of a cluster that contains three two-dimensional points as $(3, 5)$, $(2, 3)$ and $(7, 1)$. It will be $(\frac{3+2+7}{3}, \frac{5+3+1}{3}) = (4, 3)$

3.2.1 Document Data

In order to show that K-means is not only based on data in Euclidean space, we consider document data and the cosine similarity measure. The main point is to maximize the similarity of the documents in a cluster to the cluster centroid known as cohesion of the cluster. While in Euclidean data the cluster centroid is the mean, the analogous quantity to the total *SSE* is the total cohesion as seen in the following type:

$$TotalCohesion = \sum_{i=1}^{K} \sum_{x \in C_i} cosine(x, c_i)$$

In order to achieve document clustering, the necessary first step is to transform each document in a way that could be used in the context of Euclidean space. This process is called vectorization, and it involves transforming a document, e.g. a string, into a vector of numeric attributes, and will be explained in detail later.

3.2.2 Choosing clustering parameters

There are many different options regarding which attributes can be used by the K-Means algorithm. For instance, we may have as a proximity function the Manhattan (L_1) distance, with centroid median and as an objective function the minimization of sum of the L_1 distance of an object to its cluster centroid. Another option is to use Squared Euclidean (L_2^2) as a proximity function, with the mean as the centroid and as an objective function the minimization of the sum of the squared L2 distance of an object to its cluster centroid, as well. For the cosine proximity function, where the mean is the centroid, the objective is maximization of the sum of the cosine similarity of an object to its cluster centroid.

3.2.3 Choosing initial centroids

In order to choose the initial centroids, a random initialization is used. As a result, different runs of the K-means algorithm can produce different *SSEs*. In the following figure there are two different types of clusters. In the first one there is a global minimum of the *SSE* for three clusters, which is different from the second one. In the second figure there is a sub-optimal clustering that is only a local minimum. The most important step is to choose the proper initial centroids. When picked randomly, it can produce a poor result of clusters, as seen on Figure 3.

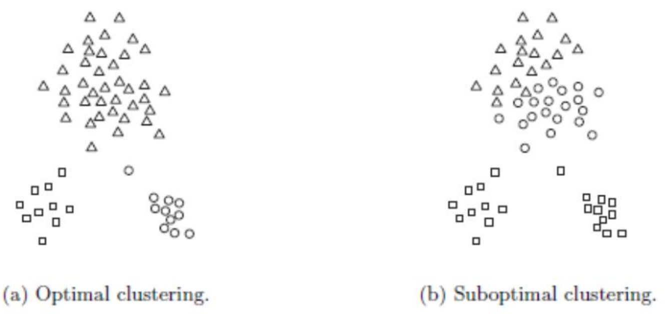


Figure 3: ”Representation of optimal and sub-optimal clustering”

A better strategy for picking the initial centroids was proposed by David Arthur and Sergei Vassilvitskii, and named $k-Means++$, which augments the initial K-Means algorithm with a simple, randomized seeding technique. This new way of creating the initial centroids has been shown to improve both the speed and the accuracy of K-Means.

3.2.4 Time and space complexity

The space requirements for the result of K-means algorithm are simple because only the data points and centroids are saved. The space that requires is $O((m+K)n)$, where m is the number of points and n is the number of attributes.

The time requirements of K-means algorithm are also simple. The time that needed is $O(IKmn)$, where I is the number of iterations required for convergence.

Summing, K-means is linear in m, the number of points, and is efficient as well as simple provided that K, the number of clusters, is significantly less than m according to Tan et.

al. (2006).

3.3 Hierarchical Clustering

The second, equally important type of clustering algorithm that will be described here, is Hierarchical Clustering. One of the biggest differences with K-Means, is that Hierarchical Clustering has no need of a prespecified number of clusters. Instead, by comparing features, this algorithm iteratively groups smaller clusters, so the number of clusters is determined by the dataset.

3.3.1 Agglomerative and divisive approaches

There are two standard approaches for using hierarchical clustering, the agglomerative and the divisive, according to Tan et al. (2006). Agglomerative hierarchical clustering is also known as bottom-up, while divisive is called top-down.

The bottom-up approach starts with the points as individual clusters and at each step merge (or agglomerate) the closest pair of clusters. However, the agglomerative requires defining a notion of cluster proximity.

The divisive approach starts with one, all-inclusive cluster and at each step a cluster is split until only singleton clusters of individual points remain. A decision must be made at this point, i.e. in which cluster to split at each step and how to do the splitting.

The first method, agglomerative, is more common than the other, and it is the one that Clusterix uses, so from now on, when using the term Hierarchical Clustering in respect to its usage by Clusterix, we mean Agglomerative Hierarchical Clustering.

3.3.2 Visual representation

Hierarchical Clustering is commonly represented as a tree-like diagram called a dendrogram, as seen in Figure 4. It appears both in relationships clusters and sub-clusters and when we merge (agglomerative view) or split (divisive view) the clusters. As stated by Manning et. al. (2009) each merge is represented by a horizontal line. The *y*-coordinate of the horizontal line is the similarity of the two clusters that were merged, where documents are viewed as singleton clusters. We call this similarity the combination similarity of the merged cluster. A fundamental assumption in Agglomerative Hierarchical Clustering is that the merge operation is monotonic.

Hierarchical clustering has no need of a pre-specified number of clusters. For sets of two-dimensional points, Hierarchical Clustering could be represented graphically using a nested cluster diagram.

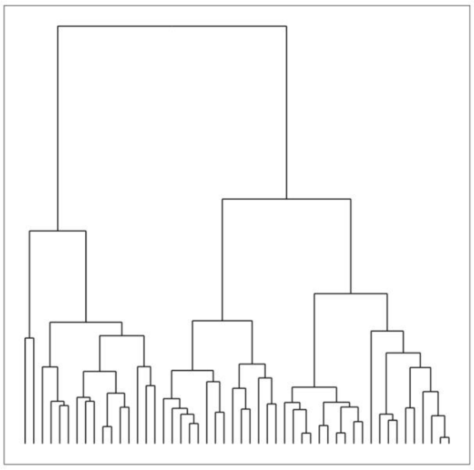


Figure 4: "An example dendrogram representing the output of a hierarchical clustering algorithm."

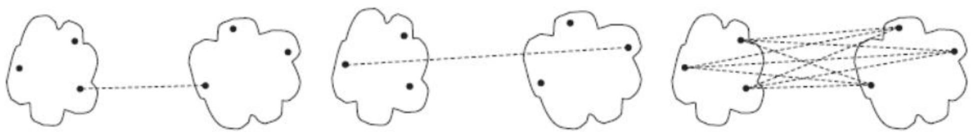
3.3.3 Defining proximity among clusters

According to Tan et al. (2006), cluster proximity is defined according to our point of view. For instance, some of the Agglomerative Hierarchical Clustering techniques come from a graph-based view of clusters such as MIN, MAX and Group Average. The MIN or single link technique considers cluster proximity as the proximity of the closest two points between different clusters. On the other hand, MAX or complete link, defines cluster proximity as the proximity between the farthest two points in different clusters. Finally, the group average technique defines cluster proximity to be the average pairwise proximities (average length of edges) of all pairs of points from different clusters. An example can be seen in Figure 5.

Alternatively, our point of view of the cluster proximity can be prototype-based, which means that each cluster can be described by a prototype, ie a data object that is representative of the cluster as a whole. In this case, each cluster is represented by a centroid and the proximity refers to the distance between clusters centroids. In this case Ward's method is used in order to measure proximity between two clusters in terms of the increase in the SSE. The Ward's method attempts to minimize the sum of the squared distances of points from their cluster centroids.

We will now present the most essential proximity functions:

Single Link or MIN: For the single link or MIN version of hierarchical clustering, the proximity of two clusters is defined as the minimum of the distance (maximum of the similarity) between any two points in the two different clusters. Using graph terminology, starting with all points as singleton clusters and adding links between points one at a time, shortest links first, then these single links combine the points into clusters. The single link technique is preferred when dealing with non-elliptical shapes, but is sensitive to noise and outliers.



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Figure 5: ”Graph-based definitions of cluster proximity”

Complete Link or MAX or CLIQUE: For the complete link or MAX version of hierarchical clustering, the proximity of two clusters is defined as the maximum of the distance (minimum of the similarity) between any two points in the two different clusters. Using graph terminology, starting with all points as singleton clusters and adding links between points one at a time, shortest links first, then a group of points is not a cluster until all the points in it are completely linked, i.e., form a clique. Complete link is less susceptible to noise and outliers, but it can break large clusters and it favors globular shapes.

Group Average: For the group average version of hierarchical clustering, the proximity of two clusters is defined as the average pairwise proximity among all pairs of points in the different clusters. This is an intermediate approach between the single and complete link approaches. Thus, for group average, the cluster proximity $proximity(C_i, C_j$ of clusters C_i and C_j , which are of size m_i and m_j , respectively, as shown below:

$$proximity(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} proximity(x, y)}{m_i m_j}$$

Ward’s Method In Ward’s method the proximity between two clusters is defined as the increase in the squared error that results when two clusters are merged. Ward’s method is very similar to the group average method when the proximity between two points is taken to be the square of the distance between them.

3.3.4 Algorithm usage on Clusterix

Trying to make Clusterix as complete and useful as possible, we added a considerable amount of options to the algorithm selection panels. By default, K-Means algorithm is using Euclidean distance, so the only option there is the number of clusters to be created. Hierarchical clustering, on the other hand, is much more elaborate, including:

- The most used linkage methods: Single, Complete, Average, Ward as well as some more obscure like Median and Centroid.
- A fair amount of distance methods, such as Euclidean, Cosine, Correlation, Manhattan, etc.

4. TEXT CLUSTERING

As stated in the Introduction, the original purpose of Clusterix was to provide an interactive environment for clustering datasets, including those who include text attributes. However, text clustering requires a certain pipeline of procedures to be implemented first, part of the process called Natural Language processing. Following, is the examination of what are the intricacies of text clustering, and the methods employed in Clusterix to increase the quality of the results.

4.1 Introduction

According to Aggarwal & Zhai (2012, pp. 77-128) clustering is helpful in the text domain, where the differences can be found in documents, paragraphs, sentences or terms. The main goal of text clustering is to organize documents which can assist the improvement of retrieval and support browsing.

Some toolkits to implement common text clustering algorithms are Lemur (The Lemur Project, n.d.) and BOW toolkit (Bow, 1998). The clustering can be applied for Document Organization and Browsing, Corpus Summarization and Document Classification according to Aggarwal & Zhai (2012, pp. 77-128).

Quantitative data clustering algorithms do not find positive applicability to clustering text data. The unique properties of text data demand the design of specialized algorithms. The characteristics can be summarized as follows:

- Even though a text representation consists of large amount of words, the given document may contain only a few hundred words. Therefore, the data that are to be grouped in clusters are few.
- The words of a lexicon are typically correlated with one another. This causes fewer concepts in a bigger space, thus creating the need for more specialized algorithms.
- It is important to normalize the specific text because the number of words vary.

All of the above are taken into consideration in Clusterix, and the methods used are described in detail on the following chapters.

4.2 TF-IDF

According to Aggarwal & Zhai (2012, pp. 1-19) the most significant word are not the ones that appear most often. These tend to be linking words such as *"the", "or", "and"* which are crucial to the structure of the document, but do not carry importance.

Instead, one of the main methods of text used for text processing is the vector-space based *TF-IDF* (*Term Frequency* $\times$ *Inverse Document Frequency*) representation (Salton, 1983).

Consider a collection of N documents. Define f_{ij} to be the frequency (number of occurrences) of term (word) *i* in document *j*. Then, define the term frequency TF_{ij} to be:

$$TF_{ij} = \frac{f_{ij}}{max_{kj} f_{kj}}$$

That is, the term frequency of term *i* in document *j* is f_{ij} normalized by dividing it by the maximum number of occurrences of any term (perhaps excluding stopwords) in the same document. Thus, the most frequent term in document *j* gets a *TF* of 1 and other terms get fractions as their term frequency for this document according to Aggarwal & Zhai (2012, pp. 1-19).

The *IDF* for a term is defined as follows. Suppose term *i* appears in n_i of the *N* documents in the collection. Then $IDF_i = log_2(N/n_i)$. The $TF - IDF$ score for term *i* in a document *j* is then defined to be $TF_{ij}IDF_i$. The terms with the highest *TF-IDF* score are often the terms that best characterize the topic of the document.

In the *TF-IDF* representation the inverse document frequency or *IDF* is used to normalize the term frequency for each word according to Aggarwal & Zhai (2012, pp. 77-128). The weight of terms which occur more frequently in the collection is decreased by the normalization further reducing the common terms. Furthermore, in spite of avoiding the influence of a single frequent term in a document sub-linear transformation function is often used.

4.3 Tokenization

Electronic text is a linear sequence of symbols (characters or words or phrases). As already mentioned, before any real text processing is to be done, text needs to be segmented into linguistic units such as words, punctuation, numbers, alpha-numerics, etc. IBM (Trim, 2013) defines tokenization as such: the acknowledgment of basic units that are processed, which units are considered basic for further analysis or process. The identification of units that do not need to be further decomposed for subsequent processing is an extremely important one. Errors made at this stage are very likely to induce more errors at later stages of text processing and are therefore very dangerous.

Different notions depend on different objectives or different language backgrounds. A token can be linguistically significant and methodologically useful.

The common process to identify tokens is with the special ”space” character (white space) in most languages. However, there are languages that do not use white space such as Chinese, Japanese or Thai.

Errors that may appear in tokenization can cause problems later on. There is a number of advanced methods that have been developed for complementing standard tokenizers. Another problem of tokenization is ”dirty text” which is unprocessed text and refers mainly to text stored in a database in fixed fields, with multiple lines per object.

4.4 Stopwords Removal

In general stop words also known with the terms ”stop word list” or ”stop list” are words or phrases used often in any language. Those words usually are excluded from many applications such as search engines, because they are used frequently and if included in the search, they could affect the result greatly. In other words, their presence in plenty of documents can steal the spotlight from main topic.

Such words could be *“a”, “an”, “the”, ”and”, ”are”, ”at”, “of”, ”be”, ”he”, ”is”, ”that”*, and many more. These words could make the clustering process much more difficult (Aggarwal & Zhai, 2012 pp.81-82). Yet, in addition to frequently used words stop lists may also contain rarely used words, as well, as they do not add anything of significance. Additionally, typographical errors and misspellings mainly coming from blogs or social networks.

According to Manning et. al. (2008) in order to create a stop list it is possible to consider some terms as the most commonly used by collection frequency, which is the amount of appearances of each term in the document collection. Stop words are mainly used in information retrieval systems and the lists contain either about 200 - 300 words or less, about 7-12 words but others claim (Aggarwal & Zhai, 2012 pp.81-82) that there are even larger stop lists. However, web search engines such as Google and Bing do not use stop lists.

The *TF-IDF* method could be used to gather frequent terms and assist in creating stop lists. The whole process of removal may help in the clustering process.

4.5 Stemming

According to Lovins (1968) the stemming algorithm is used to reduce all words with the same root or if prefixes are left untouched, the same stem to a common form by removing the various suffixes. Stemming is exploited by researchers in computational linguistics and information retrieval. In stemming the suffix and the affix of a word are striped so as to find the root (Waegel, 2011). Even though the root of a word might not seem important, it can be used to help in understanding the grammatical structure of a document or cohesion. The result of this algorithm is similar stems that can be used for stylistic or mathematical analysis and in order to calculate word frequencies of a text. However, problems still occur while using stemming.

The Porter stemmer is a stemming algorithm used as the default since 1980, because of its speed, readability and accuracy. In the Porter stemmer the suffix of the word is removed and reduces to a stem by applying 60 rules in 6 steps. In step 1 the algorithm removes plurals and the suffixes *”-ed”* or *”-ing”*. Step 2 is turning turning terminal *y* to *i* if no other vowel is in the stem, while step 3 maps double suffixes to single ones. Following, the algorithm removes the suffixes *”-full”*, *”-ness”* etc in step 4, but in step 5 it removes the suffixes *”-ant”*, *”-ence”* etc. Finally, step 6 removes the final *”-e”*.

To provide a concrete example, the words *"victory"* and *"victorious"*, would both return the stem *"victori"*. It is quite clear that this removes a lot of the noise, and makes text processing easier.

4.6 Vectorization

The Vector space model *(VSM)* was developed by Gerard Salton and his colleagues (Salton, Wong, & Yang, 1975) for the *SMART* information retrieval system (Salton, 1971). In the *VSM* each document in a collection is shown as a point in a space (a vector in a vector space) (Turney & Pantel, 2010). Points that are semantically similar are in close proximity of each other, while points that are dissimilar are distant from each other. In this space the query made by a person is a point in this vector space. Furthermore, the documents are sorted in order of increasing distance (decreasing semantic similarity) from the query and then presented to the user.

VSM is also used in natural language processing by some researchers because of its success. For instance it was applied to the Test of English as a Foreign Language *(TOEFL)* by Rapp (2003)in order to achieve automated scoring and evaluation, with impressive results. In addition, *VSM* was applied to the *SAT* college entrance test by Turney (2006) scoring as well as an average human.

VSMs perform well on tasks that involve measuring the similarity of meaning between words, phrases, and documents. Most search engines use *VSMs* to measure the similarity between a query and a document (Manning et al., 2008; Turney & Pantel, 2010). The leading algorithms for measuring semantic relatedness use VSMs (Pantel & Lin, 2002a; Rapp, 2003; Turney, Littman, Bigham, & Shnayder, 2003; Turney & Pantel, 2010).

The *D* is a document collection and *Q* the set of queries representing users' information needs. Let also t_i symbolize term i used to index the documents in the collection, with $i = 1, \ldots, n$. The *VSM* assumes that for each term t_i there exists a vector $\vec{t_i}$ in the vector space that represents it. It then considers the set of all term vectors $\{\vec{t_i}\}$ to be the generating set of the vector space, thus the space basis. If each $d_k, \forall k = 1, \ldots, p$ denotes a document of the collection, then there exists a linear combination of the term vectors $\{\vec{t_i}\}$ which represents each d_k in the vector space.

Similarly, any query *q* can be modelled as a vector $\vec{q}$ that is a linear combination of the term vectors. In the standard *VSM*, the term vectors are considered pairwise orthogonal, meaning that they are linearly independent. The similarity between a document vector $\vec{d_k}$ and a query vector $\vec{q}$ in the *VSM* can be expressed by the cosine measure given where a_{kj}, q_j are real numbers standing for the weights of term$j$ in the document d_k and the query *q* respectively as follows:

$$cos(\vec{d_k}, \vec{q}) \frac{\sum_{j=1}^{n} a_{kj} q_j}{\sqrt{\sum_{i=1}^{n} a_{ki}^2 \sum_{j=1}^{n} q_j^2}}$$

VSM provides several advantages. One of them is to automatically extract knowledge from a given corpus, which is faster and less exhausting than other approaches to semantics. Furthermore, in the *VSM* the terms are weighted by importance and make partial matches (Soboroff, 2002). On the other hand, the disadvantages of *VSM* include the assumption of terms as independent units and the fact that, even though the weighting is intuitive, it is not very formal.

4.7 Dimensionality Reduction

Advances in data collection and storage capabilities during the past decades have led to an information overload in most sciences. According to (Fodor, 2002) traditional statistical methods break down partly because of the increase in the number of observations, but mostly because of the increase in the number of variables associated with each observation. The dimension of the data is the number of variables that are measured on each observation. High-dimensional datasets present many mathematical challenges and opportunities, and are bound to create new theoretical developments. One of the problems with high-dimensional datasets is that, in many cases, not all the measured variables are important for understanding the underlying phenomena of interest.

4.7.1 Technique: Principal Component Analysis

Principal Component Analysis *(PCA)* is by far one of the most popular algorithms for dimensionality reduction (Pearson, 1901; Wold, 1987; Dunteman,1989; Jollife, 2002; Sorzano et al., n.d.). Being based on the covariance matrix of the variables, it is a second-order method (Fodor, 2002).

PCA is the default method for finding the single best (in the sense of least□square error) subspace of a given dimension, *m*, assuming that the data is zero mean and the subspace is linear. In essence, *PCA* seeks to reduce the dimension of the data by finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance.

In terms of dimensionality reduction it can be formulated (Hyvarinen, 2001) as the problem of finding the *m* orthonormal directions w_i minimizing the representation error $J_{PCA} = E\{|x - \sum_{i=1}^{m}(w_i, x)w_i|^2\}$. In this objective function, the reduced vectors are the projections $\chi = ((w_1, x), \ldots, (w_m, x))^t$.

4.7.2 Technique: Singular Value Decomposition

Another approach to the PCA problem, resulting in the same projection directions w_i and feature vectors χ uses *Singular Value Decomposition* (Golub, 1970; Klema, 1980; Wall, 2003; Sorzano et al., n.d.) for the calculations.

If a matrix *A* has a matrix of eigenvectors *P* that is not invertible, then *A* does not have an eigen decomposition. However, if A is an $m \times n$ real matrix with $m > n$, then *A* can be written using the so-called Singular Value Decomposition *(SVD)* of the form $A = UDV^T$ according to Mathworld wolfram. (n.d.).

5. IMPLEMENTATION AND TECHNOLOGIES

Clusterix was designed as a standard Python web application, based on the Flask micro-framework. Flask emphasizes simplicity and modularity instead of out of the box functionality, it is a much saner choice for Clusterix than other, more robust frameworks (e.g. Django), as the functionality required from the Flask itself is minimal, while all the plugins in the flask ecosystem can be added if necessary.

Although Clusterix is implemented as a web application, it is intended to be installed and used by individual users locally. It is not recommended to set up a public endpoint for Clusterix, due to various factors including data safety and performance. As such:

- Clusterix strives to be an easy to use tool for anyone who needs to cluster a variety of data sets. In order to move away from the toy project perspective, and turn it into a viable choice for data science, potential data confidentiality should be taken into consideration. This is why a public endpoint is not a safe route for Clusterix, as not only is the data file itself saved, but also the processed, indexed output of the file (more on that on the preprocessing section); and
- Data analysis, and the respective algorithms used, usually need considerable computer power in order to function within an acceptable time limit. In that case, the tools (Clusterix) are provided, tuned as well as possible, and it is up to the user or the company, to provide a powerful enough configuration (be it a computer, computing cluster, or a virtual machine) to use it.

The concept behind the implementation of Clusterix is to create a visual analytics tool with an iterative, feedback-based approach in mind. As stated on the introduction, the purpose of visual analytics is to give control back to the user in guiding how an algorithm is working. Clusterix achieves this through the provision of:

- Information about clusters, and the facility to compare clusters to see why particular points are closer together than others;
- validation tools to enable users to check if items they expect to cluster are close together; and
- An ability to change how information is processed by changing parameters on the fly and getting new results, without a single line of code.

A simple diagram that describes the model Clusterix is built on, is the following:

In the next part, I will give a description of the implementation and usage of the three components that Clusterix is comprised of. I will describe the various libraries and technologies used, as well as the reasons behind the choice. The project code, with installation instructions can be found at https://github.com/Lilykos/clusterix.

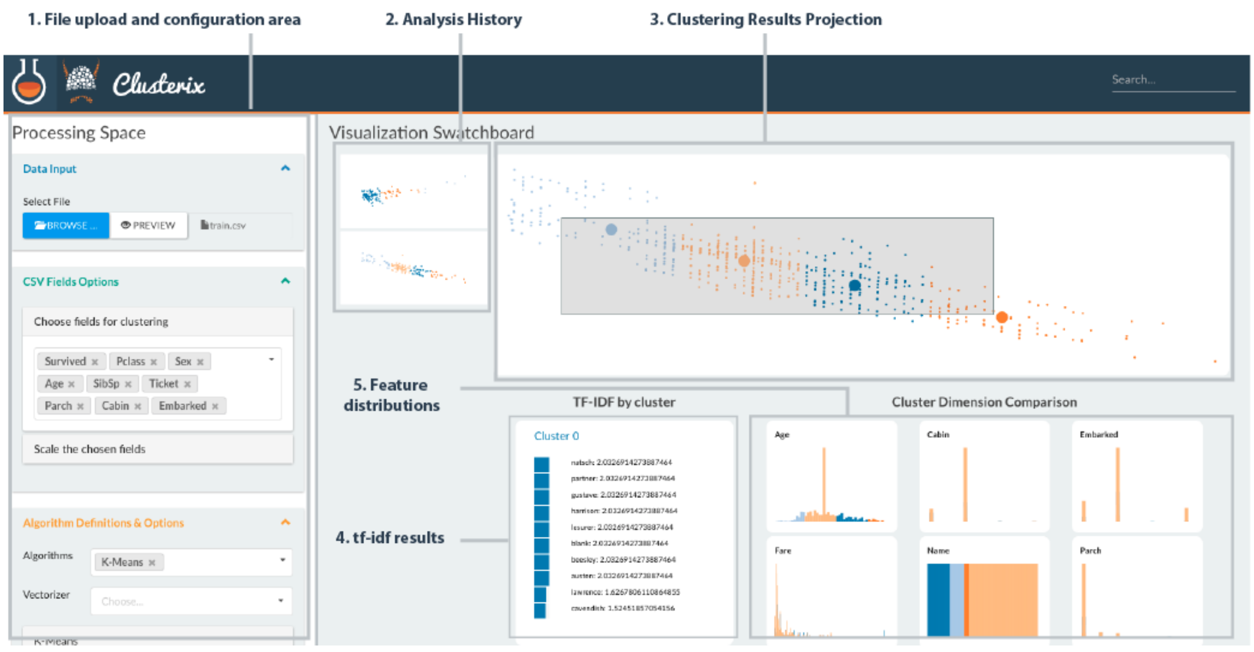
5.1 Input

5.1.1 Graphical User Interface

As mentioned before, the structure of Clusterix is that of a web application, albeit running locally. For that reason, the user interface was created using web technologies, something that makes the creation of new features and the dynamic addition/removal of options and results much easier.

Clusterix uses two CSS frameworks. The basis of the project is Bootstrap, which, due to its ease of use and plethora of modules, is the standard in modern web development. In addition to Bootstrap, we use certain elements from Semantic UI, more specifically the dropdown/options panels. This is because the out of the box capabilities of these elements make the implementation of Clusterix much easier. It is a very important part of the input, as the ease of changing parameters and extracting visual results is one of the main selling points of Clusterix.

JQuery is also used for all the standard addition and removal of elements on the browser, as well as Handlebars for the dynamic template system.



```
nabch: 2.0326914273887464
partner: 2.0326914273887464
gustave: 2.0326914273887464
harrison: 2.0326914273887464
lesurer: 2.0326914273887464
blank: 2.0326914273887464
beesley: 2.0326914273887464
austen: 2.0326914273887464
lawrence: 1.6267806110864855
cavendish: 1.52451857054156
```

Figure 6: "Clusterix's interface consists of five sections to support the analysis tasks of users."

The main modules of the Clusterix interface are presented in Figure 6, and annotated as such:

1. **The file upload and configuration area**, where the used uploads the data set and sets the parameters.
2. **Analysis history**, using layered mini scatter plots show the output from previous analyses and will be explained more later.
3. **Clustering results projection** is the main scatter plot where the visualization and

interaction happens.

4. **tf-idf results** show the most commons words within each selected cluster.
5. **Feature distribution** plot the distributions for all values within the selected area. This allows users to see which features are more important for clustering the data, so that they may adjust the selected fields in Panel 1 to remove any known biases for example.

5.1.2 File Input

Right now, the only type of input that Clusterix supports is csv files. In order to make the handling as efficient as possible, I am using Papaparse.js, a fast and well-supported library with an abundance of useful features. More specifically, Papaparse supports automatic header parsing, local and remote file input, and automatic delimiter detection. This last feature is especially useful, as the library of choice for the csv file processing (Pandas), is not always clear on what the delimiter is, and there are many possible types like comma, semicolon, white space and tabs.

The input panel used is a Bootstrap plugin, Bootstrap File Input. It is also very extensive, which allowed me to rebind the buttons used to create a preview (rendered with bootstrap modals) of the uploaded data, seen in Figure 7.

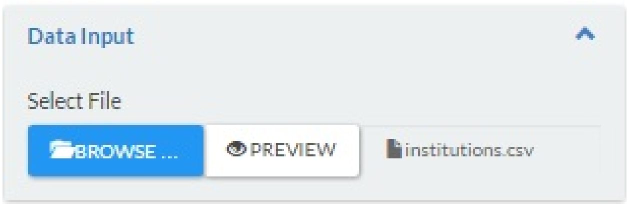
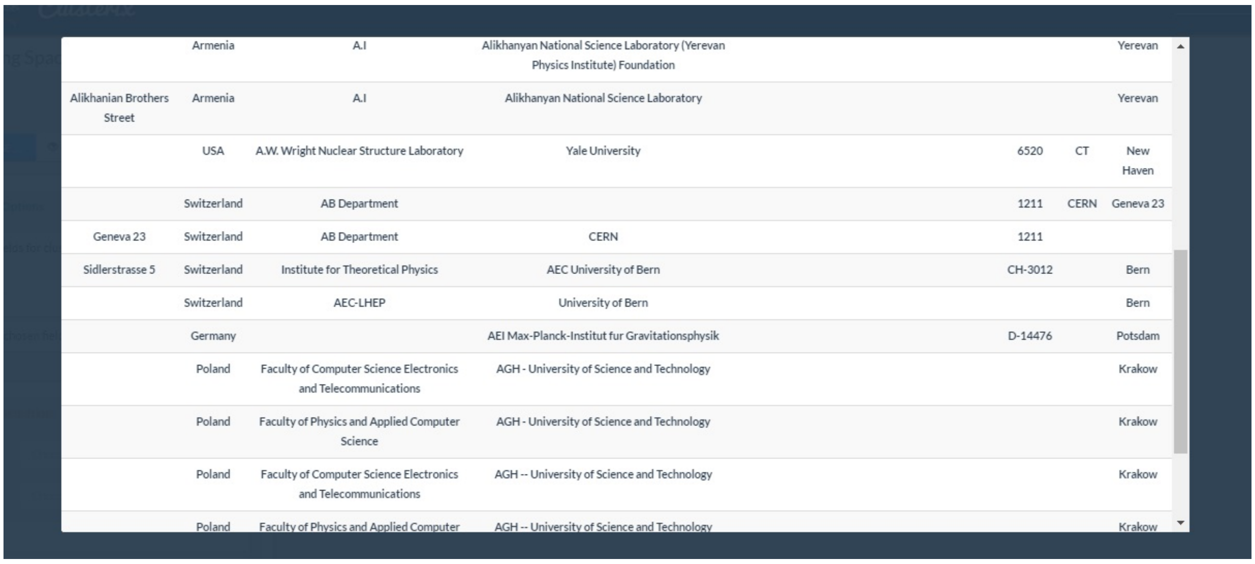


Figure 7: ”Clusterix data input panel”

The most important part of the input, is the parsing of the headers. The headers are used in the CSV Options panel, where the user can choose which columns should be taken into consideration for the preprocessing and clustering that is to follow. Furthermore, the user can provide different scalings for different columns, in order to give a statistical advantage to the dataset features that are considered more important. This is a great advantage for Clusterix, as the main reason to use it is the ease of experimentation and feedback of results. It also allows us to create a preview of the data, in the form of a modal, as seen in Figure 8. The user/data scientist is able to experiment with different numbers to “guide” the results according to his/her perceptions and external information (user stories), which has the probability of being very useful in cases of unsupervised learning, which clustering is.



	Armenia	A.I	Alikhanyan National Science Laboratory (Yerevan Physics Institute) Foundation			Yerevan
Alikhanian Brothers Street	Armenia	A.I	Alikhanyan National Science Laboratory			Yerevan
	USA	A.W. Wright Nuclear Structure Laboratory	Yale University	6520	CT	New Haven
	Switzerland	AB Department		1211	CERN	Geneva 23
Geneva 23	Switzerland	AB Department	CERN	1211		
Sidlerstrasse 5	Switzerland	Institute for Theoretical Physics	AEC University of Bern	CH-3012		Bern
	Switzerland	AEC-LHEP	University of Bern			Bern
	Germany		AEI Max-Planck-Institut fur Gravitationsphysik	D-14476		Potsdam
	Poland	Faculty of Computer Science Electronics and Telecommunications	AGH - University of Science and Technology			Krakow
	Poland	Faculty of Physics and Applied Computer Science	AGH - University of Science and Technology			Krakow
	Poland	Faculty of Computer Science Electronics and Telecommunications	AGH -- University of Science and Technology			Krakow
	Poland	Faculty of Physics and Applied Computer	AGH -- University of Science and Technology			Krakow

Figure 8: ”Clusterix preview modal, after uploading a universities dataset”

5.1.3 User-provided parameters

As soon as the data file is uploaded, the user is able to select and manipulate a variety of parameters and options, in order to get the best possible results. This is an important part of the tool, as the idea behind Clusterix is to enforce an iterative approach in data exploration. As soon as the tool runs for the first time, the data get processed and saved locally, so that the next iterations will run significantly faster. The user can change the values that, according to the observations, affect the results more, and see if the results actually justify the observations.

The parameters provided by the user are the following:

- Csv file options (seen in Figure 9):
  - Columns to be used in clustering.
  - Scale (if necessary, default is 1) of any selected column.
  - The delimiter, because as said before, Papaparse is a safer choice on the automatic detection of delimiters.
- General Algorithm Options (seen in Figure 10):
  - Algorithms that should be used (K-Means and Hierarchical clustering are the only ones supported at this moment)
  - Vectorizers (Count, Hashing, TF-IDF)
- Specific options for each algorithm:
  - Number of clusters for K-Means.
  - Number of clusters in Hierarchical Clustering (cut-off), as well as distance methods (single, complete, average, etc) and affinity (euclidean, ward, cosine, etc).

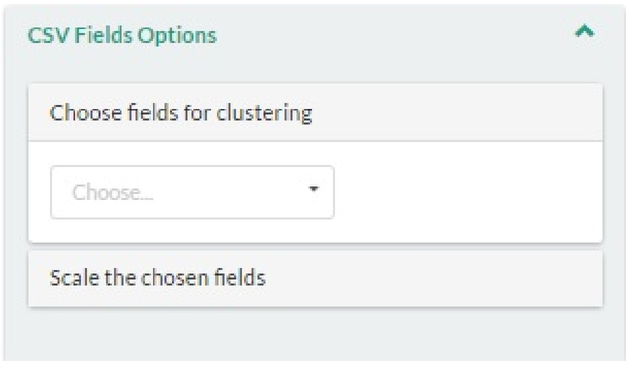
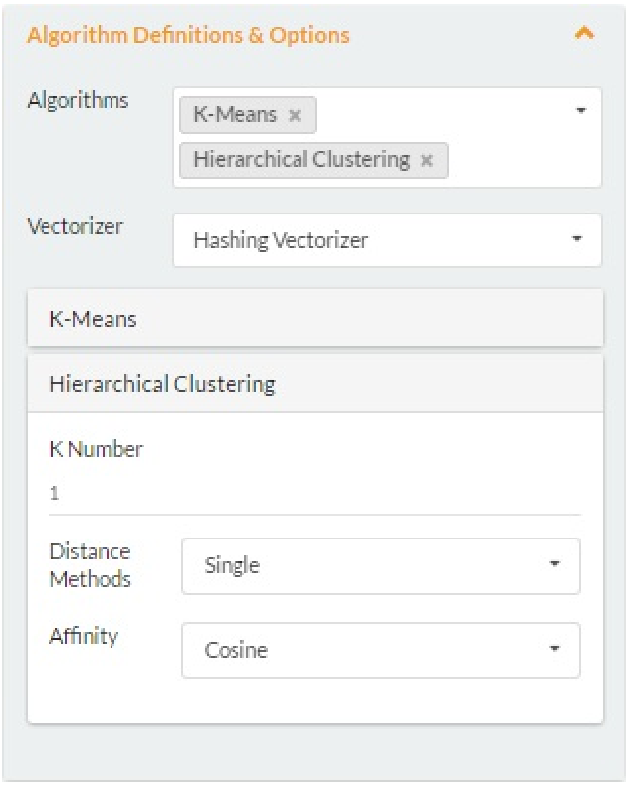


Figure 9: "Csv input options"


Algorithm Definitions & Options

Algorithms: K-Means, Hierarchical Clustering

Vectorizer: Hashing Vectorizer

K-Means

Hierarchical Clustering

K Number: 1

Distance Methods: Single

Affinity: Cosine

Figure 10: "Algorithm selection and options, including vectorizers"

All the abovementioned information is sent through an Ajax post to the server, and determines the options for the preprocessing and clustering jobs that are to follow.

5.2 Processing and results

5.2.1 Cleaning and normalizing data

The first step in any data analysis pipeline, is to make sure that the data is actually usable. Taking into consideration that datasets can have a variety of information, which is not always complete, the dataset should be put into a transformation pipeline to be cleaned and normalized. The tool of choice for this step is Pandas, a very powerful data manipulation library for Python. Pandas is centered around the concept of the dataframe, which is a two-dimensional data structure that accepts heterogenous data, making it ideal for a csv file which could have any number of different data types (integers, floats, strings, etc). Two standard methods for this procedure are being in use:

- In any numeric column, empty spaces are replaced with the median of that column;
- in any string column, the empty spaces are replaced with the "NaN" string, as a neutral placeholder. This is especially important in cases of string vectorization (to be presented later), where an empty input could make the process throw an exception.

The usage of the methods presented above, returns a normalized and relatively clean dataset. It needs to be mentioned that, because Clusterix is a visual analytics tool, this procedure is minimal and follows recommended methods without going into depth. If the dataset is in need of different methods, according to the needs of user, they should be implemented before using Clusterix to get results.

5.2.2 Data Storage

The next part of the preprocessing involves storing the data in a way that will make the access to them easy and most importantly fast. As the first and foremost problem area for Clusterix is processing and clustering of text data, there are two use cases that need to be taken into consideration:

- In order to use text data in accordance to clustering algorithms, the data need to be preprocessed with the usual Natural language Processing techniques (stemming and stopword removal). Thus, a data storage is needed that will keep track of the pre-processed input.
- In addition, statistical information is very much essential to any NLP task, e.g. TF-IDF for a certain group of input data. This is why a second store is created, for the purpose of keeping the vocabulary of each datum, without stemming but after the removal of stopwords.

In order to complete the tasks mentioned above (preprocessing and storage), I am using a pure-Python library named Whoosh. Whoosh is mainly used for full text indexing and the creation of custom search engines (e.g a blog), but it contains an amazing wealth of classes and functions that can be easily utilized to simplify some procedures.

- The Analyzer classes are to be used out of the box for text processing jobs. In addition, they can be chained, so it is easy to use StemmingAnalyzer in addition to a certain filter, or StandardAnalyzer for the extraction of the vocabulary.
- The index-based database that Whoosh creates, does not need to be given the schema explicitly. In contrast, it allows the user to provide the fields for the used dataset using *kwargs, which allows Clusterix to work for any amount of different datasets, as long as they are tabular, without changing any code whatsoever.

In conclusion, any time a new Clusterix project gets started, the input fields are being sent to create a new Whoosh schema, which then populates two databases with processed data and vocabulary data, all after the required preprocessing. The processed data will be used in the next step for clustering, whereas the vocabulary is available in case the user want to get more specific information later, through interaction with the results in the browser.

5.2.3 Technology behind the clustering algorithms

For this part, almost all the functionality comes from Scikit Learn, arguably the best machine learning library for Python. Scikit Learn is built on the foundations of numerous other scientific and numeric python packages, like Numpy and Scipy, and has a full collection of tools for data mining and analysis. In addition to this, Scikit Learn has a certain API philosophy that allows the developer to easily create data pipelines using not only the built-in classes themselves, but also by extending them using transformer mixins. The process is as follows.

5.2.4 Data pipeline

As mentioned before, the user is provided with a certain number of options that can be used to tune the whole process. By using these parameters in accordance with the previously mentioned steps, it is fairly trivial to create a pipeline of transformers, using as input the Whoosh index containing the preprocessed data, in order to create the vectors that will be the clustering algorithms input.

More specifically, after the user selects what fields are to be used and their respective scaling, there is a preconfigured pipeline of transformers for the input:

- ItemSelector: returns a vector of the data that corresponds to a specific field from the index.
- Vectorizer: if the resulting vector contains strings, then it is vectorized using one of the 3 provided vectorizers (Hashing Vectorizer, TfIdf Vectorizer and Count Vectorizer); in other case, it remains as is, as all other kinds of numeric attributes can be used without an issue.
- Scaler: scales the vector by the number provided by the user.

The above process happens for each one of the fields that the user picked, and is followed by dimensionality reduction. Having in mind that the final target of Clusterix is to present the results in an intuitive and visual manner, it makes sense that the probably huge vectors that are created for each datum need to be reduced to a two-dimensional space, to be visualized in a scatterplot. The algorithm of choice here is the Truncated SVD, mainly for performance reasons. Truncated SVD accepts a sparse matrix as an input, which means that the resulting matrix does not need to be transformed into a Numpy array or a standard two-dimensional array in order to be usable. This potentially saves quite some time as the data size gets bigger.

5.2.5 Clustering and results

Since the bulk of the work is done in the preprocessing step, clustering is simply the correct usage of the Scikit learn algorithms. Clusterix currently supports 2 types of clustering: K-Means and Hierarchical Clustering. The number of clusters, as well as the specific parameters for hierarchical clustering (distance function and affinity) have already been provided by the user.

The above process results into a json file that contains all the needed information for the visualization of the data:

- The number of the clusters;
- the cluster centers (centroids) with their respective coordinates; and
- the data, presented as nodes with their assigned cluster, their id, and their coordinates (essentially the two-dimensional vector that resulted from the vectorization and dimensionality reduction).

The json result would look something like this:

```
{’k_num’: 2,
 ’nodes’: [
     {’cluster’: 0, ’isCentroid’: False, ’id’: 2,
      ’x’: 2.65806, ’y’: 1.732050},
     {’cluster’: 0, ’isCentroid’: True,
      ’x’: 2.22044, ’y’: 1.73205},
     {’cluster’: 1, ’isCentroid’: False, ’id’: 1,
      ’x’: 1.50345, ’y’: 0.0},
      .............]}
```

This is the basis of the visualization, explained in the next part.

5.3 Output

As soon as the results are returned to the browser, they are rendered in a scatterplot, which is a mathematical diagram used to display values in a two-dimensional space. Usually the rendering involves showing two specified variables for a set of data, but this is not the case here. The main scatterplot is Clusterix is somewhat abstract, in the sense that all the values from the selected fields have been reduced into a two-dimensional space with the intention of showing the general similarity of different data nodes, based on distance. The fields do not have a specified axis, although the user is able to figure out the fields that mainly contribute to the visualization, as will be explained later.

The rendering is implemented using d3.js, a Javascript library for visualizing documents using web technologies, mainly HTML, CSS and SVG elements. D3 uses a data driven approach to bind data to the DOM (Document Object Model) of the browser, and then transform them accordingly.

5.3.1 Scatterplots

Clusterix utilizes two different scatter plot areas shown in Figure 11, all rendered using d3.js:

- A mini, non interactive scatter plot on the left side of the visualization swatch board. This is a mini version of the currently clustered data, and its main point is to provide quick comparisons between the different iterations of clustering. By keeping these mini representations visible at all times, the user can see the differences at a glance of an eye, and make more concrete choices about feature selection and parameter tuning.
- The main scatter plot in the middle of the swatch board. This plot gets re-rendered every time a new clustering task is completed, and updated with the new data. Its main feature is interactivity, as the user can select (brush) an area and explore the data. Clusterix is supposed to help the user understand why clustering works as such, why certain nodes are closer together and what similarities they have. By brushing, the user is enabled to accomplish those things, by using the third and last type of scatter plot.

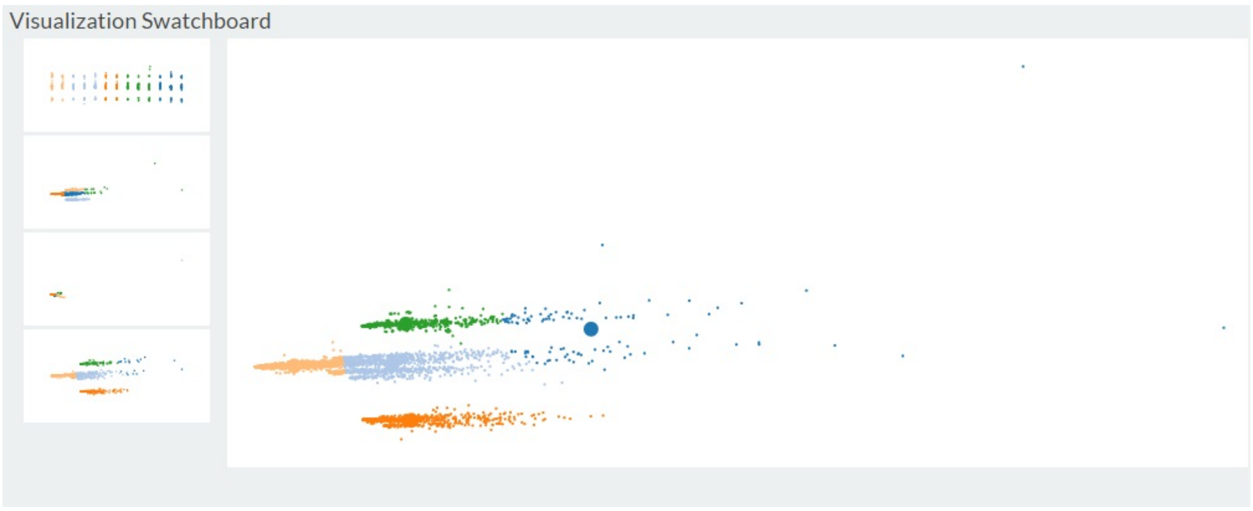


Figure 11: ”The main scatterplots of Clusterix”

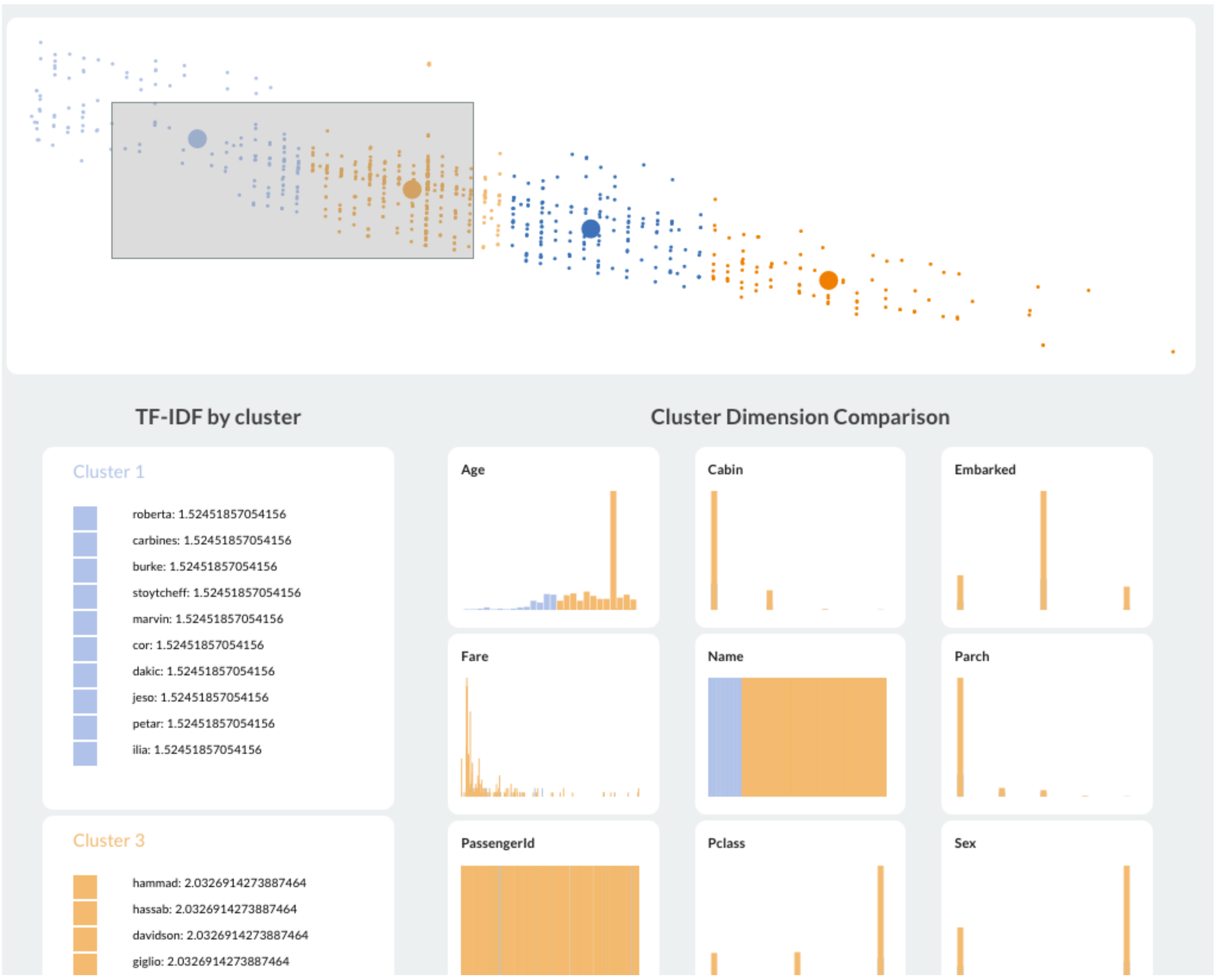
5.3.2 TF-IDF and Feature Distributions

Figure 12 shows how Clusterix displays additional information to the user about the clusters they have selected. In this example, we have both text and numeric features in the data set therefore both the tf-idf and dimension comparison panels are shown.

In this example, showing data on titanic survivors from Kaggle[1], one can clearly see that the feature/dimension that appears to be most important in distinguishing between clusters

[1]https://www.kaggle.com/c/titanic/data

1 and 3 is the age. A user could refine their clustering based on this to either remove age for consideration by the clustering algorithm, or to remove other features that carry little information. Other information is more evenly distributed.



Cluster 1

roberta: 1.52451857054156
carbines: 1.52451857054156
burke: 1.52451857054156
stoytcheff: 1.52451857054156
marvin: 1.52451857054156
cor: 1.52451857054156
dakic: 1.52451857054156
jeso: 1.52451857054156
petar: 1.52451857054156
ilia: 1.52451857054156

Cluster 3

hammad: 2.0326914273887464
hassab: 2.0326914273887464
davidson: 2.0326914273887464
giglio: 2.0326914273887464

Figure 12: ”The tf-idf section (showing most important words), and feature distributions (showing distributions for all features in the data set).”

5.3.3 Search

Another interesting feature of Clusterix is the capability to interactively search in the scatter plots for certain keywords. When the user searches for a word, the scatter plot will be re-rendered and the nodes that matched the query will be highlighted, enabling the user to understand better if the clustering was successful according to his needs. This, in combination with the TF-IDF charts, makes Clusterix a very powerful tool for quick deductions on text data.

6. CASE STUDY: CLUSTERIX ON AUTHOR AFFILIATIONS EXPLORATION

In this Chapter, we will investigate how Clusterix can be applied to the clustering of data sets to show its power in providing increased insight to users.

6.1 Case Study: University Affiliations

As the original reason for creating Clusterix was to investigate university affiliations data, and create a recommendation system, we will employ this data set and investigate whether or not we can achieve an acceptable clustering result. The main point of this experiment is to follow the process of getting a data set, investigating it, and understanding if by using Clusterix, we can cluster data logically, in a way that similar universities will be close by.

6.1.1 Data set description

The dataset provided by CERN is a text file containing a few thousand affiliations, directly parsed from papers, conferences and publications, found in the InspireHEP database. That means of course, that pre-processing is needed in order to transform the data set into a CSV file, the only kind of input currently accepted by Clusterix.

In order to parse bibliographic metadata from papers, InspireHEP is using a tool called GROBID (GeneRation Of BIbliographic Data). GROBID is trained on bibliographic sources, and is able to parse and extract data from PDF files, as well as raw strings. So, for example, in the case of the following string:

Department of Mathematics, Harvard University, Cambridge, MA, 02138, USA,

GROBID parsing would return the following XML result:

```
<affiliation>
        <orgName type=”department”>Department of Mathematics</orgName>
        <orgName type=”institution”>Harvard University</orgName>
        <address>
              <postCode>02138</postCode>
              <settlement>Cambridge</settlement>
              <region>MA</region>
              <country key=”US”>USA</country>
        </address>
 </affiliation>
```

which of course would be used afterwards to create a tabular data structure, with the following fields/features (including some missing from the example above):

- **institution**
- **department**

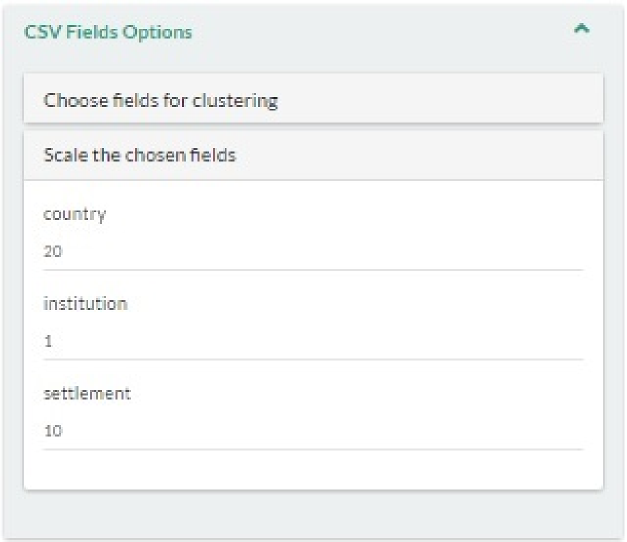
- **laboratory**
- **country**
- **region**
- **settlement**
- **address**
- **post box**
- **post code**

It is obvious that some of them are more useful than the others, e.g there is not much that one can do with the ”post box” feature, as this data set includes information about universities from all over the world.

6.1.2 Investigating with Clusterix

After uploading the data set, the first move is to decide on the features that should be used. Feature selection is very important, as random noise from features that don’t fit well together could completely invalidate any results. A simple way to start with this data set is to use only the necessary features: country, settlement (city), institution.

In cases like this, where the feature selection has a clear structure, the scaling option of Clusterix is very useful. We need to cluster universities geographically, so the order of feature importance is country, settlement, institution. We scale these features according to this order, like shown in Figure 13



country
20

institution
1

settlement
10

Figure 13: ”Scaling the selected features in Clusterix”

The best option for this kind of data would be hierarchical clustering, but due to lack of computer power, one could also use K-Means with satisfying results. We will select a cluster number of 10, which is arbitrary but, if the process is working correctly, sub-clusters will be evident after introspection. The result of this clustering is shown in Figure 14

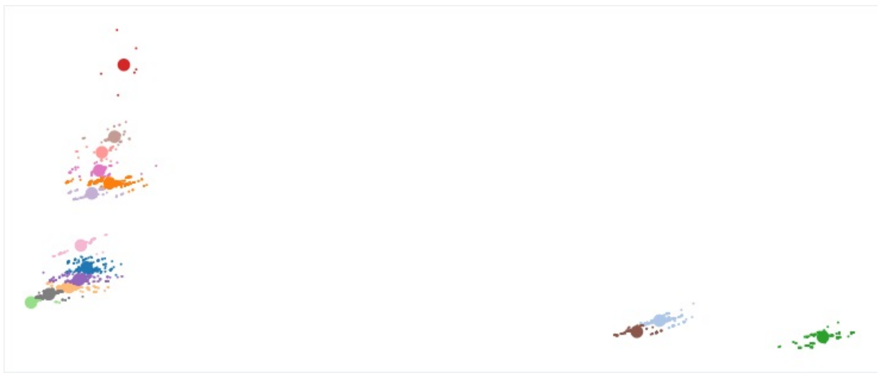


Figure 14: "Clustering the institutions data set using K-Means"

The results seem to be quite polarizing, so the logical next step is to see if the sub-clusters make logical sense. Let's start with the bottom right green cluster. After selection, we can take a look into its nodes and TF-IDF information. The top 5 TF-IDF scores are:

1. plymouth: 1.45116
2. hull: 1.45116
3. glasgow: 1.08229
4. manchester: 1.066378
5. liverpool: 1.062643

It is apparent that this cluster represents all the United Kingdom universities. Clusterix allows the user to make sure of that through the detailed information panels as shown earlier.

Clusterix's interactive environment could also be used to make sure that the sub-clusters are logically connected. For example, consider Figure 15.

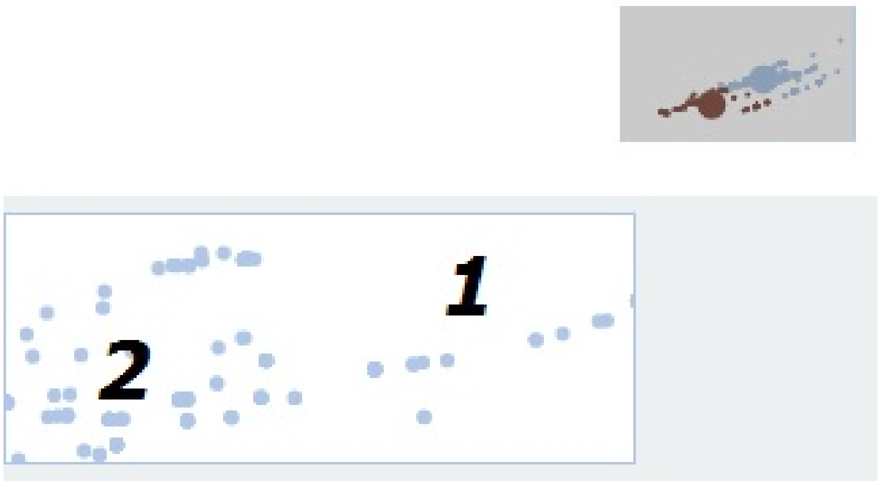


Figure 15: "Distances of nodes in clusters"

If we move the mouse over the nodes close to 1 and close to 2, it will be clear that, although both of these are part of the USA sub-cluster, the regions have differentiated their distances. This is the vocabulary of nodes close to the areas:

- Area 1:
  - department, **physics**, columbia, university, **new**, **york**, **new**, **york**
  - department, **physics**, **new york**, university, **new york**
  - stony, brook, university, department, **physics**, astronomy, stony, brook, university, **new**, **york**, **new**, **york**
  - department, **physics**, astronomy, state, university, **new**, **york**
  - nan, yang, institute, theoretical, **physics**, department,stony, brook, university, stony, brook
- Area 2:
  - center, quantum, mathematics, **physics**, qmap, department, physics, university, **california**, davis
  - department, mathematics, university, **california**, santa, barbara
  - berkley, center, theoretical, **physics**, university, **california**, department, **physics**, university, **california**, santa, barbara
  - church, street, department, **physics**, michigan, center, theoretical, **physics**, university, michigan, arbor

The very idea of allowing the user to dive into the data, and see the differences in distances and what they represent, as well as empowering him/her to reiterate the process and confirm that the false positives have been fixed (after tuning some parameters), is extremely intuitive and allows fast experimentation.

6.2 Case Study: Wine Classification

To show the general applicability of Clusterix to not only text data, but also typical classification problems, we will show how we can use Clusterix to classify wines from a data set available at the University of California, Irvine's machine learning data set repository[2].

6.2.1 Data set description

The dataset is a CSV file containing 1600 wines with 12 features including:

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol
- quality

6.2.2 Investigating with Clusterix

We loaded the data set in to Clusterix and will show our results by walking through the analysis.

On loading our data, our initial projection was that of Figure 16. From investigating the clusters, we can first see that the clusters are not very distinct. Using the brushing functionality, and the linked feature distribution plots, we could see that the feature *total sulfur dioxide* is what was influencing the clusters most.

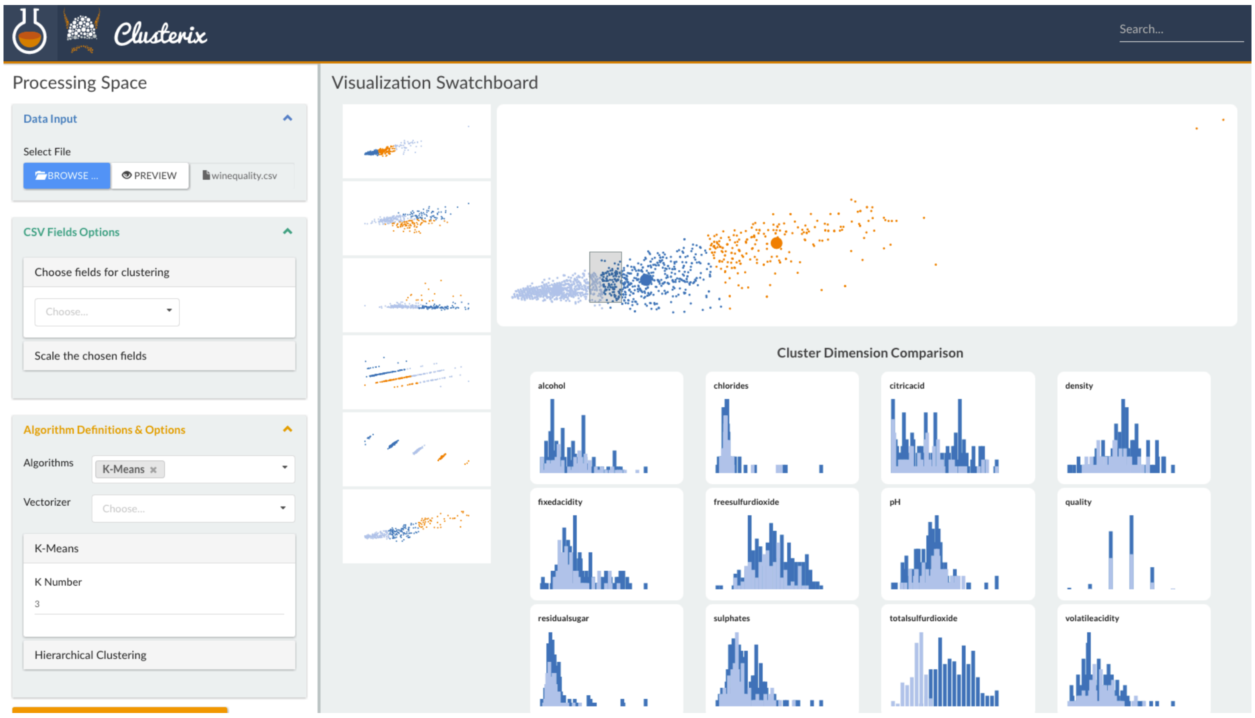
Our next step was to remove the dominant feature from consideration by the clustering algorithm which can be performed through the side panel as shown in Figure 17. We still see that there is no clear separation in the data as yet, however now the fixed acidity is the driving feature in separating the clusters.

After some more quick iterations, with a full history available, we see a better clustering in Figure 18. This new projection shows that our initial value of K is likely to be wrong since

[2]http://archive.ics.uci.edu/ml/datasets/Wine+Quality

there appears to be 6 distinct clusters now.

Clusterix enabled us to do this analysis in a matter of minutes, by quickly moving between changing model parameters and visualizing and interacting with the results.



Select File: winequality.csv

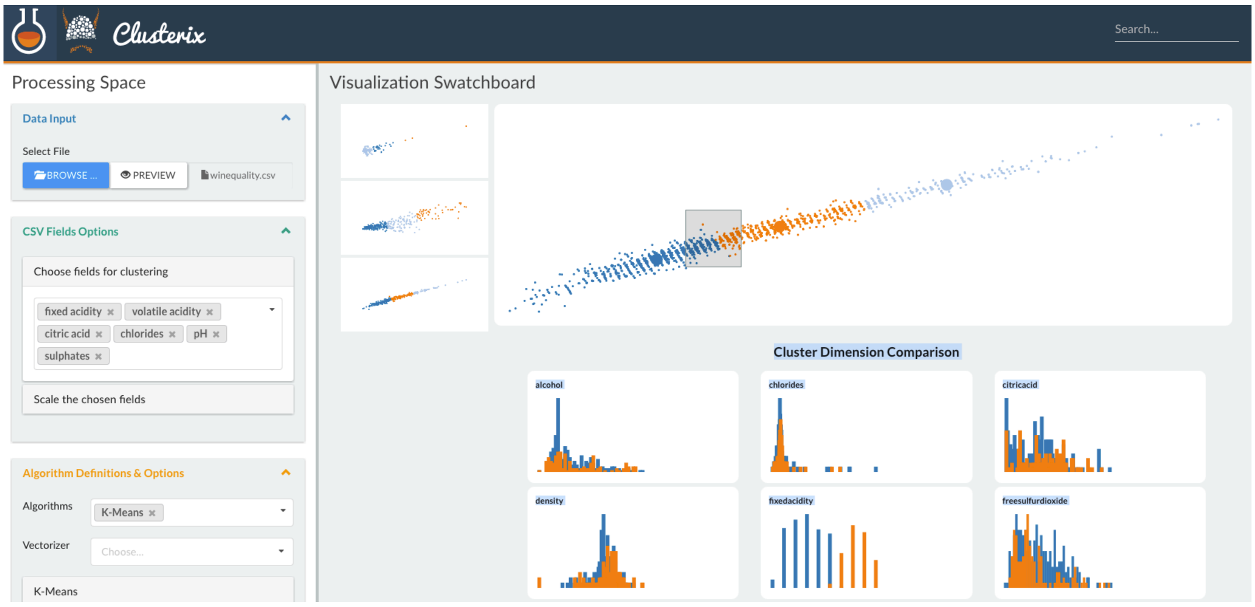
Choose fields for clustering: Choose...

Algorithms: K-Means

K-Means

K Number: 3

Figure 16: "Initial data set as loaded with K-Means and $K = 3$"



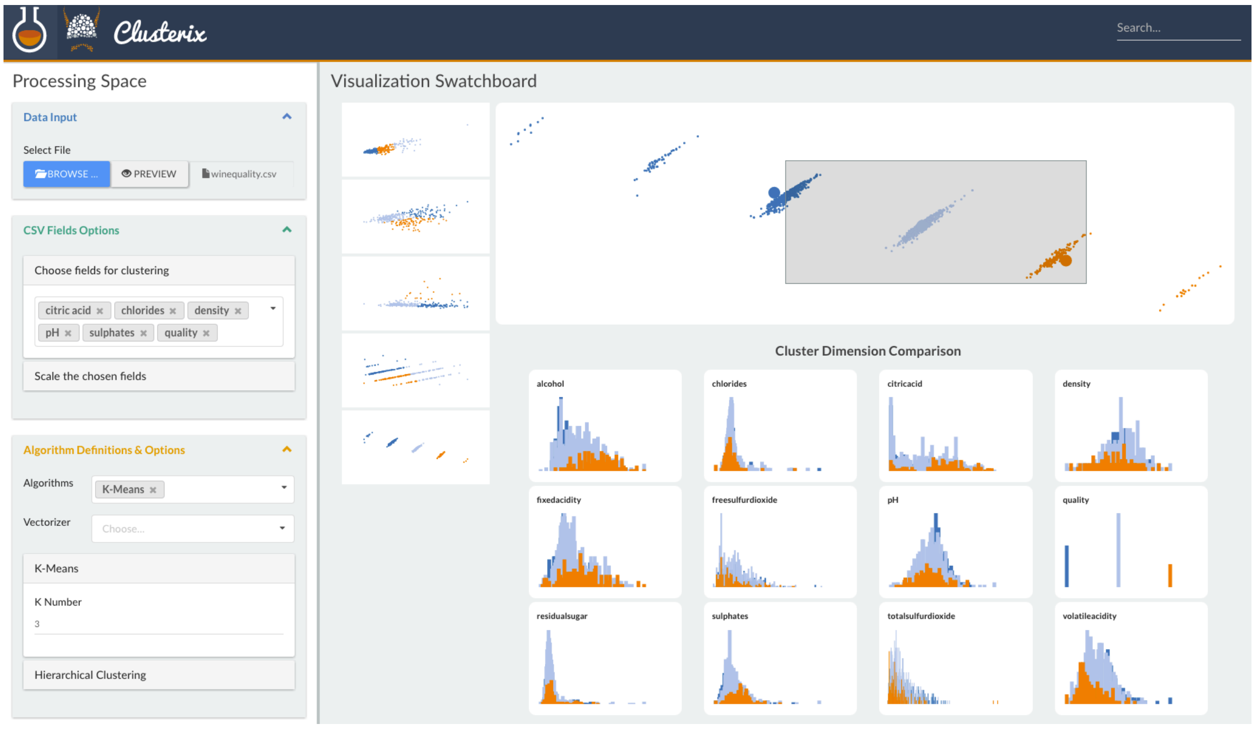
Select File: winequality.csv

Choose fields for clustering: fixed acidity, volatile acidity, citric acid, chlorides, pH, sulphates

Algorithms: K-Means

K-Means

Figure 17: "Removing sulfur changes the most important feature that now splits the data."



Select File: winequality.csv

Choose fields for clustering: citric acid, chlorides, density, pH, sulphates, quality

Scale the chosen fields

Algorithms: K-Means

Vectorizer: Choose...

K-Means

K Number: 3

Figure 18: "Removing fixed acidity and correlated fields results in a better projection of points."

6.3 Summary

Without spending too much time, and no code at all, Clusterix was able to create logically ordered clusters of data, and extract correct results with minimal work from the user.

The system provides many details that are not available in other tools, and the ability to change the clustering model, and view the results iteratively is a classic example of how the visual analytics paradigm can work to bridge human understanding and perception with machine learning tools.

7. DISCUSSION

We have presented Clusterix, a visual analytics tool to support clustering of data sets. We have shown its utility in both text-based clustering, its original purpose for INSPIREHEP, and also for more general clustering applications such as the wine quality data set. In our workflows, it has sped up the analysis process and has enabled quick iterations to find the best models to represent our data sets.

7.1 Future work

Clusterix is a prototype tool that still requires further testing and stability improvements to create a production ready tool. Important options and algorithms are already included, but we would like to give the users more freedom to interact with data, and ways to extract results from them. This is why we need to add functionality to support additional clustering algorithms, and natural language processing capabilities. More specifically, some of the short term milestones for the further improvement of the project would be:

1. **Clustering Algorithms**:
    - **HDBScan**: A new, high performance algorithm, based on the idea that not all the nodes in a data set should necessarily be assigned to a cluster, as many of the data points are noise.
    - **X-Means**: An alternative to K-Means, where the number of clusters is estimated from the algorithm, without needing the user's input.
    - **Mean Shift**: An analysis technique that is employed very often in cluster analysis, that uses density functions to produce results.
2. **Plugin Infrastructure**: Another goal is to provide an interface that would allow the user to create his own plugins. For example, knowing the general type of a data that a business will use, the user could write new functionality which will integrate with Clusterix and clean or replace data as needed.
3. **Visualizations**: We wish to provide additional visualizations and improve existing ones to make the comparison between cluster and algorithm more streamlined, as well as more approachable.
4. **Scalability**: Clusterix has two scalability concerns:
    - **Clustering performance** - Clusterix can handle moderately sized data sets, however larger data sets will require large servers for processing. A key development will be in investigating how Clusterix can interact with distributed computing architectures.
    - **Visualization** - the number of points that can be comfortably displayed in a web browser depends on a large number of factors, and technology choices. Clusterix can render visualizations using SVG and the canvas (canvas can

scale better to large numbers of data points, but with the loss of out of the box interactivity). A further choice can be the use of WebGL technologies to render large numbers of data points with high performance levels. This will be a key interest in the coming development cycle.

TERMINOLOGY TABLE

Clustering	Συσταδοποίηση
Machine Learning	Μηχανική Μάθηση
Visualization	Οπτικοποίηση
Convergence	Σύγκλιση
Centroid	Κέντρο βάρους
Stemming	Αποκοπή καταλήξεων
Vectorization	Διανυσματοποίηση
Dimensionality Reduction	Μείωση Διαστάσεων

ABBREVIATIONS, INITIALS AND ACRONYMS

CSV	Comma Seperated Values
HEP	High Energy Physics
SSE	Sum of Squared Errors
NLP	Natural Language Processing
TF	Term Frequency
IDF	Inverse Document Frequency
TF - IDF	Term Frequency times Inverse Document Frequency
VSM	Vector Space model
PCA	Principal Component Analysis
SVD	Singular Value Decomposition
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
DOM	Document Object Model
JS	JavaScript
NaN	Not a Number
API	Application Programming Interface
GROBID	GeneRation Of BIbliographic Data

REFERENCES

[1] Gomes, L. (20 October 2014), *"Machine-Learning Maestro Michael Jordan on the Delusions of Big Data and Other Huge Engineering Efforts"* [Online]. Available: http://spectrum.ieee.org/robotics/artificial-intelligence/machinelearning-maestro-michael-jordan-on-the-delusions-of-big-data-and-other-huge-engineering-efforts.

[2] Ilias Flaounas (13 May 2015), *"Data Science at Atlassian: The transition towards a data driven organization"*, [Online]. Available: http://www.slideshare.net/iliasfl/data-science-at-atlassian-the-transition-towards-a-datadriven-organisation

[3] Stanford Medicine, Biomedical Data Science Initiative [Online] Available: http://med.stanford.edu/bdsi.html

[4] CERN (2008), Worldwide LHC Computing Grid [Online] Available: http://home.cern/about/computing

[5] Pak Chung Wong and J. Thomas (2004), "Visual Analytics". in: IEEE Computer Graphics and Applications, Volume 24, Issue 5, Sept.-Oct. 2004 Page(s): 20–21

[6] Manning, C. D., Raghavan, P., & Schütze, H. (2009), An introduction to information retrieval, [Online] Available: http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[7] Tan, P. N., Steinbach, M., & Kumar, V. (2006), Introduction to data mining, [Online] Available: http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

[8] Bora, D. J., & Gupta, A. K. (2014), Effect of Different Distance Measures on the Performance of K-Means Algorithm: An Experimental Study in Matlab. International Journal of Computer Science and Information Technologies, 5(2), 2501-2506, [Online] Available: https://arxiv.org/ftp/arxiv/papers/1405/1405.7471.pdf

[9] David Arthur, Sergei Vassilvitskii. (2006), k-Means++: The advantages of careful seeding, [Online] Available: http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

[10] Aggarwal, C. C., & Zhai, C. X. (2012), Mining text data, [Online] Available: http://www.charuaggarwal.net/text-cluster.pdf

[11] Bow. (1998, September 12), A toolkit for statistical language modeling, text retrieval, classification and clustering, [Online] Available: http://www.cs.cmu.edu/~mccallum/bow

[12] Fodor, I. K. (2002), A Survey of Dimension Reduction Techniques [Online] Available: https://computation.llnl.gov/casc/sapphire/pubs/148494.pdf

[13] Lovins, J. B. (1968), Development of a Stemming Algorithm. Mechanical Translation and Computational Linguistics, 11(1-2), 22-31, [Online] Available: http://www.mt-archive.info/MT-1968-Lovins.pdf

[14] Mathworld wolfram. (n.d.), Singular Value Decomposition – from Wolfram MathWorld (July 10, 2016), [Online] Available: http://mathworld.wolfram.com/SingularValueDecomposition.html

[15] Salton G. (1983), An Introduction to Modern Information Retrieval, Mc Graw Hill

[16] Soboroff, I. (2002), IR Models:The Vector Space Mode [PowerPoint slides], [Online] Available: http://www.csee.umbc.edu/~ian/irF02/lectures/07Models-VSM.pdf

[17] Sorzano, C. O., Vargas, J., & Pascual□Montano, A. (n.d.), A survey of dimensionality reduction techniques, [Online] Available: https://arxiv.org/ftp/arxiv/papers/1403/1403.2877.pdf

[18] The Le mur Project. (n.d.), Lemur Project Home, [Online] Available: http://www.lemurproject.org/

[19] Trim, C. (2013, January 23), The Art of Tokenization [Online] Available: https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en

[20] Turney, P. D., & Pantel, P. (2010), From Frequency to Meaning: Vector Space Models of Semantics. Journal of Artificial Intelligence Research, 37, 141-188, [Online] Available: https://www.jair.org/media/2934/live-2934-4846-jair.pdf

[21] Waegel, D. (2011), The Porter Stemmer [PowerPoint slides] [Online] Available: https://www.eecis.udel.edu/~trnka/CISC889-11S/lectures/dan-porters.pdf

[22] Pearson, K. (1901), On Lines and Planes of Closest Fit to Systems of Points in Space. Philosophical

Magazine, 1901, 2, 559□572

[23] Wold, S.; Esbensen, K. & Geladi, P. (1987), Principal component analysis Chemometrics and Intelligent Laboratory Systems, 1987, 2, 37□5

[24] Dunteman, G. H. (1989), Principal Component Analysis, Sage Publications

[25] Jollife, I. T. (2002), Principal Component Analysis, Wiley

[26] Hÿvarinen, A., Karhunen, J., Oja, E. (2001), Independent Component Analysis. John Wiley & Sons, Inc

[27] Golub, G. H. & Reinsch, C. (1970), Singular value decomposition and least squares solutions Numerische Mathematik, 1970, 14, 403□420

[28] Klema, V. & Laub, A. (1980), The singular value decomposition: Its computation and some applications IEEE Trans. Automatic Control, 1980, 25, 164□176

[29] Wall, M.; Rechtsteiner, A. & Rocha, L. (2003) A practical approach to microarray data analysis Singular Value Decomposition and Principal Component Analysis Springer, 2003, 91□10

[30] Leland McInnes, John Healy, Steve Astels, hdbscan Documentation 0.8 [Online] Available: https://media.readthedocs.org/pdf/hdbscan/latest/hdbscan.pdf

[31] Dan Pelleg, Andrew moore X-Means: Extending K-Means with efficient estimation of the number of clusters, [Online] Available: https://www.cs.cmu.edu/~dpelleg/download/xmeans.pdf

[32] Cheng, Yizong (August 1995), Mean Shift, Mode Seeking, and Clustering”. IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE) 17 (8): 790–799

[33] Flask, Available: http://flask.pocoo.org/

[34] JQuery, Available: https://jquery.com/

[35] Bootstrap File Input, Available: http://plugins.krajee.com/file-input

[36] Bootstrap Available: http://getbootstrap.com/

[37] Semantic UI Available: http://semantic-ui.com/

[38] Papaparse.js Available: http://papaparse.com/

[39] JSON, Available: http://www.json.org/

[40] Pandas, Available: http://pandas.pydata.org/

[41] Whoosh, Available: https://whoosh.readthedocs.io/en/latest/

[42] Pedregosa et al, Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011 [Online] Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[43] Buitinck et al., API design for machine learning software: experiences from the scikit-learn project, 2013 Available: http://arxiv.org/abs/1309.0238

[44] Numpy, Available: http://www.numpy.org/

[45] Scipy, Available: https://www.scipy.org/

[46] d3.js, Available: https://d3js.org/

[47] GROBID (2008-2016), Available: https://github.com/kermitt2/grobid