An application of persistent homology and the graph theory to linguistics: The case of Tifinagh and Phoenician scripts

Hajar Bouazzaoui[1], Mohamed Abdou Elomary[2], My Ismail Mamouni[3]

ABSTRACT

As the origin of the Tifinagh script remains uncertain, this work aims at exploring its probable relatedness with the Phoenician script. Using tools from within topological data analysis and graph theory, the similarity between the two scripts is studied. The clustering of their letter shapes is performed based on the pairwise distances between their topological signatures. The ideas presented in this work can be extended to study the similarity between any two writing systems and as such can serve as the first step for linguists to determine the possibly related scripts before conducting further analysis.

Key words: topological data analysis, persistent homology, graph theory, writing systems, Abjad scripts, Alphabet scripts, Tifinagh script, Phoenician script.

1. Introduction

Living beings - humans and animals alike, have a need for systems of communication to ensure their survival. Humans, by their ingenuity, have developed writing systems as a conventional visual mode to represent their oral communication. While writing and talking are both tools for transmitting messages, writing has the advantage of being a reliable form of data storage that obeys the usual coding and decoding rules, which imply a shared understanding by the author and the reader of the sets of characters composing the used writing system.

Tifinagh, which is the writing system of interest in this paper, is the script adopted for Tamazight or Berber languages more broadly. Berber has been originally spoken in territories ranging from the Atlantic coast to Egypt before the arabisation of North Africa. Millions of Tifinagh inscriptions of various styles and eras tattoo the rocks of North Africa and the Sahara. A long process of cultural and identity changes begun with the emergence of Islam in the seventh century, concurrently, the linguistic map of Tifinagh (see Figure 1) retracted over the centuries until its present form, broken into islands distant from each other.

[1]Hassan I University, Department of Mathematics and Computer Science FST de Settat, IMII Laboratory. Address: Km 3, B.P.: 577 Route de Casablanca, Morocco. E-mail: h.bouazzaoui@uhp.ac.ma, ORCID: https://orcid.org/0000-0003-1860-9757

[2]Hassan I University, Department of Mathematics and Computer Science FST de Settat, IMII Laboratory. Address: Km 3, B.P.: 577 Route de Casablanca, Morocco. E-mail: elomaryabdou@gmail.com

[3]Department of Mathematics, Research Team of Mathematics, Didactic and its Applications (M@DA), CRMEF RABAT, Avenue Allal Al Fassi, Madinat Al Irfane, 10000, Rabat, Morocco. E-mail: mamouni.myismail@gmail.com
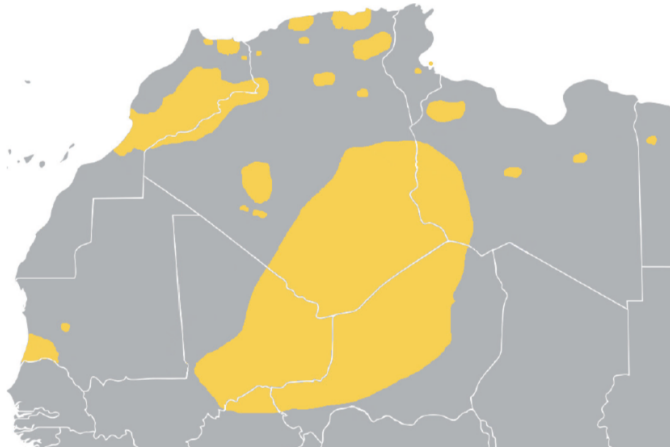
Figure 1: Current Tifinagh speaking map in Africa.

So far, there is no conclusive theory about the origin of the Tifinagh script. The majority of scholars support one these three theories (Blanco 2014):

- South-Semitic origin (Arabian and Latin scripts);

- North-Semitic origin (Phoenician and/or Punic);

- Independent invention with Phoenician influence.

Our aim in the present work is to verify whether the Tifinagh and the Phoenician scripts are indeed related.

From a linguistic point of view, the study of script evolution is not independent from historical, geographic and cultural factors. One cannot then demonstrate the relationship between scripts based solely on the study of individual graphemes (Briquel-Chatonnet 1997). However, analyzing and comparing letter shapes remains an important constituent of that study.

In order to demonstrate linguistic relatedness and to reconstruct a hypothetical common ancestral system of languages, linguists rely, among others, on the comparative method as a technique to study language development and perform comparisons on these languages (McMahon, A. and McMahon, R. 2011). However, the languages to compare are not chosen at random, and an initial stage of deciding whether some languages are related is required.

The present work, which studies the relatedness of the Phoenician and Tifinagh scripts, rely on methods that could be extended to study the relatedness of any two other scripts, and as such, serve as a first step to the comparative method, at least to the extent where only letter shapes are considered.

We believe that this is the first work that investigates the visual relationship between scripts using topological data analysis (TDA). A previous work (Sadouk et al. 2020) established a possible relationship between Phoenician and Tifinagh scripts using deep learning. The authors trained a classifier on a dataset of Phoenician letters and used a transfer learning system based on these shapes to improve the performance of Tifinagh handwritten character

recognition thereby inferring a possible relationship between the two scripts. Still, as with all deep learning systems, large samples of data were required. TDA, on the other hand, can provide robust results with only small samples of data.

To verify the relatedness of the two scripts, we adopt a topological data analysis approach based on persistent homology and graph theory. We represent each letter of the writing systems we are studying as a graph. Our aim is to study the similarity between the graphs corresponding to Phoenician letters and those corresponding to Tifinagh letters. In the literature, many graph similarity measures were studied among which we cite maximum common subgraph (Fernández and Valiente 2001), the number of mismatching edges (Zhu et al. 2012) and graph edit distance (GED) (Gouda and Hassan 2016). GED has been the most adopted one. It is the least expensive sequence of edit operations that can transform a graph G_1 to a graph G_2. In practice, however, finding the minimal edit distance is an NP-hard problem and has the drawback of having an exponential computational complexity in terms of the number of graph edit vertices.

In this work, topological information of interest in each of these graphs is summarized in persitence diagrams. Computing the *Bottleneck distance* between these topological signatures will serve as a mean to verify similarity between letter graphs and thus between Tifinagh and Phoenician scripts.

The paper is organized as follows: in Section 2, we give a brief introduction of mathematical concepts we will be using throughout this paper; we put special emphasis on persistent homology. In Section 3, we describe the method we used to perform our analysis before closing with a discussion of results and future research directions.

2. Materials and background

Homology formalizes the way topological spaces are distinguished by examining their holes. One of the most common approaches to homology is simplicial homology. It is based on associating abelian groups or modules to simplicial complexes built on top of topological spaces. One of its major advantages is that it lends itsef to relatively easy computations.

We first define what simplices are. A simplex or a *p*-simplex is the generalization of a triangle in p-dimension.

Definition 1 (*p*-simplex)
Let $e_0, e_1, ..., e_p$ be affinely independent points in $\mathbb{R}^n$. The associated convex hull, denoted $\sigma^p = [e_0, e_1, ..., e_p]$, is called a p-simplex. That is the polyhedron:

$$\sigma^p = \left\{ \sum_{i=0}^{p} t_i e_i, \ t_i \geq 0, \sum_{i=0}^{p} t_i = 1 \right\}$$
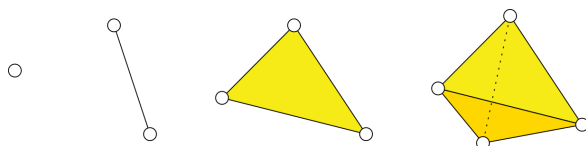
Figure 2: A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3-simplex a tetrahedron (Zhu 2013)
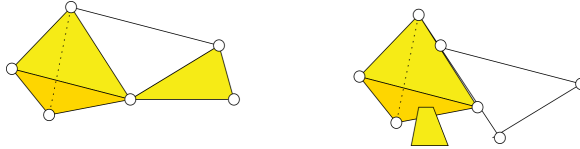
When σ and α are two simplices such that $\alpha \subset \sigma$, we call α a face of σ and σ a co-face of α.

Definition 2 (Simplicial complex)
A simplicial complex K is a finite collection of simplices satisfying the following conditions:

1. *For any $\sigma \in K$ with a face α, we have $\alpha \in K$;*

2. *If $\sigma_1, \sigma_2 \in K$ then $\sigma_1 \cap \sigma_2 =$ or $\sigma_1 \cap \sigma_2 \in K$.*

The dimension of K is the maximal dimension of its simplices.

Figure 3: Left: a simplicial complex. Right: not a simplicial complex (Zhu 2013)

Definition 3 (*p*-chain)
A p-chain is a formal finite sum $\sum_i n_i \sigma_i^p$, where σ_i^p are oriented p-simplices of a simplicial-complex K and $n_i \in \mathbb{Z}$.

The set $C_p(K)$ of all *p*-chains of *K* is a $\mathbb{Z}$-module. The following $\mathbb{Z}$-linear map :

$$\partial_p : C_p(K) \to C_{p-1}(K) \tag{1}$$

is called a *boundary map*, it is defined at the level of the generators as follows:

$$\partial_p(\sigma) := \sum_{i=0}^{p} (-1)^i [e_0, e_1, ..., \hat{e}_i, ..., e_p] \tag{2}$$

where $\sigma = [e_0, e_1, \cdots, e_p]$ is an oriented $p-$simplex and $\hat{e}_i$ means that e_i is omitted. Thus, the boundary of a tetrahedron is the alternative sum of its four triangles, the boundary of a triangle is the alternative sum of its three edges and the boundary of a line segment is the difference of its two endpoints. A direct computation shows that

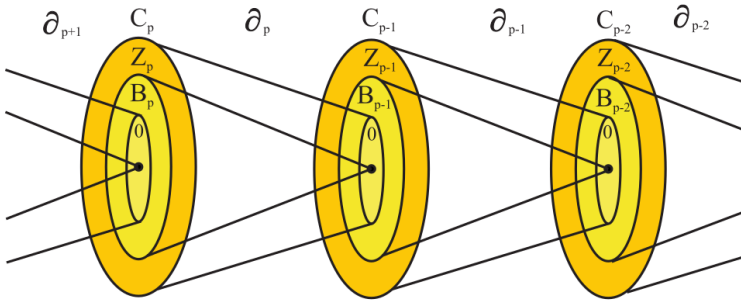$$\partial_p \circ \partial_{p+1} = 0. \tag{3}$$

In other words

$$Im \partial_{p+1} \subset \ker \partial_p. \tag{4}$$

This yields the following exact sequence, called a *chain complex* of *K* :

$$0 = C_{n+1}(K) \overset{i}{\hookrightarrow} C_n(K) \overset{\partial_n}{\longrightarrow} C_{n-1}(K) \overset{\partial_{n-1}}{\longrightarrow} \cdots \overset{\partial_1}{\longrightarrow} C_0(K) \overset{\partial_0}{\longrightarrow} C_{-1}(K) = 0 \tag{5}$$

where $\hookrightarrow$ denotes the inclusion map. The figure below illustrates the evolution of this chain complex.

Figure 4: Chain, cycle and boundary groups and their mappings under boundary operators. (Horak, Maletić and Rajković 2009)

Elements of $Z_p := \ker \partial_p$ are called *p-cycles*, those of $B_p := Im\partial_{p+1}$ are called *p*-boundaries. In particular, any *p*-boundary is a cycle, but the inverse does not always hold. The obstruction for a cycle to be a boundary is encoded in the quotient

$$H_p(K) := \frac{Z_p}{B_p}. \tag{6}$$

called the p-th homology group of *K*. Its rank, defined as

$$\beta_p(K) := \dim_Z H_p(K), \tag{7}$$

is called the *p-th Betti number* of *K* and it encodes the number of *p*−dimensional holes in the simplicial complex *K*. In particular, β_0 denotes the number of connected components of *K*. For more details, we refer the reader to these standard references (Hatcher 2002) and (Spannier 1966).

2.1. Persistent homology

Persistent homology, one of the main tools in topological data analysis, proved its usefulness in many real world applications among which shape analysis, medical imaging and network sensing are only a few examples. In many of these applications, data is given as a point cloud. Persistent homology keeps track of homology classes as a nested sequence of simplicial complexes is built on top of the data. The "lifetime" of a homology class is an indication of the relevance or irrelevance of homological information.
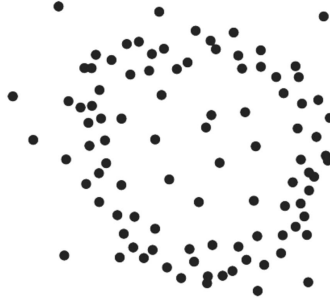
Figure 5: A noisy point cloud data

Let $\mathscr{P}$ be a cloud of points embedded in $\mathbb{R}^n$. One may associate a *filtration* to $\mathscr{P}$, that is a finite increasing sequence of sub-complexes

$$\mathscr{P} = K_0 \subset K_1 \subset \cdots \subset K_n. \tag{8}$$

For every $i \leq j$, the inclusion map $K_i \hookrightarrow K_j$ induces the homology homomorphism

$$f_p^{i,j} : H_p(K_i) \longrightarrow H_p(K_j) \tag{9}$$

at each dimension p. This yields the homology sequence

$$H_p(K_0) \longrightarrow H_p(K_1) \cdots \longrightarrow H_p(K_n). \tag{10}$$

As we go from K_{i-1} to K_i, we gain new homology classes and lose others as they become trivial or merge with each other. Persistent homology groups are defined as follows.

Definition 4 *The p-th persistent homology groups, denoted $H_p^{i,j}$, are defined to be the images $H_p^{i,j} := \mathrm{Im} f_p^{i,j}$. Their ranks $\beta_p^{i,j} := rank(H_p^{i,j})$, are the corresponding p-th persistent Betti numbers.*

We note that
$$H_p^{i,j} = Z_p(K_i)/(B_p(K_j) \cap Z_p(K_i)). \tag{11}$$

A class γ is born at time $t = i$ if $\gamma \notin H_p^{i-1,i}$. It dies at time $t = j$, when it becomes trivial or when it merges with an older class as we go from K_{j-1} to K_j, that is, $f_p^{i,j-1}(\gamma) \notin H_p^{i-1,j-1}$ but $f_p^{i,j}(\gamma) \in H_p^{i-1,j}$. The figure below illustrates this scenario.
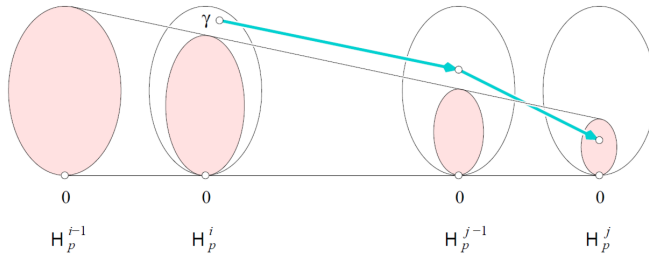
Figure 6: Example of a homology class with birth time $t = i$, and death time $t = j$. (Edelsbrunner and Harer 2010)

We can encode this evolution in a persistence barcode, which is a set of intervals whose first endpoint indicates the birth-time of the homology class, while the second one indicates its death-time. Short line segments correspond to noise, while persistent line segments imply relevant homological information.
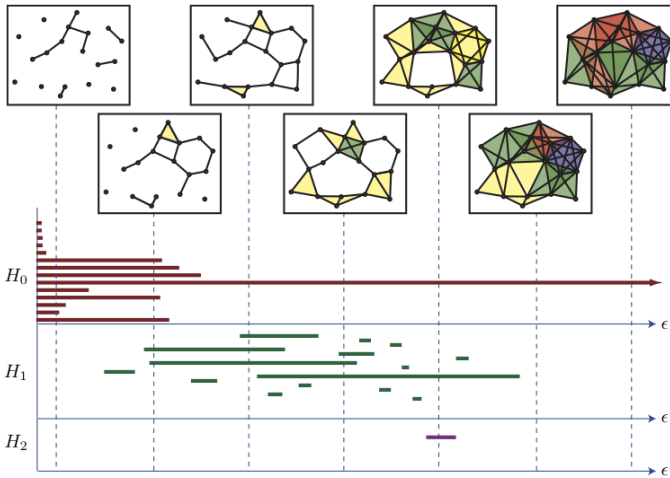
Figure 7: Example of a point cloud and its associated Vietoris-Rips complex and barcode (Ghrist 2008)

Barcodes can be computed efficiently by using a matrix reduction algorithm. Surprisingly, we can get all this information with a single reduction. We order the time appearance $t(\sigma_i)$ of a simplex σ_i as follows: $t(\sigma_i) < t(\sigma_j)$ whenever σ_i is a face of σ_j. Then we set the *boundary matrix*, ∂, which stores all that information, that is the binary matrix,

$$\partial[i, j] := \begin{cases} 1 & \text{if } \sigma_i \text{ is a face of } \sigma_j \text{ of co-dimension one ;} \\ 0 & \text{otherwise.} \end{cases}$$

Let $low(j)$ be the row index of the lowest non-null coefficient (when it exists) in the column j. A matrix is called reduced if $low(j) \neq low(k)$ whenever $j \neq k$. In other words, no two columns have lows in the same level. One way to get a reduced matrix R from the

boundary matrix ∂ is to add columns from left to right. (see Algorithm 1).

Algorithm 1 : Smith Reduction Algorithm

  Input: Boundary matrix
  Output: Reduced boundary matrix
  for $j = 1$ to n **do**
    while $\exists j' < j$ with $low(j') = low(j)$ **do**
   add column j' to column j
    end while
  end for

Theorem 1 (Pairing theorem, see (Edelsbrunner and Harer 2010))
Let R be the reduced matrix obtained from the boundary matrix. There is a persistence pairing (i, j) of a homology class whenever $i = low(j)$.

    The filtrations built on top of data can also be described topologically using persistence diagrams. These are multisets of $\mathbb{R}^2$ that encode information about homology groups. A homology class that appears at time i and disappears at time j is represented by the point of coordinates (i, j). The multiplicity of that point represents the number of features with the same birth and death times. The persistence of each class is the real value $j - i$.
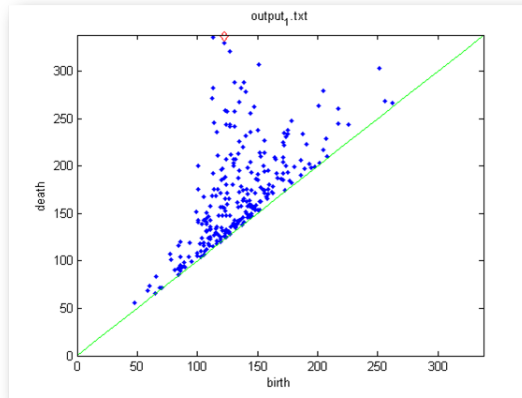
Figure 8: Example of a persistence diagram (Nanda 2017)

    In order to compare topological signatures present in the resulting persistence diagrams, we compute their Bottleneck distance.

Definition 5 *Bottleneck distance*
Given two persistence diagrams D and E, their Bottleneck distance (w_∞) is defined by:

$$w_\infty(D,E) := \inf_\eta \sup_{x \in D} \|x - \eta(x)\|_\infty$$

where η ranges over bijections between D and E.

For further details on persistent homology, we refer the reader to these standard references (Edelsbrunner and Harer 2010) and (Ghrist 2008).

2.2. Zigzag Persistent homology

A more general approach to persistent homology is zigzag persistent homology, which we will use in this paper, in this section, we introduce some of its key principles. In this setting, both forward and backward maps are permitted between topological spaces. Let $\mathbb{X}_1 \leftrightarrow \mathbb{X}_2 \leftrightarrow \cdots \leftrightarrow \mathbb{X}_n$ be a sequence of topological spaces. The maps between these spaces induce maps between chain complexes $C(\mathbb{X}_1) \leftrightarrow C(\mathbb{X}_2) \leftrightarrow \cdots \leftrightarrow C(\mathbb{X}_n)$. The homology sequence $H_p(\mathbb{X}_1) \leftrightarrow H_p(\mathbb{X}_2) \leftrightarrow \cdots \leftrightarrow H_p(\mathbb{X}_n)$ obtained by applying the homology functor H_p forms a zigzag module.

A finite-dimensional zigzag module can be decomposed as a direct sum of interval modules $\bigoplus \mathbb{I}_{[b,d]}$, where $\mathbb{I}_{[b,d]}$ is the homology class existing in the spaces from $H(\mathbb{X}_b)$ to $H(\mathbb{X}_d)$. The information needed to compute this decomposition is encoded in one filtration; the right filtration (Gunnar, De Silva and Morozov 2009).

Definition 6 ((Milosavljević, Morozov and Skraba (2011)) *The right-filtration of a space $H(\mathbb{X}_i)$ is the collection of its subspaces $R^n = (R_0, R_1, \ldots, R_n)$ such that $R_i \subseteq R_j$ whenever $i \leq j$. The filtration is defined as follows.*
*If $n = 1$, then $R^0 = (0, H(\mathbb{X}_1))$.*
If $H(\mathbb{X}_i) \xrightarrow{f} H(\mathbb{X}_{i+1})$, then $R^{n+1} = (f(R_0), f(R_1), \ldots, f(R_n), H(\mathbb{X}_{i+1}))$.
If $H(\mathbb{X}_i) \xleftarrow{g} H(\mathbb{X}_{i+1})$, then $R^{n+1} = (0, g^{-1}(R_0), g^{-1}(R_1), \ldots, g^{-1}(R_n))$.

For more details on zigzag persistent homology, we refer the interested reader to (Carlsson and De Silva 2010).

3. Method and results

Some scripts undergo a process of transformation over a long period of time, while others are a result of deliberate mixing of traits adopted from multiple other scripts. In this work, we assume that the Tifinagh and Phoenician scripts are related if, by introducing a sequence of minimal random transformations on the Phoenician letters, we obtain clusters of similar letters each containing letters from both scripts. We denote the set of Phoenician letters by $\mathbb{P}$ and that of Tifinagh letters by $\mathbb{T}$. To account for the dynamics of the script letters in $\mathbb{P}$, each of these letters is represented as a dynamic graph by allowing operations such as adding or removing vertices and edges. A dynamic graph is a graph $G = \{G_1, G_2, \ldots, G_n\}$ on which a sequence of updates is performed, G_i being the modified graph at time i. In this study,

((a)) Tifinagh letters   ((b)) Phoenician letters
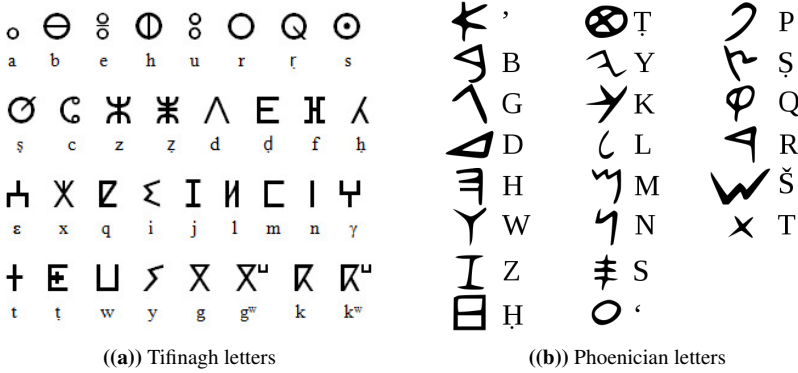
Figure 9: Tifinagh and Phoenician letters

we only allow graph transformations within 1 Graph Edit Distance (GED) from the original ones.

Zigzag persistent homology, which is a generalization of persistent homology, is more adapted for studying dynamical graphs. It allows for both inserting and deleting edges and vertices which correspond respectively to adding and removing simplices from the constructed simplicial complex.

A clustering is finally performed based on the pairwise bottleneck distances between topological signatures in order to detect similar letters in the two sets $\mathbb{P}$ and $\mathbb{T}$.

We summarize our approach in the following steps:

1. Represent each letter in $\mathbb{P}$ as a time-varying graph $G = \{G_0, G_1, \ldots, G_n\}$, G_i being the letter graph at step i.

2. Associate a metric space representation to each graph G and build a dynamical simplicial complex.

3. Compute the zigzag persistent homology of G.

4. Make a clustering of $\mathbb{P}$ and $\mathbb{T}$ graphs on the basis of the pairwise bottleneck distances between their topological signatures in order to detect similarity between letters.

3.1.　Dynamic graphs

The dynamic graphs are constructed as follows:

- The letter is represented by a finite undirected graph. The vertex set V is the set of the intersection points present in that letter.

- A vertex function f_v that maps the state of the vertex v at time t to its state at time $t+1$.

- An edge function f_e that maps the state of the edge e at time t to its state at time $t+1$.

The undirected graph is represented by an adjacency matrix $\mathscr{M}$. The dynamics correspond to adding and removing nodes and edges and updating the matrix accordingly. For an edge addition (resp. deletion) event between vertices v_i and v_j, the function f_e assigns 1 (resp. 0) to $\mathscr{M}_{i,j}$ and $\mathscr{M}_{j,i}$. The function f_v on the other hand, adds a new row and column to the matrix $\mathscr{M}$ whenever a new edge is added and deletes the row and column corresponding to a vertex when it is removed.

3.2. Metric space representation

To each graph G_i, we associate a metric space representation. A metric on G_i is obtained by computing a matrix of shortest path distances between nodes using the Floyd Warshall algorithm (Floyd 1962) which we implemented using the Networkx library (Hagberg, Swart, and S Chult 2008). The Floyd Warshall algorithm is a dynamic programming algorithm, which works in the following fashion: let d_{ij}^k be the shortest path from i to j with intermediate vertices chosen among $\{1, 2, \ldots, k\}$. Then, for $k > 1$, $d_{ij}^k = min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$.

We then build the Vietoris Rips complexes of the graphs on top of these metric spaces using Dionysus library (Morozov 2012). Given a distance matrix, we compute a sorted filtration filled with the 1-skeleton of the clique complex built on the points at distance at most six from each other, six being the maximum scale at which the Vietoris-Rips complex is computed.

3.3. Computing Zigzag Persistent Homology

In practice, given a time-varying graph $G = \{G_0, G_1, \ldots, G_n\}$, we start by constructing a simplicial complex of G_0, the graph instance at time $t = 0$. This simplicial complex is dynamically modified; as we add new vertices/edges or remove them, simplices are added or removed. At time $t = 0$, a simplicial complex K_0 is created. At time $t > 0$, K_t is the simplicial complex associated with the updated graph at time t. In the case of an addition event, a $k-$simplex is added to K_t if it was not present in K_{t-1}, while the simplices that were at time $t - 1$ and did not appear at time t are removed from the complex in the case of a deletion event. We then compute zigzag persistence of this dynamic simplicial complex using *Dionysus* (Morozov 2012).
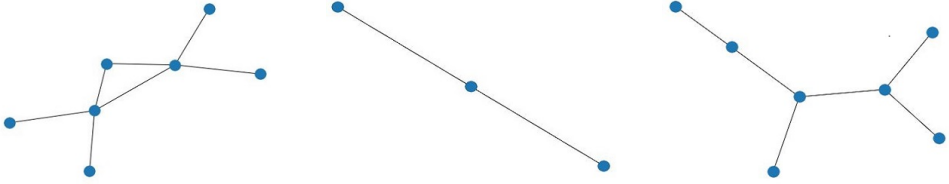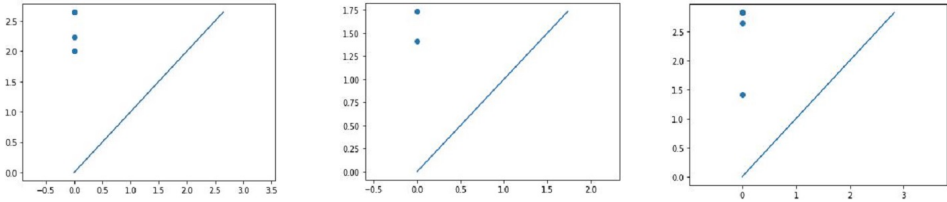
Figure 10: Sample Phoenician Letters Graphs

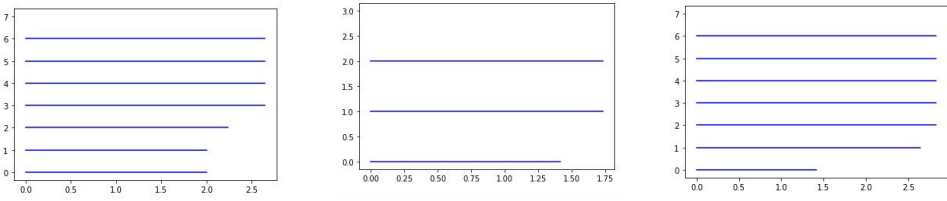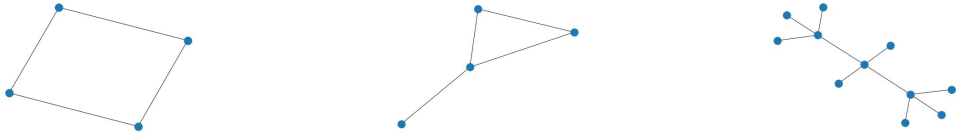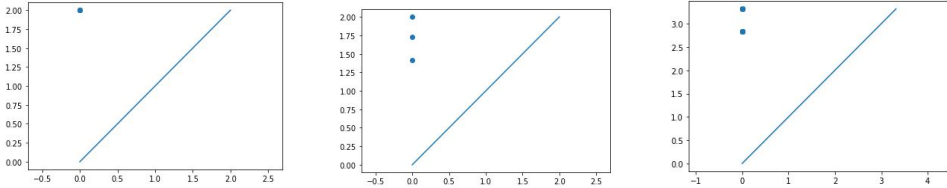**Figure 11:** Their Persistence Diagrams

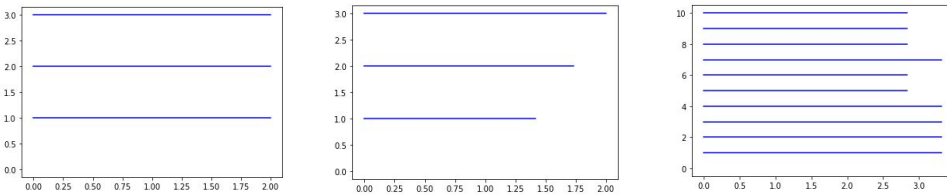**Figure 12:** Their Barcodes

Figure 13: Sample Phoenician Letters

Figure 14: Sample Tifinagh Letters Graphs

Figure 15: Their Persistence Diagrams

Figure 16: Their Barcodes

Figure 17: Sample Tifinagh Letters

3.4. Clustering

In order to verify the aforementioned claim, i.e. Tifinagh being related to Phoenician, we measure similarity between the time-varying graphs representing the $\mathbb{P}$ letters and those representing the $\mathbb{T}$ letters. After computing the persistence diagrams associated with the simplicial complexes built on top of each graph, we compute the pairwise bottleneck distance between persistence diagrams. We obtain a distance matrix on the basis of which we perform hierarchical clustering, more specifically in this case an agglomerative clustering. Agglomerative clustering starts by considering each singleton as a cluster. The clusters are then inductively combined until some stop criterion is satisfied. In this work, the update at each step is performed using a complete linkage which measures inter-cluster dissimilarity based on the maximum distances between all data points.

3.5. Results

We notice that, except for a few distinct points, each cluster in the right figure contains both Phoenician and Tifinagh letters suggesting similarity between the two.

P_n denotes Phoenician letters while T_n denotes the Tifinagh letters. The agglomerative

((a)) Dendrogram   ((b)) Clustering results
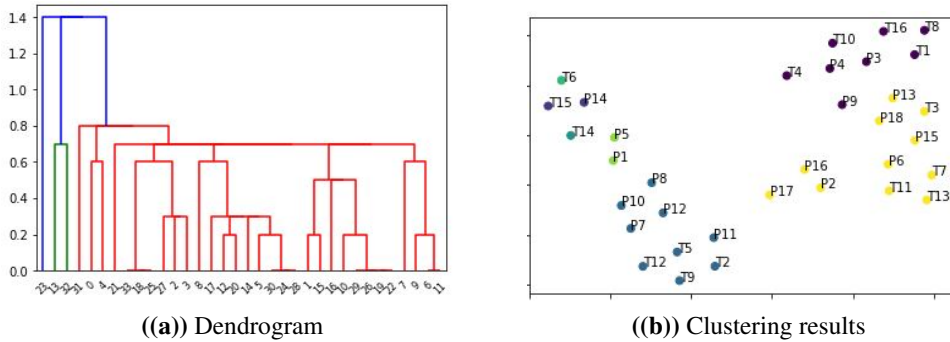
Figure 18: Dendrogram and Clustering Results

clustering with a complete linkage we used separates data into seven clusters, each containing both Tifinagh and Phoenician letters suggesting similarity between the two. A cluster also contains letters from the same script. This is due to the homogeneity present within each script; letters of the same script tend to have a common pattern that distinguishes them from other scripts. Hierarchical clustering produces a graphical representation between data points in the form of a hierarchical tree (Figure (A)) that we used for finding the optimal number of clusters.

The results we obtained only suggest a possible visual relationship between the graphemes of the script. This relationship can be due to Tifinagh being derived from Phoenician or Tifiniagh being created under the influence of Phoenician. The nature of that relationship might be a question for a future work.

4. Conclusion

In this work, we demonstrated how TDA and persistent homology in particular can be used to verify the relatedness between two writing systems. Even though we restricted our analysis to the study of similarity between the Phoenician and Tifiniagh scripts, the method we used can be extended to compare any two writing systems. A future work might explore the nature of this relatedness, i.e. whether one script is derived from the other or one was built under the other's influence.

References

Blanco, J., (2014) . Tifinagh & the the IRCAM, Explorations in Cursiveness and Bicamelarism in the Tifinagh script.*Master Of Arts In Typeface Design, University Of Reading*.

Briquel-Chatonnet, F., (1997). De l'araméen à l'arabe: quelques réflexions sur la genèse de l'écriture arabe. *Paris: Bibliothèque Nationale De France*.

McMahon, A., McMahon, R., (2011). Language Classification by Numbers. *Oxford: Oxford University Press*.

Sadouk, L. et al., (2020). Handwritten Phoenician Character Recognition and its Use to Improve Recognition of Handwritten Alphabets with Lack of Annotated Data. *International Journal Of Advanced Trends In Computer Science And Engineering*.

Fernández, M., Valiente, G, (2001). A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22, pp. 753–758.

Zhu, G., Lin, X., Zhu, K., Zhang, W., Xu Yu, J., (2012). TreeSpan: efficiently computing similarity all-matching. *Proceedings Of The 2012 ACM SIGMOD International Conference On Management Of DataMay*, pp. 529–540.

Gouda, K., Hassan, M., (2016). CSI_GED: An Efficient Approach for Graph Edit Similarity Computation. *Proc. Of ICDE'16*.

Hatcher, A., (2002). *Algebraic Topology*, Cambridge University Press.

Spanier, E., (1966). *Algebraic Topology*, McGraw-Hill Inc.

Edelsbrunner, H., Harer, J., (2010). *Computational Topology. An Introduction*, Amer. Math. Soc., Providence, Rhode Island.

Ghrist, R., (2008). Barcodes: the persistent topology of data.*Bulletin Of The American Mathematical Society*, 45, pp. 61–75.

Gunnar, C., De Silva, V., Morozov, D., (2009). Zigzag persistent homology and real-valued functions.*Proceedings Of The Twenty-fifth Annual Symposium On Computational Geometry (SCG '09)*, pp. 247–256.

Milosavljević, N., Morozov, D., Skraba, P., (2011). Zigzag persistent homology in matrix multiplication time. *Proceedings Of The Twenty-seventh Annual Symposium On Computational Geometry (SoCG '11)*. Association For Computing Machinery, New York, NY, USA, pp. 216–225.

Carlsson, G., De Silva, V., (2010). Zigzag persistence. *Found Comput Math 10*, pp. 367–405.

Floyd, R., (1962). Algorithm 97: Shortest Path. *Communications Of The ACM*, 5, p.345.

Hagberg, A., Swart, P., S Chult, D., (2008). Exploring network structure, dynamics, and function using NetworkX. *Proceedings Of The 7th Python In Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, And Jarrod Millman (Eds), Pasadena, CA USA*, 5, pp. 11–15.

Morozov, D., (2012). Dionysus. http://www.mrzv.org/software/dionysus/

Coulmas, F., (2008). Typology of Writing Systems. *Band 2*, pp. 1380–1387. Available at: `https://doi.org/10.1515/9783110147445.2.9.1380`

Gelb, I., (1963). *A Study of Writing*, Chicago University Press, 2nd edition.

Hill, A., (1967). The typology of writing systems. *Papers In Linguistics*, pp. 92–99.

Horak, D., Maletić, S., Rajković, M., (2009). Persistent homology of complex networks. *J. Stat. Mech. Theory And Experiment*, pp. 30–34.

Pulgram, E., (1976). *The typologies of writing-systems*, Mont Follick Series.

Pichler, W., (2007). The origin of the Libyco-Berber script. *Actes Du Colloque International, Le Libyco-berbére Ou Le Tifinagh: De L'authenticité À L'usage Pratique*, pp. 187–200.

Sampson, G., (1985).*Writing Systems: A Linguistic Introduction*, Hutchinson & Co. Ltd, London.

Slaouti Taklit, M. (2004). *L'alphabet Latin serait-il d'origine berbére?*, Harmattan, Paris.

Unger, J., DeFrancis, J. (1995). Logographic and Semasiographic Writing Systems: A Critique of Sampson's Classification. *Scripts And Literacy. Neuropsychology And Cognition*, 7, pp. 44–58.

Nanda, V., (2017). Perseus: The Persistent Homology Software. Available at: `http://people.maths.ox.ac.uk/nanda/perseus/`

Zhu, X., (2013). Persistent homology: An introduction and a new text representation for natural language processing. *Proceedings Of The Twenty-Third International Joint Conference On Artificial Intelligence*.