Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN
CULTURE LETTERARIE E FILOLOGICHE

CICLO XXV

**Settore Concorsuale:** 01/B1 - INFORMATICA
**Settore Scientifico Disciplinare:** INF/01 - INFORMATICA

# Computational methods applied to undeciphered scripts from the Aegean and Cyprus: Linear A and Cypro-Minoan

**Presentata da:** Michele Corazza

**Coordinatore Dottorato:**
Marco Antonio Bazzocchi

**Supervisora:**
Silvia Ferrara

**Co-supervisore:**
Fabio Tamburini

**Esame finale anno 2023**

**Abstract**

The study of ancient, undeciphered scripts presents unique challenges, that depend both on the nature of the problem and on the peculiarities of each writing system. In this thesis, I present two computational approaches that are tailored to two different tasks and writing systems. The first of these methods is aimed at the decipherment of the Linear A fraction signs, in order to discover their numerical values. This is achieved with a combination of constraint programming, ad-hoc metrics and paleographic considerations. The second main contribution of this thesis regards the creation of an unsupervised deep learning model which uses drawings of signs from ancient writing system to learn to distinguish different graphemes in the vector space. This system, which is based on techniques used in the field of computer vision, is adapted to the study of ancient writing systems by incorporating information about sequences in the model, mirroring what is often done in natural language processing. In order to develop this model, the Cypriot Greek Syllabary is used as a target, since this is a deciphered writing system. Finally, this unsupervised model is adapted to the undeciphered Cypro-Minoan and it is used to answer open questions about this script. In particular, by reconstructing multiple allographs that are not agreed upon by paleographers, it supports the idea that Cypro-Minoan is a single script and not a collection of three script like it was proposed in the literature. These results on two different tasks shows that computational methods can be applied to undeciphered scripts, despite the relatively low amount of available data, paving the way for further advancement in paleography using these methods.

# Introduction

In the study of ancient writing systems that emerged when writing was invented, there is one geographical area that is, in a sense, an anomaly. This area is located in the Aegean and on Cyprus, in the Bronze Age. In this area and time period, a multitude of writing systems that that are at the origin of writing in Europe are attested, and three of them are completely undeciphered. This thesis is concerned with the application of computational methods to Linear A and Cypro-Minoan, which are two undeciphered scripts from the Aegean and Cyprus, respectively. While they are both syllabic systems and they are widely considered to be related, the experiments that we performed over them are very different. While for Linear A we considered a very small component of the script, namely the signs used for the notation of fractional values, for Cypro-Minoan we are interested in an overarching questions regarding its inventory of signs. For this reason, the methods used to analyse these two writing systems are widely different and depend strongly on the tasks that need to be performed.

Since our experiments are performed through computational means, it is crucial to highlight which are the main advantages of these methods in the study of undeciphered scripts. One such advantage is the ability of computational methods to process large amounts of data in a very limited time. This property can be useful, for example, when evaluating the quality of a multitude of different solutions for the same problem, or when analyzing the statistical properties of a given script. Furthermore, depending on the task, it can be desirable to use methods that need very little prior information on the scripts. This approach produces results that are not based on any hypothesis on the writing system, avoiding biases that can be introduced if a single interpretation is provided to the system a priori. Finally, another crucial advantage of computational

models regards the possibility of creating visualizations and metrics that highlight the main characteristics of the outputs of these methods. This is a crucial property, as it allows a multi-disciplinary team, like the one involved in the experiments described in this thesis, to intuitively interpret results even when they are not familiar with the specific methods that are used. Through this interpretation, crucial insight can be gathered and used to improve the methods.

The usage of computational methods for undeciphered scripts, however, also leads to unique challenges, especially when methods based on machine learning are used. In recent years, the fields of natural language processing and computer vision have been dominated by machine and deep learning, in which models leverage data in order to learn the desired behaviour. The most successful models use an increasingly large number of parameters, which require large datasets in order to succeed. In fact, it can be argued that the size of the model and the number of samples used for training is one of the main factors for the success of any machine learning system. The application of these systems to undeciphered languages, then, can be challenging in this context. In particular, the available data is often scarce, meaning that the application of large state-of the-art models is not always the best possible choice for this situation. Additionally, since the scripts are undeciphered, it is often impossible to assess the quality of any solution, since the factors that determine whether the model is producing meaningful results is often unknown. For these reason, any system that aims to investigate undeciphered scripts needs to be adapted to the specificities of each writing system, as well as the task that it needs to perform.

While the application of machine learning models and computational methods to undeciphered scripts is, by itself, an interesting task from a purely technical point of view, this thesis is also concerned with the investigation of specific aspects of these writing systems. For this reason, the prior information provided to the computational methods must be consensually agreed upon in the literature, and not involve any speculative hypothesis of any kind. Additionally, the results emerging from the experiments must be supported by quantitative measures, in order to assess the quality and plausibility of the results. Throughout this thesis, statistical tests are used whenever possible, either to provide a random baseline that shows how a completely random solution compares with

our methods, or by applying statistical tests comparing two different approaches for the same problem. The rigorous evaluation of any result obtained from our models is crucial in this context, where the interest is not only in the development of novel computational methods, but also in their application to open questions in the study of ancient scripts.

This thesis, then, contains the results of a multi-disciplinary investigation in the study of undeciphered scripts, and it is concerned with both computational and paleographic challenges. For this reason, I attempted to provide a meaningful background of both fields, in order to allow scholars from both disciplines to understand the basic information needed to assess our claims on their merits. In particular, the first two chapters of the thesis are dedicated to a description of the origin of writing in the Aegean and Cyprus and an introduction of the computational methods used in later chapters, respectively. Whenever feasible, I tried to provide examples, visual or otherwise, in order to ease the comprehension of the more complex technical topics, while retaining the more formal approach to the definition of these concepts. Chapter 3, then, is focused on the state of the art and challenges that are relevant when applying computational methods to ancient scripts in general. The rest of the thesis is dedicated to the development and application of computational methods to undeciphered scripts. In Chapter 4, an approach involving constraint programming and ad-hoc metrics is used to attempt a decipherment of the fraction signs of Linear A. Then, in Chapter 5, the usage of the deciphered Cypriot Greek Syllabary in order to develop an unsupervised model for Cypro-Minoan is described. Finally, in Chapter 6, the application of this unsupervised model to Cypro-Minoan is described, as well as the reclassification that is supported by the results of these experiments.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Early Writing in the Aegean and Cyprus

The subject of this thesis is the conceptualization, development and application of computational methods to the study of undeciphered writing systems attested in the Aegean and Cyprus from the Bronze Age, using a script from Iron Age as well for a preliminary validation step. In this area we can find what are widely believed to be the oldest writing systems in Europe and, interestingly, four of them are undeciphered. In particular, in the Aegean we find Cretan Hieroglyphic, Linear A, Linear B and Phaestos disc, whose status is debated. On Cyprus, two scripts that are thought to be closely related to the ones of the Aegean are attested, namely Cypro-Minoan and the Cypriot Syllabary. While we will discuss in more detail the scripts that are part of our computational studies, it is crucial to provide an overview of all the early writing systems found in these two areas, as most experts agree that they are related, since they include common graphemes and they broadly share a system of numeric notation. Before discussing writing on these two islands, it is useful to establish a chronology. Since any dating of documents is subject to uncertainty, experts created chronological periods for both the Aegean and Cypriot civilizations, shown in Figure 1.1. Since this thesis is multidisciplinary in nature and does not deal with questions of chronology, our aim is to show, as far as it is possible, the known time periods when each of the four main writing systems of the area were attested.
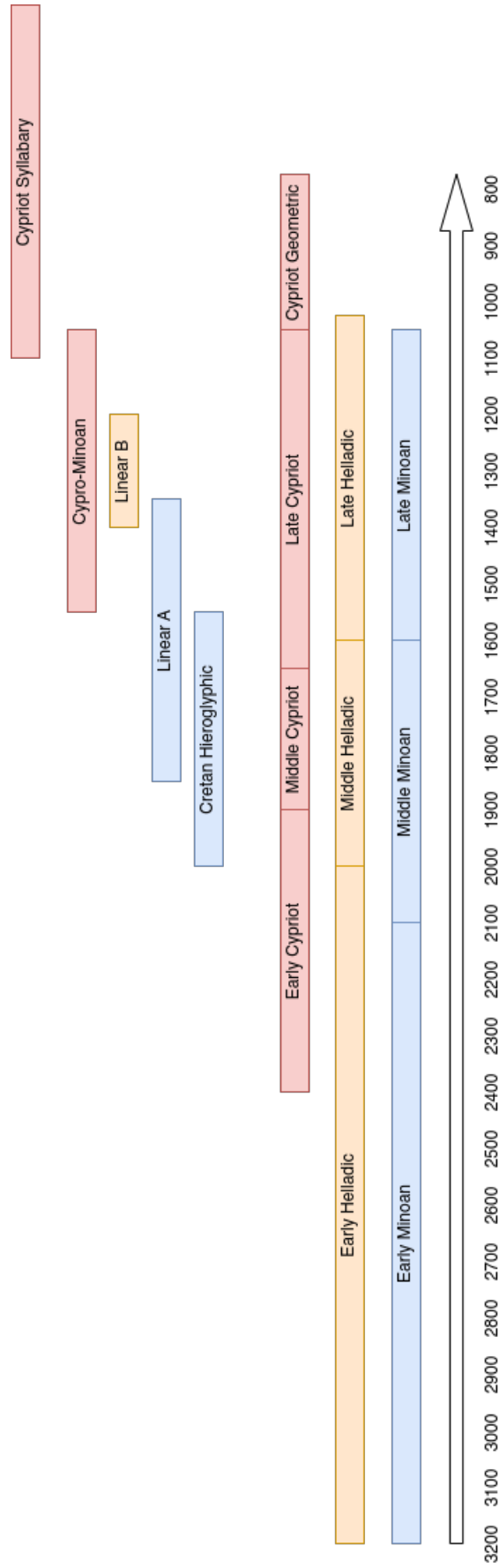
Figure 1.1: From the top, the first four rectangles show the chronologies for the CYpriot Syllabary, Cypro-Minoan, Linear B and Linear A; from the top, the bottom three rectangles show the Cypriot, Helladic and Minoan chronology based on the classification of pottery. All dates are BC. The colors are used to match writing systems with their relevant chronology.

## 1.1 Early writing in the Aegean

In this section we discuss the three principal writing systems attested in the Aegean during the Bronze age: Cretan Hieroglyphic, Linear A and Linear B. While this thesis discusses only a specific aspect of Linear A and it is instead more focused on early scripts from Cyprus, the close ties that exist between the writing systems used by the Minoan and Mycenaean civilizations with the ones attested in Cyprus merit further discussion. While another writing system is allegedly attested on the Phaestos Disc, since it is an isolated document and its authenticity is debated we chose to exclude it from our discussion, as very little can be inferred about a single document, using computational means or otherwise.

### 1.1.1 Cretan Hieroglyphic

The earliest known form of writing found in the Aegean is Cretan Hieroglyphic. The first documents in Cretan Hieroglyphic were discovered by Evans in 1900 (Evans, 1899) in Knossos palace. He described this new writing system as Hieroglyphic, as it was his belief that signs in this script had an "ideographic" quality. Furthermore, it was his belief that the signs in Cretan Hieroglyphic derived their iconic nature from other hieroglyphic writing systems like Egyptian (Evans, 1909). Most available documents were found in a limited geographical area in Crete, extending from the eastern side towards the center of the island. Very few documents were found outside the island, on Kythira and Samothrace.

Most of the Cretan Hieroglyphic corpus has been published in CHIC (Godart et al., 1996). In this corpus, the authors divide the documents in Cretan Hierogliphic in three different categories:

- Archives on clay: this group includes documents that have been hypothesized to be used for administrative purposes. This category includes nodules, "blades", bars and clay tablets.

- Seals and seal impressions: in Cretan Hierogliphic one of the methods of writing used was impression using seals. In this category, which is the most attested in

terms of number of documents, we find the results of imprinting on clay and a series of seals of various shapes.

- Other documents: in this category, the authors collected a few other documents that are not part of any other category, mostly composed of inscribed pottery.

While no precise chronology can be established for Cretan Hierogliphic, we can approximately determine that the available documents were written between 2000/1900 BCE and 1700/1650 BCE (Karnava, 2016). This means that Cretan Hierogliphic and Linear A coexisted on the island of Crete for a time, before the former was apparently abandoned as a writing system.

While Cretan Hierogliphic is undeciphered and it is unlikely that a decipherment will be possible in the near future, the structure of the writing system has been extensively studied and we have some knowledge about its internal structure. The writing system is mainly syllabic and we can establish this fact by looking at the total number of attested signs. According to CHIC, Cretan Hierogliphic is composed of 96 syllabograms and 13 logograms. Some syllabograms are also hypothesized to be used as logograms in some instances. Two systems of numeric notations are also used: one for integers (arithmograms) and one for fractions (klasmatograms). The arithmograms are particularly interesting, as they are broadly shared with not only Linear A and Linear B, but even Cypro-Minoan and the Cypriot Syllabary. It is an additive, cumulative system using base 10 (it has signs for 10,100,1000) and the signs are generally found in a descending order. This system of notation is broadly shared with Linear A, Linear B, Cypro-Minoan and the Cypriot Syllabary. Interestingly, a final category of graphemes is the stiktograms. These signs were apparently used to mark the beginning of a word, but they were not consistent, as they are only attested in some documents and not in others. This usage of punctuation to mark the beginning of a document is probably motivated by the fact that Cretan Hierogliphic is considered to use both left to right and right to left writing order.

While Cretan Hieroglyphic is not one of the writing systems that will be examined using computational approaches in this thesis, it is interesting to note that it might be possible to find graphemes in Cretan Hieroglyphic that seem to have been adopted

by Linear A. Furthermore, some sequences in Cretan Hieroglyphic are also attested in Linear A.

## 1.1.2 Linear A

Linear A is the other writing system used by the Minoan civilization. It was named by Evans who found the first documents in Linear A and Linear B in Knossos palace. In order to differentiate them from Cretan Hierogliphic documents, he called these two writing systems "linear". The standard corpus for Linear A is contained in GORILA (Olivier and Godart, 1975). The accepted chronology for the available documents of Linear A dates them approximately between 1800 and 1450 BC. The corpus of Linear A inscriptions is relatively large when compared with Cretan Hierogliphic, as it comprises of approximately 1500 documents and over 7000 individual signs. The number of syllabograms in Linear A is over 100, and, like in Cretan Hierogliphic, some of them are also used as logograms in some documents. Unlike Cretan Hierogliphic, where the number of attested logograms is relatively low, perhaps due to the limited size of the corpus, in Linear A we have hundreds of distinct logographic signs, some of which are ligatures. These logograms are used to represent different kinds of goods and the ligatures are used to differentiate between different variants of the same commodity. In addition to the syllabograms and the logograms, Linear A uses the same arithmograms used for the notation of integers in Cretan Hierogliphic. Finally, a system of notation for fractions is used, which will the target of our computational analysis on Linear A. Unlike Cretan Hierogliphic, where the writing order was arbitrary, Linear A appears to be mostly written from left to right except at the beginning of its usage. This writing order is a characteristic that is shared with most of the other writing systems from the Aegean and Cyprus. In particular, in addition to Linear A, Linear B and Cypro-Minoan are all written from left to right, while the Cypriot Greek Syllabary is written from right to left.

Despite this relative abundance of signs, Linear A is still undeciphered, probably at least in part due to the nature of its documents. The number of documents in Linear A is relatively high compared to the total number of glyphs that are attested in them, which already shows that these documents are for the most part very short. For a

comparison, the deciphered Linear B is composed of approximately 6000 documents and over 70000 signs, more double the number of signs per document when compared with Linear A. This relatively low number is detrimental to any attempt at decipherment, as very little statistical knowledge about the underlying language can be extracted from a small number of short sequences. In particular, the corpus is composed of:

- Over 300 clay tablets;

- Approximately 1000 "cretules", small clay nodules stamped with seal impressions;

- Over 100 roundels;

- A multitude of other, less attested, types of documents, such as stone pottery and nodules.

The length of sequences in these documents is usually very short and tablets, which contain the longest sequences in Linear A, are mostly used as a tool for accounting. Typically, each line is composed by one or more syllabograms, followed by a numeral of some kind and a logogram, to indicate the type of good that is notated in the line. This very structured pattern allows us to do some calculations on documents which include totals, which is denoted in a known way in Linear A. This is very helpful when trying to analyze the system for fraction notation in Linear A.

One of the more interesting aspects of this writing system is the fact that it shares most of its syllabograms with Linear B, which is a deciphered writing system and for which we know the phonetic values of the syllabograms. This means that, while the writing system is still undeciphered, in the sense that we are unable to reconstruct the underlying language, we can assign the phonetic values from Linear B to Linear A and read the content of the texts (Steele and Meißner, 2017). Unfortunately, this has not, so far, lead to a successful decipherment of the script, but it has enabled researchers to understand some repeated formulas, such as the usage of KU-RO for "total" (Schoep, 2002; Younger, 2003). While it is fairly certain that the syllabic signs from Linear A were used for Linear B as well, one might ask if their phonetic values have changed when adapting from Linear A to Linear B, as this would mean that it is impossible to use the phonetic values of Linear B for Linear A. Interestingly, one of the first computational

approaches to the study of ancient languages is found in Packard (1971) and it uses computational techniques to answer this question. It uses two factors to determine whether the same phonetic values can be used for matching syllabograms in Linear A and Linear B. The first technique has to do with the supposed vowels and their relative position in words, showing that the supposed vowels in Linear A behave as expected. Additionally, this computational approach was used to detect common words between the two writing systems, and it detected fourteen names that are attested in both Linear A and Linear B. While there are some slight variations, as sometimes the vowels in open syllables might change between the two scripts, these results can be used to justify the usage of phonetic values form Linear B syllabograms for their Linear A counterparts.

### 1.1.3 Linear B

Linear B is the third writing system attested in the Aegean, approximately from 1400 to 1200 BC. It is attested mostly on clay documents, including tablets, nodules, vases and labels. Like Linear A, the first documents in Linear B were discovered by Evans while excavating Knossos Palace. This writing system was the one used by the Mycenaean civilization and its use extended beyond the island of Crete. For this reason, documents in Linear B were found in Pylos, Thebes and Mycenae in mainland Greece, as well as Knossos and Cydonia in Crete. It is has the most extensive corpus of all the writing systems mentioned in this thesis, since it contains more than 6000 inscriptions. It is a logo-syllabic, system, composed of 87 syllabic signs and 100 logograms, which are used for commodities or objects. In addition to the syllabograms and logograms, arithmograms that are identical to those used in Linear A are also used for the notation of integers. Finally, Linear B uses logographic signs that denote units of measurements for various types of goods. This writing system is also deciphered and we know that it transcribes a dialect of ancient Greek.

The first insights into Linear B were due to Evans, who managed to discover the values of the arithmograms, which are broadly shared with the other writing systems attested in the Aegean and Cyprus. Another interesting discovery, by Kober (1945), was the presence of triplets of sequences where the first two signs never changed, while the last one often varied. Additionally, some of these triplets had a different suffix

depending whether the following logograms was masculine or feminine. This suggested that the language transcribed by Linear B was probably inflectional (Kober, 1948). This intuition was the starting point for the eventual decipherment of Linear B by Michael Ventris, an architect who studied this writing system in his free time. His intuition was that if these observed suffixes constituted a form of inflection, the ones appearing after the same signs probably shared the same consonant, with the vowel being the only part that changed. Despite his efforts, this first step was not sufficient to derive the phonetic values of any syllabogram. For the next breakthrough, he was able to use the signs from the Cypriot Greek Syllabary to derive some phonetic values for Linear B, namely the syllables "na" and "ti". Through statistical means, Ventris was able to also find the vowel "a". Armed with these few phonetic values and the hypothesis that the Cretan toponyms transcribed by Linear B were preserved in Ancient Greek, it was possible to look for these toponyms, in particular "Amnisos" (a-mi-ni-so). By finding a word with "a" as the first syllable and "ni", two other phonetic values were found, and this kickstarted a process to decipher other phonetic values, which eventually lead to the complete decipherment of this script.

While Linear B is not one of the writing systems that are the subject of our investigation, it is still useful to notice that, being the only deciphered writing system from the Aegean, it can be used as a source of comparison, especially when it comes to Linear A. The fact that we can read most syllabograms in Linear A is also due to the fact that they are shared with Linear B.

## 1.2   Cyprus

In this section, we discuss the two syllabic writing systems attested on Cyprus, from the late Bronze Age until the Iron Age. These are the subject of our main contribution: two very similar machine learning models that are able to cluster the glyphs of ancient writing systems, organizing them by grapheme. In addition to this, the application of one of these models to Cypro-Minoan allows us to investigate interesting and still debated aspects of Cypro-Minoan.

### 1.2.1 Cypro-Minoan

Cypro-Minoan is the earlier writing system attested on Cyprus, from 1550 to 1050 BC approximately. The complete corpus contains approximately 3500 signs. It is a syllabic system, with little or no evidence for logographic signs, contrary to what happens in Linear A and Linear B. It is undeciphered and so far no evidence has been found to support any hypothesis on the underlying language it transcribes. It is found on a multitude of documents, in particular:

- Clay balls: these are the most common type of document as we find 87 such balls. They contain very short sequences of signs and various hypothesis for their use have been formulated, for example the idea that they were used for some kind of game.

- Clay tablets: despite being relatively rare in the corpus, since only 8 of them were found, they contain most of the signs of the script. These are longer inscriptions and, unlike the ones found in Linear A, they do not appear to be used for book-keeping, as they do not contain as many arithmograms.

- Clay and metal vases: another relatively common type of document, including relatively short inscriptions on pottery.

- A multitude of less attested types of documents, including seals, cylinders in various materials. These are less attested and, aside from the cylinders, they usually contain a limited number of signs.

While most of the documents were found on Cyprus in various excavation sites (Enkomi, Kition, etc), some documents were found outside the island. Most of these were found on the coast of Syria, in the ancient city of Ugarit, suggesting that at the very least commercial exchanges between the island and the mainland were frequent.

Cypro-Minoan is characterized by the fact that no single proposal for the inventory of signs has been agreed upon by experts. In particular, a proposal by Masson (1974) even hypothesizes that Cypro-Minoan is not, in fact, a single writing system, but it is divided in three different groups. This division in subscripts was adopted by Olivier when he published his corpus for Cypro-Minoan (Olivier, 2007) and, while his categorization is

not the same as the one proposed by Masson, we will use his distinction in this thesis. According to Olivier, the four variants of Cypro-Minoan are:

- Archaic Cypro-Minoan or CM0: this is the earliest variant of Cypro-Minoan and it is attested on a single clay tablet from Enkomi. Among experts, the separation between this documents and the others is not debated, as it seems clear that this is an early variant of the script.

- CM1: this is the second largest group, and it is composed of all the documents attested in Cyprus which are not clay tablets. It is characterized by thinner signs and most of the documents using it are very short.

- CM2: it is the largest subgroup, composed of thee clay tablets found in Enkomi. Compared to CM1, its glyps are generally thicker and use more squarish shapes.

- CM3: the last group, which includes all the documents attested in Syria.

The epigraphic differences between these three subgroups are a consequence, according its proponents, of the existence of different languages that were transcribed using different variants of Cypro-Minoan. This situation, where there is no consensus whether Cypro-Minoan is in fact a single script or a group of very closely related writing systems has caused the scholars to hypothesize different sign inventories, depending on the position the different authors take in this debate. This means that we cannot be certain of the exact number of graphemes present in Cypro-Minoan.

The two corpora for Cypro-Minoan were compiled by Olivier (2007) and Ferrara (2013a) and they were used extensively in our experiments. Interestingly, the two authors have differing opinions on the tripartite division of the script. What is less doubtful is the idea that Cypro-Minoan is somehow related to the writing systems from the Aegean. The main hypothesis is that this script is probably derived from Linear A, with some parallels between signs being noted in various works (Faucounau, 1977; Ferrara, 2012; Steele, 2013; Valério, 2016; Valério and Davis, 2017).

### 1.2.2 Cypriot Syllabary

The last script that we discuss in this section is the Cypriot Syllabary, attested on Cyprus between the 11th and 4th century BC, during the Iron Age. It is deciphered and we know that most of the documents written in the Cypriot Syllabary transcribe the Arcado-cypriot dialect of Ancient Greek. However, another language is transcribed using the same script, called Eteocypriot. This language is still unknown, since very little can be derived from a small number of inscriptions, despite the fact that the phonetic values for the Cypriot Syllabary are known. In this thesis, we will refer to the Cypriot Syllabary texts transcribing Greek as the Cypriot Greek Syllabary, in order to avoid any confusion with the Eteocypriot documents. In addition to the two different languages, there are also two variants of the script, characterized by different shapes for some signs. These are commonly called "Common" and "Paphian"and they are found in different geographic locations.

Like Cypro-Minoan, it is a syllabic system and, when compared with Linear A and Linear B, it has dropped the usage of logograms. Like the other writing systems mentioned in this thesis, it also uses the same system of arithmograms for numeric notation that is broadly shared among all the writing systems from the Aegean and Cyprus. The syllabary itself is composed of 56 signs. It has been attested on a relatively large number of inscriptions, approximately 1400 (Perna, 2015).

Interestingly, the Cypriot Greek Syllabary was the first writing system from Cyprus and the Aegean to be deciphered. It was already deciphered in the 19th century by George Smith (Smith, 1872), thanks to a bilingual document called the "Idalion biliingual". As discussed previously, the decipherment of the Cypriot Greek Syllabary was instrumental for the initial hypothesis used by Ventris when deciphering Linear B.

The script is considered to be closely related with Cypro-Minoan and some work has been done (Valério, 2016) on using the Cypriot Greek Syllabary and Linear A phonetic values to derive possible readings, while also checking for toponyms and names using these readings. While the script has already been deciphered, it can be instrumental for any investigation in Cypro-Minoan and it can also be used, as we will discuss later, to develop computational systems that can then be adapted to the undeciphered script.

# Chapter 2

# Computational methods for paleography

The main goal of this thesis is the development and application of computational methods to ancient, undeciphered writing systems. When dealing with this type of data from undeciphered scripts, multiple technologies can be used to achieve progress in the field. The main deciding factor for the choice of technology regards the precise goals that the system should achieve, as well as any prior knowledge about the specific writing system. In this thesis, we will describe two different approaches, that are aimed at two very different types of scientific questions and data. First, a more rigorous approach, based con constraint programming, that was applied to attempt a decipherment for a numerical notation. Systems of numerical notation are generally more constrained and presents more regularities than general purpose writing systems, so constraint programming can be a useful tool to investigate their values. Then, I will briefly describe how machine learning models work in the context of natural language processing and computer vision. These machine learning approaches are better suited to deal with the complexities and irregularities that emerge when investigating the signs of a writing system and in natural language. These are the methods that are the main focus of the thesis, as a combination of computer vision and natural language processing are used in order to investigate the undeciphered Cypro-Minoan.

## 2.1  Constraint programming

Constraint programming is a paradigm that, instead of explicitly defining the steps involved to produce a solution for a problem, expresses the problem in terms of variables with specified domains, and constraints involving variables. Formally, a constraint program is composed of

$$X = \{x_1, \ldots, x_n\} \tag{2.1}$$

$$D = \{d_1, \ldots, d_n\} \tag{2.2}$$

$$C = \{c_1, \ldots, c_m\} \tag{2.3}$$

$$f(x_1, \ldots, x_n) \tag{2.4}$$

Where:

- $X$ is the set of variables that we want the program to assign;

- $D$ is the set of domains for each variables. This means that $x_i \in d_i \quad \forall i$.

- $C$ is a set of constraints. Each constraint might involve only one or more variables.

- $f$ is an optional cost function involving one or more of the variables, which determines how good any assignment for the variables is. Lower values correspond to higher quality solutions while higher values have lower quality.

The goal is then to find one or more possible assignments for all variables of the form $x_i = v_i$, where $v_i \in d_i$, that respect all constraints $C$ and, optionally, have the lowest cost function $f(X)$. For example, if we want to solve the following constraints for variables $a, b, c$:

$$a \in \{1, 3\} \tag{2.5}$$

$$b \in \{2, 7\} \tag{2.6}$$

$$c \in \{2, 5\} \tag{2.7}$$

$$a < b \tag{2.8}$$

$$b \leq c \tag{2.9}$$

We obtain a single possible assignment for the variables of $a = 1, b = 2, c = 5$. While this is a very simple problem with a single solution that can be derived by hand, most situations involve under-constrained programs, where more than one solution is possible given the constraints. The number of variables, domain size and number of constraints for variables are also usually higher than what we show in the example, resulting in non trivial programs.

The most important component of a constraint programming system is its solver, which can support different kinds of variables (eg integers or real numbers) as well as different strategies for finding solutions. The simplest case involves only linear relationships and it is called linear programming. In this case, ad-hoc approaches such as the simplex algorithm can be applied in order to efficiently find the solution of the linear system of equation. Another class of programs, called integer linear programming, is involved with solving programs that are composed of linear relations between integer variables and can also be solved efficiently.

While for some classes of problems the existing algorithms are computationally efficient and they allow us to tackle large programs involving many variables and constraints, more complex situations often require an extensive exploration of the solution space, resulting in long, often intractable, computations. This is the case, for example, when the constraints are non linear and involve real variables. One possible approach in this case is to test assignments for variables, while reducing the domain of other variables accordingly, what's called constraint propagation. This allows the solver to consider only some assignments for the variables, greatly reducing the search space for a solution. In these situations, it is crucial to limit the size of the domain for the variables as much as possible and to produce a stringent set of constraints, in order to reduce the computational complexity of the problems.

## 2.2   Machine learning

In recent years, many successful attempts to solve the most challenging natural language processing and computer vision tasks are based on Machine Learning approaches, leveraging in particular deep learning models. In order to discuss the merits and limits

of these techniques, we first need to describe what separates these techniques from the more conventional computational models and describe the recent advancements in the field.

We define as machine learning any approach where, instead of explicitly implementing a series of steps (an algorithm) to solve a task, we learn a solution from the data itself. While this approach is not necessary for most tasks, it is very advantageous when we are dealing with data that is either human generated and unstructured or, more generally, raw data from the real world. The main reason for this advantage is the fact that encoding in rules some characteristic stemming from real word or human-generated data is often impossible. If, hypothetically, we wanted to detect a cat in a photograph, coming up with rules that define it and distinguish it from other animals would be almost impossible. What are the main characteristics of a cat, that distinguish it on an image? Even if we can describe them in words, how do these relate to pixels in an image? For this reason, it is preferable to use an approach where our system learns from many images, so it can generalize what a picture containing a cat looks like.

In general, machine learning systems use a peculiar pipeline in order to learn from data: first, we show our model some training data (training set), so it can learn from it. Then, we typically need to assess the performance of the system by applying our machine learning model to other data (test set), separated from what we used during training. We use this separation between training set and test set to assess whether the model can generalize on new data or whether it just memorized the training set without obtaining more general knowledge about the task.

In order to describe a machine learning system, we need to describe two main characteristics: the model that we use and the training procedure. The model is the software component that needs to learn from data, and it is a very important aspect of any machine learning and deep learning system. With respect to the training procedure, we can distinguish three main types of learning:

- **Supervised learning**: the model learns from data that contains annotations for the specific task that we are interested in. For example, if we want to detect cats in pictures, we need to provide the model many images and information about whether each image contains a cat or not.

- **Unsupervised learning:** in unsupervised learning, we do not provide any information to the model that is not contained in the data itself. A prominent example is a clustering, where we provide the model with the raw data and teach it to categorize the data in groups called clusters. Crucially, only the intrinsic characteristics of the raw data are used to perform this task.

- **Reinforcement learning:** in reinforcement learning, the goal of our model is to learn to explore and interact with an environment. In order to teach this kind of behaviour, we provide the model with rewards for exploration and rewards that are tied to the final goal of the system. A typical example of these kinds of models is a model that needs to traverse a maze from entry to exit. In this case, we might reward the model whenever it explores new areas of the maze, and finally reward the completion of the maze.

### 2.2.1 Loss functions for different tasks

We described how machine learning avoids defining a precise algorithm to solve problems. However, in order for the concept of learning to be meaningful, we need to formally define an objective for our models. This object comes in the form of a loss function to minimize: when we apply our model to the training set we want to minimize a value obtained from the data and gold standard values that we want to predict. In order to simplify things, we will only consider the supervised case, as it is the simplest way to train a model. Perhaps the simplest possible choice of loss function is the mean squared error:

$$MSE(X, Y) = \frac{1}{N} \sum_{i=1}^{n} (M(x_i) - y_i)^2 \tag{2.10}$$

Where:

- $M$ is our machine learning model;

- $x_i \in X$ are all the data samples in the training set;

- $y_i \in Y$ are all the annotations associated with each sample $x_i$;

- $N$ is the total number of samples present in the training set.

This loss function measures the mean of the squared errors between what the model predicts and the correct values. This simple function can be applied when the output we desire from of our machine learning model is one or multiple real valued number. This kind of task is called a **regression** task. One example might be a model that predicts the temperature of the air from the meteorological data of the previous days. However, this is not the only possible task for a machine learning model.

Instead of real values, we might want to predict categories or even binary values. Our model is then defined to output one or more binary values and we call this a **classification** task. In this case, the most common choice is the usage of cross-entropy based loss functions. The cross-entropy is defined between two probability distributions $p$ and $q$ over the same set, where $p$ is the real distribution of the set, while $q$ is an approximation. The cross-entropy between the two sets is then defined as the average bits (or nats, depending on the basis of the logarithm used in the calculation) that are needed to encode data coming from distribution $p$ with a coding scheme optimized for $q$ instead. Intuitively, this measures the difference between the two probability distributions. By minimizing the cross-entropy, we attempt to create a distribution $q$ that is able to approximate $p$ as much as possible. In the case of multiple categories, we define the categorical cross entropy loss as follows:

$$L(X,Y) = -\sum_{i=1}^{n}(y_i \log M(x_i)) \tag{2.11}$$

Where: $y_i$ and $M(x_i)$ are vectors of values between 0 and 1. While many other alternatives are possible, the two loss functions defined in equations 2.11 and 2.10 are sufficient to create the two most common types of machine learning models, classifiers and regressors.

## 2.2.2   Machine learning models

We briefly mentioned that the model is a crucial aspect of any machine learning system, but we still did not define what the role of a machine learning model is. The model is the object of any learning procedure: it contains parameters or weights that are updated during the learning procedure that are used to approximate the function that
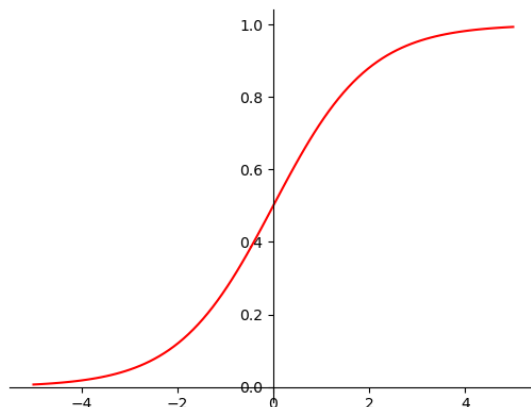
Figure 2.1: Sigmoid function

we want to compute. Formally we can define a step of the training procedure as follows:

$$W_{n+1} = W_n + \Delta W_n \tag{2.12}$$

Where $W_n$ represents the weights of our model at step $n$. Our model $M$ is a function in $\mathbb{R}_x \to \mathbb{R}_y$, where x and y can be integers or tuples of integers. Finally, $\Delta W_n$ is an update that is applied to our model at step $n$. The $\Delta W_n$ term is usually chosen to minimize the loss function, which guides the model to approximate the expected behaviour. In the case of a supervised function it is typically defined in terms of the input data and its respective labels. From this very general framework, many different models can be defined. Perhaps the simplest version is the linear regression, which can be defined as:

$$M(x) = Wx + b \tag{2.13}$$

Where $W$ is a matrix of weights and $b$ is a vector of biases. This linear regression function has some interesting properties, since it consists only of a linear projection of the input data. When combined with mean squared error as a loss function, this function is convex, so there is always a single, optimal solution where the loss reaches its minimum value and all around it the loss function is increasing.

In order to adapt the linear regression model to a simple binary classification task, we can force its output vector to contain only values between 0 and 1 and replace its loss

function with the binary cross-entropy. This can be obtained by applying what's called a sigmoid function:

$$M(x) = \sigma(Wx + b) \quad \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.14}$$

If we observe the plot for the sigmoid function $\sigma$ (Figure 2.1), we can clearly see that the function produces values that tend to 1 as the input x increases, and values that tend to 0 when the values decrease. Formally:

$$\lim_{x \to \infty} \sigma(x) = 1 \quad \lim_{x \to -\infty} \sigma(x) = 0 \tag{2.15}$$

This means that the output values of a logistic regression classifier are always between 0 and 1, so we can treat them as binary values. In particular, we round any output greater or equal to 0.5 to 1 and any value lower than 0.5 to 0.

While this regression model (and its corresponding classifier, the logistic regression classifier) are certainly useful in many fields, they cannot approximate all functions. To illustrate why linear models cannot approximate all functions, let's observe how a
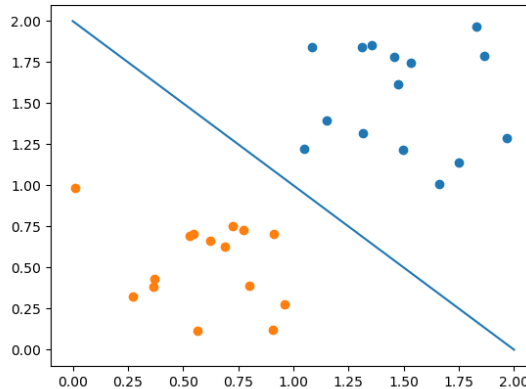


Figure 2.2: Logistic regression classifier applied to linearly separable data

logistic regression classifier behaves in two dimensions. In Figure 2.2, we can see how a logistic regression classifier separates two classes (in blue and orange in the figure) in two dimensions: it uses a line to separate two classes. We say then that the data is linearly separable. While this approach works on the toy dataset we created, however, classes
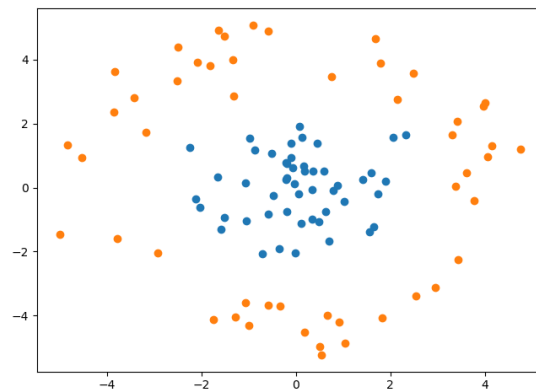
Figure 2.3: Data that is not linearly separable
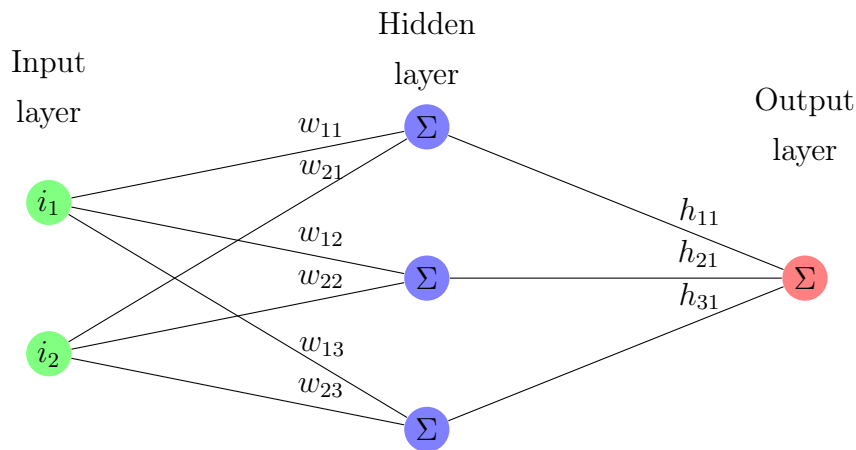


Figure 2.4: A small neural network with one hidden layer

that are distributed in more complex shapes cannot be separated by a single line. For example, we show in Figure 2.3 that a toy dataset composed of two classes found in a circle and in a ring around it respectively cannot be separated by a line. Therefore the two classes are not linearly separable.

For more complex data, which usually is not linearly separable, we need a more powerful model in order to approximate complex functions over it. In particular, we can use non linear models, which introduce a non linear function in the model definition. The typical example of these models is an artificial neural network. Formally, we define

a neural network with one hidden layer as:

$$M(X) = f_2(W^{(2)}H(X) + B_2) \tag{2.16}$$

$$H(X) = f_1(W^{(1)}X + B^{(1)}) \tag{2.17}$$

While the algebraic notation used above in equation 2.17 is most commonly used, a more intuitive visualization of the behaviour of a neural network can be found in Figure 2.4. If we observe each neuron of the hidden layer in blue, we can see that all the four input values contribute to its value, in addition to separate weights for each input and hidden neuron, denoted as $w_{ij}$. Similarly, the final layer of the network uses input values from all the hidden layer neurons. In the two lines of the equation, we can clearly see the usage of two different sets of weights in order to compute the final results. In particular, two functions $f_1, f_2$ are applied to the outputs of each layer. While $f_2$ can be the same sigmoid function we apply to logistic regression classifiers in order to obtain outputs between 0 and 1, the definition of $f_1$ is more interesting. In particular, $f_1$ is usually defined as a different non linear function. Possible examples are the rectified linear unit (ReLU), hyperbolic tangent and sigmoid. By stacking two nonlinear layers we can then approximate more functions, since it has been theoretically proven that both neural networks with an arbitrarily large single hidden layer (Cybenko, 1989; Hornik, 1991) and neural networks with arbitrarily many hidden layers (Lu et al., 2017) are what we call universal functions approximators. This means that they are able to approximate all continuous functions, infinitely more than simple linear models. This is the theoretical foundation for the improved results that we obtain by replacing a linear model or a logistic regression classifier with a neural network, provided that enough data is available. An intuitive explanation of the increased expressive power of neural networks when compared to linear models stems from the idea that each layer in a neural network learns more and more sophisticated representations of the input, allowing the model to gradually "abstract" the data in order to perform complex tasks.

### 2.2.3   Gradient Descent

While they are more expressive than linear models, the usage of neural networks also has some drawbacks. In particular, if we plot the loss function we use during training,
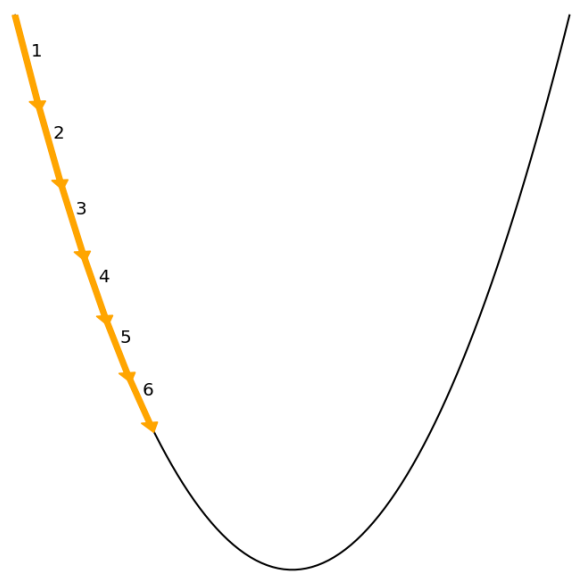
Figure 2.5: An example of gradient descent in two dimensions: the slope of the curve is used to move the weights towards the minimum of the curve. The numbers are used to show the iterative applications of gradient descent.

we will see that the solution space is not convex anymore, and that we have many local minima for a given training set and loss function. While the weight values of a linear regression model can be determined algebraically, such a strategy is impossible to apply to a neural network. Therefore, another technique is necessary. This approach, the backpropagation algorithm, is based on a concept called gradient descent. Gradient descent is the process where, by following the slope (gradient) of the function created by applying the loss function to the outputs of the model, we alter the weights of the model in order to reduce the loss. We can show the process in two dimensions by using Figure 2.5. Formally, we have a neural network $M$ that is composed of weights $W = W^{(i)}$ for each layer $i \in 1 \ldots N$ and a training set $X = (x_i, y_i)$. The weights of the neural network are initialized with random values. We can then apply the network to the training set to obtain the predicted labels $M(X)$. We can then compute the loss of the network $L(M(X), Y)$. This allows us to compute the gradients of the loss function with respect

to the all the weights in the network:

$$\sum_i \frac{\delta L}{\delta W}(x_i) \tag{2.18}$$

In the case of a two layer network like the one in equation 2.17 we will compute two distinct gradients:

$$\sum_i \frac{\delta L}{\delta W^{(1)}}(x_i) \quad \sum_i \frac{\delta L}{\delta W^{(2)}}(x_i) \tag{2.19}$$

This step is the reason for the backpropagation name, as the gradients of the neural network layers are computed by applying the chain rule backwards (from the output layer to the input layer). Finally, we can use the gradients to update the weights of each layer in the network:

$$W^{(1)} = W^{(1)} - \gamma \sum_i \frac{\delta L}{\delta W^{(1)}}(x_i) \tag{2.20}$$

$$W^{(2)} = W^{(2)} - \gamma \sum_i \frac{\delta L}{\delta W^{(2)}}(x_i) \tag{2.21}$$

$$\tag{2.22}$$

Where $\gamma$ is a value used to control the size of the gradient descent step. Typically, the update step is applied multiple times, until a stopping criterion is reached. The approach of the previous equations is called batch gradient descent, as it uses the whole dataset at once before updating the weights of the neural network. However, the most common approach is to use mini-batches, smaller portions of the training set, to update the weights of the network. This approach is applied over the entire dataset multiple times and the data is shuffled after each iteration or epoch. This procedure is called mini-batch gradient descent and it is generally more computationally efficient, as considering only smaller subsets of the training set allows the usage of GPUs[1] during training, resulting in a faster training time.

   Another interesting aspect of neural network training arises from the non convexity of their loss functions. This of course complicates the training procedure, as the algorithm

---

[1]Graphics Processing Units: they are commonly used to train neural networks as they can perform many algebraic operations in parallel.
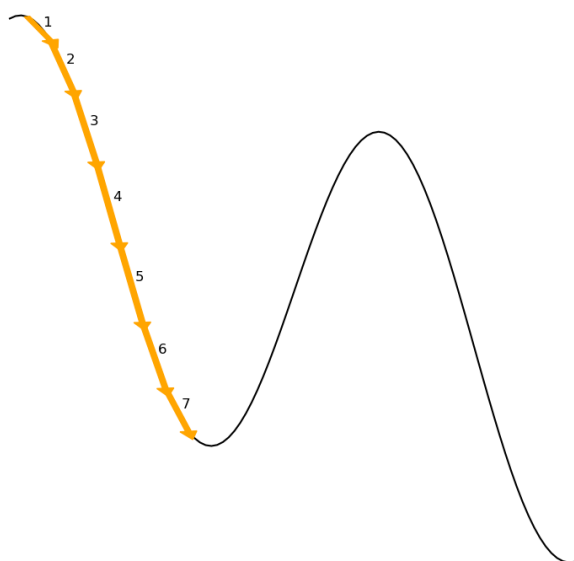
Figure 2.6: In this simple example, the training procedure never finds the global minimum on the right, instead getting stuck in the local minimum on the left. The numbers are used to show the iterative applications of gradient descent.

can get stuck in many local minima and never reach the global one (See Figure 2.6). This situation is another reason why mini-batch gradient descent is preferable to batch gradient descent, as the difference between mini batches in the training set will help the training procedure from getting stuck in shallow local minima, since the mini batches used during training are shuffled each time the whole training set has been used (at each epoch). The probability of getting stuck in a local minimum also decreases when the number of parameters of the network is really high.

### 2.2.4   Generalization power and neural networks

A final consideration about neural network training regards their ability to memorize data. In particular, it is possible for a model to memorize the training data too well (overfitting), which results in a less general model that is unable to deal with unseen data as well as it does with data from the training set. This behaviour is shown in a visual way in Figure 2.7.

First, we can notice that in this example, some orange points are found in the area
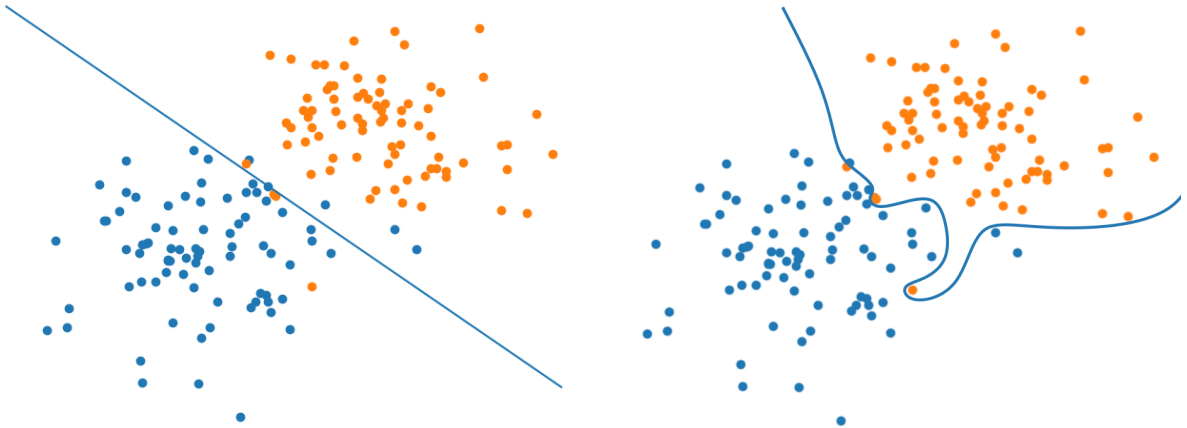
Figure 2.7: A logistic regression model not overfitting the data on the left, and a more complex model overfitting on the right.

where most blue ones are, and vice versa. These points are in fact statistical anomalies, as we can see that they are different from the more general distinction between the two classes of the dataset. These anomalous points have no effect on the logistic regression classifier shown on the left, as using a simple line to separate the points does not allow the model to be influenced by a few anomalous points. When we apply a more complex model, exemplified by the right image, we can see that it learns to distinguish every single point correctly. While this means that the model is perfectly accurate on the training set data, this property is in fact undesirable, as new points that are not in the training data will not follow the arbitrary distinction that was learned by this more complex model. It would be better, instead, to rely on a more simple distinction such as the one learned by the logistic regression classifier, as that is more likely to classify correctly new data.

Since overfitting can be problematic for neural networks, it is necessary to evaluate the models ability to generalize. Therefore usually experts separate the available data in three groups: a training set, a validation set and a test set. While the training set is used to learn the parameters of the model; the validation set is used during training to evaluate whether the model can generalize on new, unseen data; finally, the test set is used to evaluate how the model performs. Since we use a validation set, we can also stop training once we see that the loss on the training set is still decreasing, while the one on the validation set is not, as this means that we are starting to memorize the training set

instead of learning a more general model. This approach is called early stopping. More robust procedures are also used, such as K-fold cross validation.

While overfitting is often an issue in the context of supervised learning, it might not be possible to evaluate how well the model generalizes in an unsupervised setting. Additionally, it might even be useful to force the model to overfit when no other in domain data is available at all and the model is unsupervised. We will discuss this situation, as forcing a form of overfitting on data proved useful when evaluating unsupervised neural models for signs.

## 2.2.5   Deep Learning for images

While in principle any feed forward neural network model can be used for images, there is a crucial property of images that makes the application of a feed forward neural network such as the one from equation 2.17 problematic. In particular, if our goal is to build a system that is able to decide whether a photograph contains a cat or not we can immediately recognize that the position the cat occupies in the image is irrelevant to perform this task. With a feed forward neural network, however, each pixel is considered as a separate entity and it has its own weights in the model. This means that the whole model is not translation invariant, so the position of a pixel in the image matters and the whole image cannot be translated in space without changing how the model will interpret it. For this reason, the first step towards powerful computer vision tools was the creation of convolutional neural networks (LeCun et al., 1989): while feed forward neural networks have separate weights for each pixel, these models use a kernel. The kernel is a matrix of weights that is applied to pixels in the image one by one and detects patterns in their neighborhood, like horizontal and vertical edges. This property allows the model to consider smaller areas of the image separately, by applying the same kernel to them (see Figure 2.8) and detect features no matter where they are located in the image.

While in the example of Figure 2.8 we used a single matrix as the input to the convolutional model, in practice most color images have three dimensions, for red, green and blue. In addition to this, multiple kernels are applied to the same image, in order to capture multiple features. In addition to that, most models use a pooling layer after
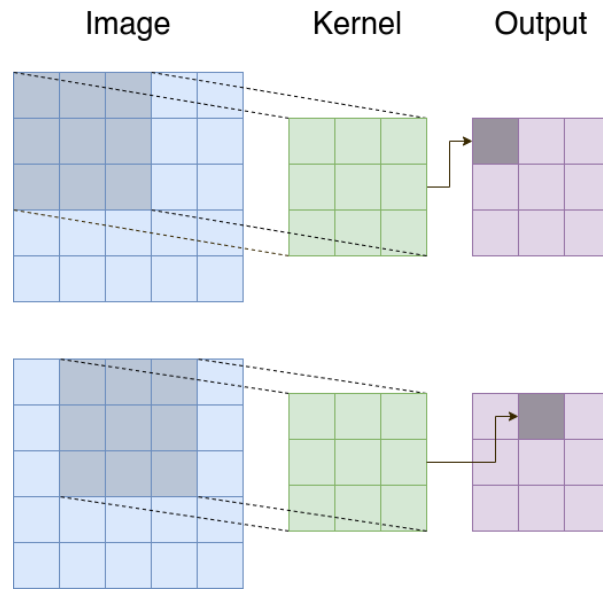
Figure 2.8: A simple visualization of a convolutional neural network: the kernel in green is applied to each 3x3 area of the image in blue, producing a cell in the output matrix in purple.

each convolution. This pooling layer performs a down sampling operation, using a fixed size and applying a simple operation such as a maximum or average.

Since the creation of convolutional neural networks, multiple improvements have been made to their architecture. An interesting development is the introduction of batch normalization (Ioffe and Szegedy, 2015), a mechanism that learns a mean and standard deviation from the training data in order to normalize even new data. Another development is the usage of residual connections within the model. These are connections that skip layers, and they have the benefit of reducing gradient vanishing issues during training. The first proposal of these types of model (He et al., 2016) greatly advanced the state-of-the-art for computer vision and it created the successful ResNet family of models, which are still used to this day.

### 2.2.6 Machine learning in Natural Language Processing

One of the most interesting aspect of computational approaches to paleography is it's multi-disciplinary nature: while the subject of studies might be glyphs, usually represented by images, they are also transcribing a language. For this reason, it is important to give a brief description of how modern natural language processing models work and how the contemporary techniques were developed over time from simpler ones.

| Text | I | did | not | find | the | map | so | get | lost |
|---|---|---|---|---|---|---|---|---|---|
| I did not find the map, so i did get lost | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| I did find the map, so i did not get lost | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 2.1: An example of how to represent sentences using unigrams. Notice how the two sentences have opposite meanings, but they share the same vector representation when using unigrams.

One of the most fundamental question that any machine learning approach for natural language processing has to deal with is how to represent textual information, or rather how to learn good representations for it. The simpler approaches are based on a simple count of words or combination of words, called n-grams. The simplest representation for a sentence is based on unigrams or counts of single words. This synthetic representation of text using unigrams is also called a Bag of Words (BOW) approach. This approach, however, is not always ideal as it might produce two identical vectors for different sentences. Table 2.1 shows that by only using unigrams we are not able to distinguish where negation is located in a sentence, resulting in two sentences with opposite meanings being represented by the same unigram vector. For this reason, it is common to combine unigrams with counts of combinations of words: bigrams, trigrams and so on. In this case, we gain the ability to see some of the more complex patterns that emerge from considering combinations of words, but we also have a more sparse vector, where it's harder for machine learning models to learn meaningful patterns, since most combinations of words are rare. This result is obvious, as considering combinations of more words result in them being more rare than the individual words that compose them.

More sophisticated approaches are based on word embeddings. These approaches are founded on the distributional hypothesis, which postulates that words that have similar meanings are generally surrounded by a similar distribution of words. In order to leverage this concept, then, we can train a model to predict a word from its context, in what's called the Contiguous Bag of Words (CBOW) approach, or we can use a word to predict the surrounding ones (Skip-gram). Both of these methods were proposed in one of the more successful approaches for creating word embeddings based on distributional semantics, called word2vec (Mikolov et al., 2013b,a). From these models we obtain dense vector representations of words, where each word in the vocabulary is represented by a vector, and the distance in the resulting vector space can be broadly thought of as a measure of semantic similarity. In addition to this, some analogies can be shown algebraically in the vector space, where, for example, we can observe that $\overrightarrow{king} - \overrightarrow{men} + \overrightarrow{woman} \approx \overrightarrow{queen}$. In this example, $\overrightarrow{woman}$ denotes the vector representing the word "woman".

Word embeddings are often used in combination with recurrent neural networks, models that are able to deal with sequences of inputs of arbitrary length while preserving memory about the whole sequence. More recently, however, a new class of models was developed, based on a model called the transformer (Vaswani et al., 2017). This model is usually pre-trained in a self-supervised way, by masking some words in a sentence or document and then teaching the model to reconstruct the missing word from the remaining textual information. One of the most prominent transformer models is called BERT (Devlin et al., 2019) and a single pre-trained model can then be re-trained (fine-tuned) for any specific task in computational linguistics. While a detailed description of BERT and transformers is beyond the scope of this thesis, it is important to note that they have multiple advantages when compared with more traditional models, as they are generally able to deal with longer sequences of words, resulting in improved performance in most natural language processing tasks. This, however, comes at the cost of an expensive pre-training procedure that requires a larger amount of data to train as well as increased computational requirements. Due to the fact that we are mostly interested in applying machine learning techniques to images, as well as the scarcity of available data and the relative short sequences that are characteristic of ancient, undeciphered

writing systems, the usage of transformer models is unfeasible in the context of this thesis.

### 2.2.7 Evaluating a machine learning model

An important aspect of any machine learning method is also the definition of criteria and procedures that can be used to evaluate its performance, in order to choose the best methodology for a given task and improve the state-of-the-art. In this section, we will discuss the main approaches and metrics that can be used to evaluate a machine learning model.

When annotated data is available and the model is trained in a supervised way, the first step is usually to divide our dataset in three: a training set used during training; a validation set that can be used to avoid overfitting of the model and to assess the model performance; a test set that is ideally kept separate, so that any choice of hyperparameter is not directly optimized for this specific test set. This way, we are not tuning the model for a specific test set, but instead we obtain a general model for our specific task. In order to evaluate our model, however, it is necessary to select one or more metrics that allow us to evaluate the performance of the model in an objective, measurable way. It is also important to consider the fact that, when using neural networks, the parameters of the model are initialized randomly at the beginning of the training procedure. This means that, to an extent, all the trained models will have a different behaviour. For this reason, it is often desirable to train multiple versions of the same model, initialized with different, random, parameters (Reimers and Gurevych, 2017). This way, a comparison between the metrics produced by different models can prove that an observed difference is not due to random chance. While this approach is ideal, one must note that there is a trade-off between the time needed to train multiple versions of the same model and the ability to achieve statistically significant results, not dependent on the random initialization of neural models. To achieve such a comparison, the mean and standard deviation of the metrics can be reported, and they can be accompanied by statistical tests such as the student t test or the Mc Nemar statistical test.

Among the metrics often used to evaluate machine learning models one of the simplest

one is *accuracy*, which is defined as follows:

$$a = \frac{|\{\hat{y}_i \in \hat{Y} \ : \ \hat{y}_i = y_i\}|}{|Y|} \tag{2.23}$$

Where:

- $Y$ contains all the labels of the test set for the task that the model needs to learn;

- $\hat{Y}$ contains all the predictions from our model.

Our formula, then, measures how many assignments ($\hat{y}$) from the model are correct, over the total size of the test set. When the number of classes is large, it sometimes makes sense to consider also a more relaxed version of accuracy. If a model produces a ranking of the most probable classes for each sample, we can check whether the correct one appears in the top N position. This concept is called top N accuracy.

While accuracy is very useful when the cardinality of each class that the model needs to distinguish, it can be very problematic when we observe imbalance between different classes. If, for example, we are interested in creating a model which is tasked with detecting hate speech online, a relatively rare phenomenon, the usage of accuracy can be very problematic. In this situation, if hateful comments on social media constitute only 1% of our test set, a model that always selects the "non-hateful" class would achieve a very high accuracy value of 99%. Furthermore, a more nuanced model that is able to distinguish some hateful comments at the expense of a lower accuracy over the majority non hateful class would certainly have a lower accuracy than the naïf approach of only selecting the majority class. For this reason, we can construct metrics that are better suited to deal with this situation, starting from some basic concepts. First, we select one of the classes as the positive class, so that we can apply the following concepts, extended from binary classification tasks:

- True Positives (TP): for our positive class, how many samples are classified correctly as belonging to it;

- True Negatives (TN): for our positive class, how many samples are classified correctly as not belonging to it;

- False Positives (FP): for our positive class, how many samples are classified incorrectly as belonging to it;

- False Negatives (FN): for our positive class, how many samples are classified incorrectly as not belonging to it;

We can then define two metrics that are derived from these values, *precision* ($p$) and *recall* ($r$):

$$p = \frac{TP}{TP + FP} \tag{2.24}$$

$$r = \frac{TP}{TP + FN} \tag{2.25}$$

Notice that precision and recall are defined based on the choice of a positive class, which means that each class will have its own precision and recall values. In the previous example concerning a very unbalanced hate speech dataset, we might want to optimize our model in order to obtain high precision and recall values for the hate speech class, since that is the goal of our model. In the previous example, we would obtain 0 precision and 0 recall, indicating that our model does not behave correctly. Let's consider another simple example to explain the usefulness of both metric. Let's imagine a test set of 1000 samples, split in two classes of equal size. If we use a naif classifier, which classifies everything in our positive class, we obtain a very high recall of 1.0 but a lower precision value of 0.5, which perfectly illustrates why both values are important when performing a classification task. A more synthetic metric which is often used is the harmonic mean between precision and recall, called *F1 score*:

$$F1 = 2\frac{pr}{p + r} \tag{2.26}$$

In our previous example, a simple arithmetic mean between a precision of 0.5 and a recall of 1.0 would be 0.75. The resulting F1 score, however, is 0.67, since the harmonic mean is more prone to producing lower values when one of the two metrics is low.

Aside from the standard metrics for classification, another set of possible metrics is available when dealing with clustering tasks. These are tasks where an unsupervised model attempts to categorize the data in a certain number of clusters. In this situation, it is crucial that the metrics are insensitive with respect to the indices that are assigned

to the different clusters, since this class of algorithms does not know about the indices assigned to them in the annotated test set. Additionally, since some clustering algorithms determine the number of clusters on their own, the metrics must not rely on the fact that the classes in the test set are as many as the clusters determined by the algorithm.

Among the metrics that are useful when clustering, there are two that can be thought of as analogous to precision and recall. Given a number of samples $N$, $C$ class labels, $K$ clusters, $n_{ck}$ samples belonging to class $c$ and cluster $k$ and $n_k$ samples belonging to cluster k, we define *homogeneity* as:

$$h = 1 - \frac{H(C|K)}{H(C)} \tag{2.27}$$

Where $H$ is Shannon's entropy, defined as:

$$H(C|K) = -\sum_{c,k} \frac{n_{ck}}{N} \log \frac{n_{ck}}{n_k} \tag{2.28}$$

Notice that the term $\frac{n_{ck}}{n_k}$ measures the ration between number of samples belonging to cluster k and cluster c and the total number of samples in cluster k. The intuition is then that a cluster containing many items from the same class will have a higher homogeneity value. Similarly, we can define *completeness* as follows:

$$c = 1 - \frac{H(K|C)}{H(K)} \tag{2.29}$$

If we expand the entropy term $H(K|C)$, we obtain a term $\frac{n_{ck}}{n_c}$, which indicates the ratio between the number of samples in class $c$, cluster $k$ and the total number of signs in class $c$. This means that for a cluster to be complete, it needs to include most of the samples from each class it contains. Like we saw for precision and recall, we cannot rely on a single metric to ensure a good clustering of our data, since we can easily construct degenerate solutions with high homogeneity or high completeness. A clustering with a single sample in each cluster produces an homogeneity score of one, since all clusters only contain a single class. Similarly, if our algorithm produces a single cluster containing all data, its completeness will also be one, since the single clusters contain all samples belonging to all the classes it contains. In order to create a more synthetic metric, *V-measure* can be used, which is defined as the harmonic mean between homogeneity and

completeness:

$$v = 2\frac{hc}{h+c} \tag{2.30}$$

Like for the F1 score, the usage of a harmonic mean penalizes situations where one of the metrics has a very low value compared to the other.

Another common metric that is used to evaluate the quality of clustering is the mutual information, which, like homogeneity and completeness, is based on information theory. Given a set $S$ of $N$ elements, consider two partitions of $S$, $X = \{X_1, \ldots, X_p\}$ and $Y = \{Y_1, \ldots, Y_q\}$ we can construct a contingency matrix between the two:

| $X \diagdown^Y$ | $Y_1$ | $Y_2$ | $\ldots$ | $Y_q$ | sums |
|:---:|:---:|:---:|:---:|:---:|:---:|
| $X_1$ | $n_{11}$ | $n_{12}$ | $\ldots$ | $n_{1q}$ | $a_1 = \sum_{j=1}^{n} n_{1j}$ |
| $X_2$ | $n_{21}$ | $n_{22}$ | $\ldots$ | $n_{2q}$ | $a_2 = \sum_{j=1}^{n} n_{2j}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| $X_p$ | $n_{p1}$ | $n_{p2}$ | $\ldots$ | $n_{pq}$ | $a_p = \sum_{j=1}^{n} n_{pj}$ |
| sums | $b_1 = \sum_{i=1}^{p} n_{i1}$ | $b_2 = \sum_{i=1}^{p} n_{i2}$ | $\ldots$ | $b_q = \sum_{i=1}^{p} n_{iq}$ | |

Table 2.2: Contingency matrix for the partitions $X$ and $Y$ of the same set $S$

Where $n_{ij} = |X_i \cap Y_j|$ represents the number of elements in common between the sets $X_i$ and $Y_j$. Then, the probability of a random object from $S$ to belong to $X_i$ according to the partition $X$ is:

$$P_X(i) = \frac{|X_i|}{N} \tag{2.31}$$

And the same equation can be applied to the probability of a point to belong to $Y_i$ partition $Y$. Finally, the probability that a point of $S$ belongs to $X_i$ in partition $X$ and to $Y_j$ in partition $Y$ is:

$$P_{XY}(i,j) = \frac{|X_i \cap Y_j|}{N} \tag{2.32}$$

Using these equations, we can define the mutual information between two clusterings of the same data as:

$$I = \frac{P_{XY}(i,j)}{P_X(i)P_Y(j)} \tag{2.33}$$

The mutual information of two random partitions of the same data, then, measures how much information (in bits or nats, depending on the basis of the logarithm in the

formula) one can acquire about one partition by observing the other. Different corrections can be applied to this formula, including adjusted mutual information, which corrects the formula for the effects of random chance and can be used to measure the performance of a clustering algorithm. In order to define this adjusted version, it is necessary to define the expected mutual information between two random clusterings, which can be calculated using a hypergeometric model of randomness:

$$E\{I(X,Y)\} = \sum_{i=1}^{p} \sum_{j=1}^{q} \sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N} \log\left(\frac{Nn_j}{a_ib_j}\right) \frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_i-n_{ij})!(N-a_i-b_j+n_ij)!}$$

(2.34)

Where:

- $(a_i + b_j - N)^+$ denotes $\max(a_i + b_j - N, 1)$;

- $a_i, b_i$ denote the sums shown in the contingency matrx (Table 2.2)

In addition to the expected value of the mutual information, we also need the entropy of a partition, given by:

$$H(X) = -\sum_{i=1}^{p} P_X(i) \log P_X(i)$$

(2.35)

Finally, we can define the adjusted mutual information as:

$$AI = \frac{I(X,Y) - E\{I(X,Y)\}}{\max(H(X), H(Y)) - E\{I(X,Y)\}}$$

(2.36)

While mutual information is a raw measure of information in bits or nats, therefore dependent on the type of data that we are interested in, adjusted mutual information has a range of $[-1, 1]$, where a random partition of the data has an expected adjusted mutual information of 0.

Another metric commonly found in the literature when evaluating clustering algorithms is the *Rand index*. Given a set S of elements and two different partitions of it in groups, $X$ and $Y$, we define:

- $a$ is the number of times that two elements belong to the same subset both in $X$ and $Y$;

- $b$ is the number of times that two elements are in different subsets in both $X$ and $Y$;

- $c$ is the number of times that two elements are in the same subset in $X$ but in different subsets in $Y$;

- $d$ is the number of times that two elements are in a different subset in $X$ but in the same subset in $Y$.

Using this values, we can define the Rand index as:

$$R = \frac{a + b}{a + b + c + d} \tag{2.37}$$

Intuitively, this metrics can then measure the ratio of how many pairs show agreement between two clusterings and the total number of pairs of elements in $S$. By using a pairwise metric, this metric can deal with the two partitions $X$ and $Y$ having a different number of subsets, and it is in the range $[0, 1]$. Similarly to what is possible for mutual information, it is also common to adjust Rand index for random chance, obtaining the adjusted Rand index, which is commonly preferred to the non adjusted version of the metric. Given two clusterings $X = \{X_1, \ldots, X_p\}$ and $Y = \{Y_1, \ldots, Y_q\}$ of a set S of N elements, we first construct the contingency matrix (see Table 2.2) used previously for the mutual information. Then, the adjusted version of the Rand index can be defined as:

$$AR = \frac{\sum_{i=1}^{p} \sum_{j=1}^{q} \binom{n_{ij}}{2} - \left[ \sum_{i=1}^{p} \binom{a_i}{2} \sum_{j=1}^{q} \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_{i=1}^{p} \binom{a_i}{2} + \sum_{j=1}^{q} \binom{b_j}{2} \right] - \left[ \sum_{i=1}^{p} \binom{a_i}{2} \sum_{j=1}^{q} \binom{b_j}{2} \right] / \binom{N}{2}} \tag{2.38}$$

Where $a_i, b_i$ are the sums defined in the contingency matrix (Table 2.2). Adjusted Rand index, unlike Rand index, has values between -1 and 1, having negative values when the Rand index is less than its expected value. A random partition of the data has an expected adjusted Rand index of 0.

While in some cases an annotated test set in order to evaluate the performance of a machine learning model, when applying computational approaches to ancient, undeciphered writing systems it is often the case that we have no certainty to base our evaluation on. In this case, it is useful to create a proxy task, by evaluating our models on similar

writing systems that have been deciphered if possible. Furthermore, the results from any neural model, especially one that has been trained in an unsupervised way, cannot be taken holistically. Instead, it is useful to examine overarching trends and task the model with specific scientific questions, instead of using more broad techniques such as a complete clustering of the data. Another possibility is the selection of a few data points that are not in contention among experts to perform an evaluation that has a more limited scope. Another crucial aspect of these kind of studies is the use of statistical tests whenever possible: since it is impossible to produce an evaluation of the model based on the data, we can at least rule out the effect of chance on our results, showing that the positive results from the model would be very rarely produced by random chance alone. Crucially, any claim of evidence on contentious issues must be preceded by a thorough evaluation of the model itself, as the level of accuracy of the model can have a serous impact on the final results. Finally, any type of evaluation for an unsupervised model for undeciphered writing systems must be tailored to the specific questions that we attempt to answer using the model, as well as the specific characteristics of the writing sytem.

# Chapter 3

# State of the art and challenges of computational paleography

When dealing with ancient writing systems using computational techniques and machine learning in particular, it is crucial to consider both the overarching challenges that arise from this domain and the previous approaches that have been proposed in the field. In particular, the two main challenges that we had to face when dealing with ancient, undeciphered writing systems were the relative scarcity of data available and, crucially, the fact that some cases there is no agreement over some aspects of the writing language, which can lead to the usage of unsupervised machine learning models. In this chapter I focus on these two aspects, while providing insight into the state-of-the-art for both low resource learning and unsupervised learning applied to images and natural language.

## 3.1   Low resource learning

One of the main factors that determines the effectiveness of any machine learning approach is the availability of data. Even ignoring the presence of annotations for the task that we need to perform, any approach requires a sufficient amount of data to produce meaningful results. In fact, increasing the amount of data available during training, even without altering the techniques involved, typically results in improved performance. In addition to this, the main risk that arises from applying a powerful

neural network with a very limited amount of available data is overfitting: since the amount of data is so limited, the model will be able to memorize training data instead of learning meaningful patterns, resulting in a model that is less able to deal with new data and a decrease in performance for the test set. For this reason, an important trade off must be considered, between the expressive power of the selected model and limiting the number of its parameters, so that it can avoid overfitting. When the amount of data is limited, in fact, it can be useful to resort to simpler models with fewer parameters to train, as they are less susceptible to overfitting when compared with more powerful neural models. While this is a sensible approach, in practice the usage of simpler models is a double-edged sword, since their limited expressive capability can be insufficient to perform the desired tasks.

In addition to a limited amount of training data in general, the other issue that arises in most models dealing with signs from ancient writing systems is the presence of graphemes that are very rare, sometimes appearing only once in the available data. In fact, due to the Zipf distribution that arises from considering the frequency of letters, this can be true even for alphabets such as English. One analysis of letter frequencies for cryptographic goals (Lewand, 2000) shows that the least frequent letter "Z" has a frequency of 0.0007, appearing on average only 7 times in 10 thousand letters. This problem is exacerbated when we are dealing with syllabaries and logographic writing systems, as in this case the number of signs increases and, in turn, their individual frequencies are smaller than for alphabet letters. For example, considering hiragana, one of the two syllabic systems used to write Japanese, we can observe that, in a corpus of 20 million signs, the most frequent hiragana grapheme is attested approximately two million times, while the least frequent one appears only once (Chikamatsu et al., 2000). When the frequencies are so low, we are unable to learn any pattern from a handful of signs, and any result involving them needs to be carefully evaluated as modern computational approaches are generally unable to learn meaningful patterns from so few attestations. Since most early writing systems use syllabograms and logograms and undeciphered ones usually have corpora that are in the tens of thousands of signs or less, we are bound to have to deal with hapaxes and, crucially, it is quite possible that our corpora do not contain all signs or combinations of signs that were originally used in the writing system.

In addition to this, there are other factors such as the usage of different media to write, variations in the sign shapes in different geographic areas and at different times, which can complicate any analysis, manual or automatic.

In recent years, in the field of low-resource learning multiple approaches have been developed for dealing with situations where data is very scarce. In particular, some approaches are aimed at the study of how machine learning models behave when a very limited amount of training data is available:

- Few-Shot Learning: in this context, we are interested in performing a task that is only shown to the model a few times, sometimes even once (One-shot learning);

- Zero-shot learning: in this context, our aim is to perform a task that has never been seen by the model.

While at first glance Few-Shot learning and Zero-shot learning might seem similar, there is a crucial difference in the approaches that are used for these tasks. For one or few-shot learning, we can for example apply some form of data augmentation, in order to produce multiple augmented versions of the same sample, which can then be used in the training set. For example, we can apply cropping, color distortion, translations and rotations to images, as well as random horizontal or vertical flips, so that we can produce an arbitrary amount of different versions of each image. For textual data, one possible solution is to use synonyms and perform random replacements for individual words, so that a single sentence can be represented in multiple ways. More sophisticated approaches are based on pre-trained word embeddings (Wang and Yang, 2015). These approaches are of course very dependent on the kind of data and task that the model needs to perform, as for example adding random typographical errors might be beneficial for a model dealing with noisy data from social networks, while it might be detrimental when we are interested in analyzing text from a book which has been proof-read and contains few typographical mistakes. These augmentation-based approaches, in addition to helping with few shot learning and, in general, with a limited amount of data, can be used even when data is abundant, as they can improve the robustness of the models and reduce the memorization of data from the training set.

Another class of solutions can be applied to both zero-shot and few-shot learning,
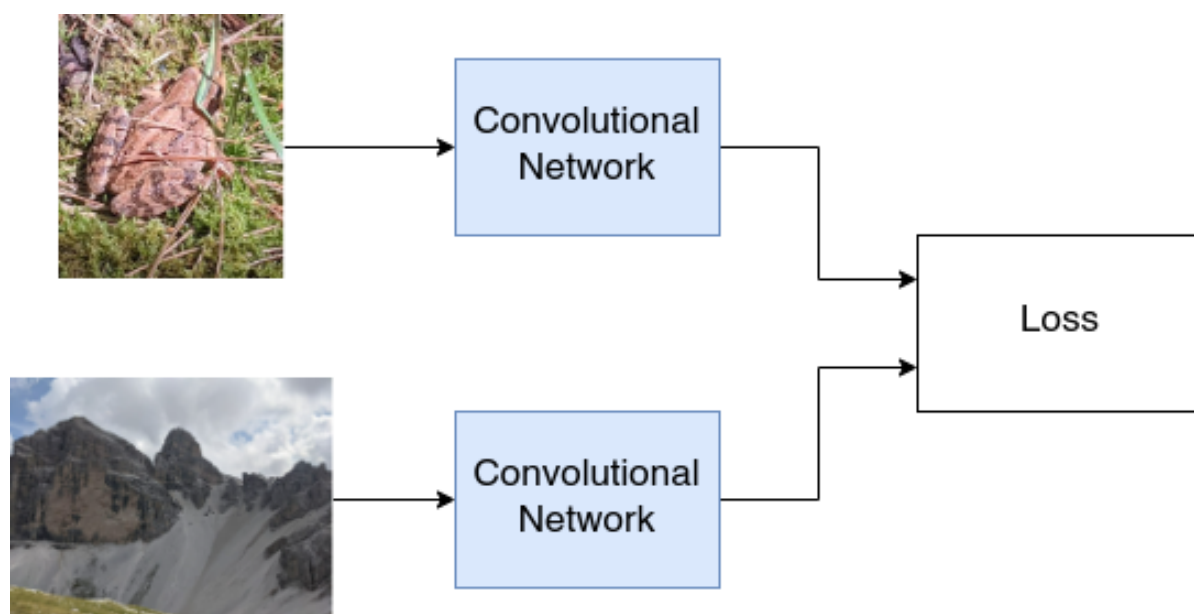
Figure 3.1: Basic Architecture of a Siamese network

and it is based on the idea that the model can be taught different but related tasks, then applied to a new one, using a method called transfer learning. One possible example of this approach in computer vision is the application of so-called siamese networks (see Figure 3.1). These networks are called siamese because they use two identical branches. These branches are used to produce two vector representations for two images. Finally, a loss function, for example the triplet loss (Chechik et al., 2010), can be applied to the two vectors from the two identical branches of the network, typically involving a distance metric in its formulation. The idea behind this loss is to minimize a distance function for identical couples of samples, while separating different pairs. The resulting vector space, then, can be used to compare different images. Furthermore, it has a crucial advantage over a more traditional classifier, which typically assigns a neuron to each class and then trains the model to produce a higher value for the correct class for each training sample: since the model only works on pairs of images, it can produce results even for new, unseen categories and it does not generally require an a priori definition of the different classes partitioning the data. The application of a siamese network requires training over a task that is very similar to the zero-shot task, as otherwise the model cannot learn any useful information for the other task. This means that, when dealing with ancient,

undeciphered writing systems, it can be very problematic to choose a task that can fill this role, since the unknown nature of the scripts does not allow a complete comparison with other writing systems. Additionally, since the training happens on another script, it can be challenging to incorporate information that is known about the target writing system, such as the attested sequences that compose it.

The approach used in siamese networks is also similar to other kinds of zero-shot learning methods. With powerful transformer models, in fact, attempts have been made to perform zero-shot tasks in natural language processing. These approaches are able to leverage the extensive pre-training procedure that is applied to these models, in order to ask them to perform tasks that are either unseen or for which very few training samples are available. Most of these approaches are based on language models, which means that they are able to produce text following an initial sequence of characters that is provided by humans. Therefore we can perform a task that has never seen before by the model by just providing an the initial sequence in the form of a question, so that the model can continue the text by answering it. There are many approaches that follow this idea (Brown et al., 2020; Wei et al., 2022) and it is one of the most interesting developments in the field of natural language processing. While these methods are surely interesting for the field of natural language processing, the amount of data required is unattainable in the context of ancient, undeciphered languages. Furthermore, they require a perfect transcription of the textual content, which is also not always available.

In this section, I examined some of the main research approaches with the aim of tackling the scarcity of data for machine learning models. While these approaches are interesting and informative for anyone attempting to build models for ancient undeciphered writing systems, they require a large amount of data that can be used for any transfer learning procedure, which is not always available. Most machine learning approaches for paleography are in fact undermined by the amount of available samples, which, unlike what happens for most tasks involving linguistic tasks, where new data can often be collected, is a hard limit, since scholars have no control over the possibility of finding of new documents.

## 3.2    Unsupervised learning

When dealing with signs from ancient, undeciphered languages, it is sometimes desirable to use a completely unsupervised approach, since the disagreement between experts can be substantial and it is therefore crucial to produce results that do not rely on any theoretical point of view regarding the subject matter. This section constitutes a brief discussion of the state-of-the-art in unsupervised learning for both images and natural language, as both domains are involved when dealing with writing systems.

While we did not characterize them as such, we have in fact discussed about some common unsupervised methods for natural language, since any pre-training procedure for transformer models is, in fact, unsupervised, even if some authors prefer to use the term self-supervised in this context, since the final goal of the model is to use the pre-training model only as an intermediate step for a different kind of task. While the main goal of pre-training is to produce general models that can be applied to multiple tasks by fine-tuning them with labeled data, unsupervised approaches do exist and they are also based on similar tasks as the ones used by BERT and other transformer models. While many different tasks are possible, one notable field is unsupervised machine translation. These models are tasked with translating text from one language to another without having access to parallel sentences translated in both languages. These approaches can use autoencoder models, where text is encoded as a relatively small vector by one portion of the model (encoder) and then it gets reconstructed by the other portion (decoder). More recently, the usage of (masked) language models is more commonplace. In both cases, we can imagine building an unsupervised machine translation model that is trained on both English and Italian text. In addition to each individual word, the encoder is provided with an indication of the language of the text, while the decoder is also provided with the language that we want as an output of the model. Now, by training the model to encode both English and Italian sentences using the same weights, we produce a single vector space for both languages. By providing an English sentence to the encoder and asking for the Italian to the decoder, we can then attempt an unsupervised machine translation, that is only possible because some portion of words are shared between languages and therefore they act as a source of alignment for the model. Many models have been constructed by using similar methodologies (Lample et al., 2018; Zhu et al., 2020) and

they constitute one of the main applications of unsupervised learning in natural language processing.

Another common application of unsupervised methods is the categorization of topics in textual content. The goal of these methods is to categorize different documents in multiple topics, usually basing this choice on the frequency of words found in them. In this field, the most well-known approach is Latent Dirichlet Allocation (LDA), which is a Bayesian model that is able, when provided with a number of topics, to partition documents in different groups. It works by deriving a topic-word distribution, which determines how probable it is for a word to belong to a certain topic. It then uses this information to derive document-level classification in different groups.

While we provided an overview of some approaches for unsupervised learning in natural language processing, the domain of computational paleography also includes the application of machine learning models to the graphical representations of signs. For this reason, it is useful to examine the main approaches in unsupervised learning for images, mostly based on convolutional neural networks. In particular, we can categorize different approaches based on the main task that they perform in order to teach the model how to produce good quality vector representations for images. One of the simpler alternatives in this situation is to build an autoencoder (Kramer, 1991), which is tasked with reconstructing the image from a dense low dimensional representation. More sophisticated methods task the model with the reconstruction of a jigsaw puzzle (Noroozi and Favaro, 2016), or with matching two augmented versions of the same image (Caron et al., 2020). Another class of methods uses clustering as a proxy: by applying a convolutional neural network to images, we obtain vector representations for each one of them. Once we obtain these vectors, we can use a clustering algorithm such as K-Means to cluster the data and use the categorization derived from this procedure to train the model with pseudo-labels (Caron et al., 2018). Another approach called SCAN (Van Gansbeke et al., 2020) uses a two step approach: fist, it minimizes the distance between each image and an augmented version of it, then it finds the nearest neighbors for each image and classifies each image to a group depending on its nearest neighbors.

Since many different methods are available for unsupervised learning, depending on the type of task that the model needs to solve and on the type of data that is involved,

not all of them are a good fit for paleographic data. In particular, most transformer based models require an amount of data that is not available to us. Even if that was not the case, any unsupervised machine translation approach relies on the existence of common terms that can be used as "anchors" to match the two languages. This condition is not possible for undeciphered writing systems, where it is often impossible to determine which language the script encodes, even when a phonetic reading is available for the signs. When investigating the sign inventory and therefore considering the categorization of signs themselves, the application of unsupervised models for images is also challenging, as they do not consider linguistic context as a feature and therefore they need to be adapted to the domain. However, their application can be useful and using powerful convolutional models can be less challenging than the application of transformers in the context of ancient writing systems.

## 3.3    State-of-the-art in computational paleography

In recent years, a number of scholars has attempted to apply computational techniques to ancient writing systems. In general, we can divide these approaches in two different strands: on one hand, we have purely textual based methods, which deal with ancient texts that have already been transcribed. On the other, we have methods that involve also the graphical representation of signs in order to produce their results.

Among the purely textual approaches, two articles have been published dealing with the restoration of ancient Greek texts that have been damaged. In Assael et al. (2019) an encoder-decoder model using a recurrent component (LSTM) with an attention mechanism is applied to the restoration of ancient Greek text. In particular, the best performing model combines a character level and word level representations for documents, using context to reconstruct damaged sections of the inscriptions. In order to train and evaluate the model, complete sequences have some of their text masked by using a special character "?", which indicates that the model needs to predict their value. The resulting model, called PHYTIA, is then evaluated using a specific metric, the character error rate, which measures the fraction of characters that are incorrectly predicted by the model. PHYTIA is able to achieve a 30% character error rate, which is even better than the

57% character error rate achieved by a human expert. In a follow up article by the same authors (Assael et al., 2022) an improved model is constructed for the same task, called ITHACA. This model is based on a transformer architecture and, in conjunction with the ability to reconstruct damaged portions of text, it is also able to provide a prediction for the time period that the document was inscribed in and its geographical provenance. In order to provide a useful tool for researchers, ITHACA is able to output the top 20 predictions for missing signs and histograms representing the most probable time periods and locations for each document. The model, like its predecessor, has been evaluated on sequences where some text has been removed, and it improves upon PHYTHIA in a substantial manner. On this new test set, while PHITYA achieves 32% accuracy, ITHACA is able to achieve an accuracy of 62%. The results for the geographical prediction are also encouraging, with an accuracy of 71%, while the model is able to predict the correct year for an inscription with an average error of $\pm 29$ years.

Similarly to the ancient Greek models, another method has been proposed to reconstruct missing signs in Babylonian Akkadian (Fetaya et al., 2020). Like ITHACA, it is a recurrent neural network. However, instead of training a masked language models, the authors used a character level language model as their training task, meaning that they trained the model to predict a sign from those preceding it. As a baseline, they used a simple n-gram model which uses either the characters preceding the current one or the characters surrounding it in order to predict the current character. A first evaluation has been performed by only removing a single sign, and it compares an LSTM that is able to use only the beginning of the sentence before the missing sign (LSTM start), an LSTM using the whole sentence (LSTM full) and 2-gram models using the two signs before the missing one (2-gram start) and the two around the missing one (2-gram full), respectively. Surprisingly, 2-gram full is able to outperform LSTM start, obtaining an accuracy of 75%. The best performing model is the LSTM full one, with an accuracy of 85%. In addition to this evaluation, the authors provide an interesting test where they increase the number of signs that need to be predicted by the model, showing that when predicting 2 signs the accuracy decreases to 48%, while with three tokens the accuracy is very low at 24%. While the top-5 and top-10 accuracies have higher values, this results shows that, despite the model's ability to reconstruct a single damaged sign,

its effectiveness decreases dramatically when it is task with the reconstruction of longer sequences.

Another application of computational methods to undeciphered writing systems regards the possibility to apply automatic methods to the transliteration of undeciphered scripts. These approaches use a symbolic representation for signs, since they are usually focused on the recognition of cognates, words that are related or have the same origin, between an undeciphered script and a related known writing system. By reconstructing cognates, these systems also provide a transcription of each sign in the undeciphered script, since they operate by transcribing between the signs in the two writing systems. Additionally, most of these methods also rely on the prior knowledge about the related writing system, since it is compared with the undeciphered script in order to reconstruct cognates. It must be noted that, so far, no complete decipherment of any undeciphered script has been achieved using automatic means.

The existing attempts to decipher unknown scripts through computational means started with more conventional, statistical methods, while in recent years some neural approaches have been presented. The first contribution to the field of automatic decipherment of ancient scripts was proposed in Knight and Yamada (1999); Knight et al. (2006) and they consist of a Hidden Markov Model, which learns a mapping from signs to the corresponding phonemes. In Snyder et al. (2010) a non-parametric Bayesian framework is proposed, which considers both the mapping between signs of related scripts, as well as mapping between morphemes. Berg-Kirkpatrick and Klein (2011) propose a framework based on an objective function that is able to match both the alphabets and the lexica of two writing systems. A coordinate descent procedure is then used to find solutions to the resulting combinatorial optimization problem. To the best of our knowledge, Luo et al. (2019) describes the first neural approach for automatic decipherment of ancient scripts, based on a sequence-to-sequence model, informed by patterns in language change derived from historical linguistics. The model is trained in an unsupervised way by using a minimum-cost flow approach. Finally, in Luo et al. (2021) a neural model is informed with phonetic priors, in order to tackle a situation where the segmentation of the unknown script is partial and the related language is also unknown.

While the decipherment of unknown scripts is a common goal for computational

methods for ancient writing systems, these approaches have never been tested on un-deciphered scripts to the best of our knowledge. Additionally, the main focus of this thesis is Cypro-Minoan, and one of the open questions is whether it transcribes a single language or whether multiple languages are encoded. For this reason, techniques that aim at automatic decipherment are not used in this thesis.

In addition to the aforementioned approaches that are aimed at the reconstruction of missing sequences of text, another similar goal is the matching of fragments. When a document is broken, it is often useful to attempt to determine whether the fragments are part of the same document and whether it is possible to join them to reconstruct the full textual sequences. In order to match papyrus fragments, a model called Papy-S-Net has been proposed in Pirrone et al. (2019). In order to achieve this goal, a training set is created, by artificially cutting images of a single papyrus in multiple fragments. Then, a convolutional siamese network is trained to determine whether two fragments belong to the same papyrus or not, using a simple two class classification loss and applying softmax to the final layer of the network. The authors compare the model to a similar siamese network (Koch et al., 2015) that uses two consecutive convolutions instead of one, as well as a similarity measure. They show an improved performance of their model in terms of true positive rate and true negative rate, which have high values of 79% and 83%, indicating that it is possible to apply a siamese model to match fragments of papyrus with a relatively high rate of success. In Ostertag and Beurton-Aimar (2020) some of the same authors improve and extend their siamese network to apply it to pottery markings from the Hut-Repit site in Egypt. In addition to performing a matching task to determine whether two fragments belong to the same object, they also attempt to determine the relative position of the two fragments. This way, they can apply a global reconstructing pipeline based on a directed graph, constructed by pairwise alignments, where each node is a patch and the edges encode the alignment direction. Each patch can have up to 4 neighbors, one in each direction. From a set of N patches that might be from the same object:

- A patch is chosen at random and it becomes the "query patch";

- Each of the other patches is compared with the query to determine the alignment direction. The alignment is considered as a candidate if the probability that the

siamese network determines is higher than 0.7.

- If the alignment direction is considered correct, a new edge is created and the procedure can continue, until all edges have been examined.

While the authors do not provide an overall evaluation of their graph-based pipeline, they show that their model is able to achieve 96% accuracy on a large dataset of pottery.

Another interesting task that has been attempted with computational methods is the concept of Scribal Hands attribution, where the aim is to determine whether two or more documents have been inscribed or written by the same person or not. In particular, a forensic investigation of Hebrew ostraca has been performed in Shaus et al. (2020), by using two different algorithms, one based on a combination of features extracted from images, the other on binary pixel patterns. Interestingly, the two algorithms have been compared with the analysis from a forensic document examiner. While there is no ground truth to assess the validity of the computational approaches, they show no disagreement with the expert, since they only diverge in situations where the two algorithms remain agnostic, while the expert hypothesizes that the documents were written by the same person. In total, the authors identify 12 different writers, which has interesting implications for the diffusion of literacy in this period.

In Popović et al. (2021) an investigation is conducted over the Great Isaiah Scroll, belonging to the Dead Sea Scrolls, which constitute the oldest manuscripts of the Hebrew Bible and other Jewish texts. An open question regarding this text is whether the observed variations in writing style indicate the contribution of multiple scribal hands or not, in particular concerning a transition that is present from two set of columns in the document. In this article, the authors perform multiple analysis on the Great Isaiah Scroll. First, they separate the document in columns, and extract a column image from each column. The primary analysis uses various methods to extract features from the column images, which are then used to measure the chi-square distance between them. Since no ground truth is available to evaluate the raw distances, the authors use Principal Component Analysis to perform dimensionality reduction on the feature vectors and examine the resulting vector space visually in three dimensions. After this primary test, the secondary analyses are aimed at establishing whether it is possible to detect a transition of style between the two sets of columns. In a secondary step, a different kind

of feature extraction is performed over the columns, followed by various steps leading to the computation of nearest neighbors for each column. With this data, it is possible to reveal the phase transitions between the two scribal hands, by determining a change in behaviour of the nearest neighbors when the two scribes alternate in the document. Finally, a tertiary analysis was performed by producing visual tools that can aid experts in a post-hoc evaluation of the variation of shapes in the scroll. During the primary analysis, it is possible to observe a separation between the two sets of columns in the three-dimensional visualization of the dataset. From the various techniques in the secondary test, a phase transition can be observed that shows that, when transitioning from one column set to the other, the nearest neighbor of each column changes significantly on average: when in the first set of columns, the nearest neighbor is in the first set, and the same is true for the second set. The authors are also able to guess a phase transition that is in line with the hypothesized participation of two scribes to the writing of the document. Finally, they can highlight the variation in shape that is observed in characters in the two column groups. These extensive results show promise and they provide a strong indication that two scribal hands were involved in the drafting of the Great Isaiah Scroll.

One final example of a computational approach for scribal hands discrimination is Srivatsan et al. (2021), which focuses on creating a neural model that is able to distinguish between scribal hands in Linear B. In particular, the model uses a combination of a reconstruction loss and two discriminative loss components. In addition to the image, the authors use two embeddings to represent the grapheme and the scribe, respectively. This means that two glyphs from the same document will share the same grapheme embedding and scribe embedding. These two features are used in a decoder network, which uses transposed convolutions to recreate the original image. Crucially then, the model is not able to access the original image, but it needs to produce a prototypical shape that a certain scribe would use for a specific grapheme. The two discriminative components of the loss are used to teach the model to determine whether a glyph was inscribed by a given scribe and whether a glyph represents a given grapheme. To perform these tasks, the image and embeddings for the scribe and grapheme, respectively, are fed to a discriminative network including a convolutional component. This way, the model is

taught to determine, from an image and its corresponding scribe or grapheme embedding, whether the image was inscribed by the scribe encoded by the embedding and whether the image represent the grapheme with the given embedding. At each training step, the discriminators are provided with one true match and one false match, in order to train their cross-entropy based losses. The dataset used for tests was extracted from drawings present in the literature and multiple augmented versions of each sign were produced by applying horizontal and vertical translation, dilation and erosion. This step was performed in order to increase the availability of data for the training procedure. In order to evaluate the model, a comparison is performed with a basic autoencoder model, showing that there is a significant improvement in terms of F1 score. Another metric, QVEC, is used to compare the scribe embeddings learned by the model with manual features extracted manually by experts in Skelton (2008).

So far the discussion of the state-of-the-art in computational paleography has focused on deciphered writing systems, however there are some instances in the literature of systems designed for undeciphered scripts. One such script is Proto-Elamite, which has been researched with computational methods in various articles. In Born et al. (2019) the authors propose a hierarchical clustering and a topic modelling approach for Proto-Elamite and manage to produce results that were already obtained manually, as well as new findings. In order to perform a clustering of the signs, the authors use the transcriptions of the signs and propose three different techniques:

- a neighbor-based approach that uses as a feature the number of times a sign appears before or after each sign in the corpus;

- a HMM that clusters groups of signs based on an emission model with 10 hidden states;

- another clustering that depends on the distributional properties of the signs, called a generalized brown clustering.

This multi-algorithm approach allowed the authors to compare the clusterings. Their findings suggest that, while many previously established graphemes are indeed grouped together consistently independently of the approach, some inconsistencies are present, as well as some other consistent behaviours that were not part the previous literature

about Proto-Elamite. Another contribution of this article is the application of a 10 topic LDA model to the corpus, which also yielded promising results.

In Born et al. (2021), some of the same authors continued experimenting with computational techniques applied to Proto Elamite, focusing on an interesting feature of the writing system: complex graphemes. These are graphemes that appear to be composed from simpler ones, with various modes such as framing or inscription of one sign in another. While the orthographic compositionality of these signs is evident, it is not clear whether some form of semantic compositionality is also involved. In order to investigate this question, the authors use an image-aware language model based on a convolutional neural network followed by a recurrent layer, that is able to consider both the transcription of signs and their graphical representations. They then use the vector representations learned from the signs to test for compositionality, using every combination of textual and image features. Their results show that a degree of semantic compositionality exists for a subset of complex graphemes at the very least, since this compositionality can be detected even while excluding visual features from the model.

# Chapter 4

# Linear A fractions

The aim of this chapter is to describe our attempt at decipherment for the Linear A fraction values by using a multidisciplinary approach, involving both computational and paleographic considerations. This work has been presented in Corazza et al. (2021) and it involves the application of a constraint programming approach in order to extract all possible solutions from the given constraints. Then, we develop multiple metrics in order to objectively evaluate the quality of each solution and produce a plausible decipherment. Crucially, the usage of a constraint-based approach is possible thanks to the fact that deciphering a system of numeric notation involves applying rules that can be formally defined in mathematical form, as well as many properties that can be measured using ad-hoc metrics. To the best of our knowledge, this is the first attempt at deciphering a system of numeric notation using constraint programming and well defined, formal, definition for the overall quality of the favored solution. In addition to the usage of constraint programming and metrics, paleographic considerations are also used in order to assess the possible values of the fractions, resulting in a multi-stranded, iterative, approach that is nevertheless based on objective constraints and measures.

## 4.1  The Minoan numbers: integers and fractions

Since most of the documents in Linear A are administrative in nature (Salgarella and Castellan, 2020), the usage of signs that denote numerical values is very commonplace

in the available texts. As discussed in section 1.1.2, the Minoans used a system of notation for integers which is broadly shared with the other writing systems from the Aegean and Cyprus. In particular, it is a cumulative additive system and the numbers are written in descending order from left to right. The system uses base 10, with signs for ten thousands, one thousand, one hundred, ten and one (see Table 4.1). In order to transcribe a value of 927, for example, the Minoans would write, from left to right, 9 times the "100" sign, then two times the "10" sign, and finally 7 times the "one" sign.



Figure 4.1: The system of numerical notations for integers in the Aegean. On the top, the raw, base 10, values; on the bottom their usage in a cumulative-additive way to produce double their respective values.

In addition to the signs for integers, another system of numerical notation was used in Linear A to transcribe fractional values. This system is comprised of 17 signs (see Figure 4.2), transcribed in the literature (Olivier and Godart, 1975) with capital letters A, B, D, E, F, H, J, K, L, L2, L3, L4, L6, W, X, Y, and $\Omega$. Like the system of notation for integers, the fractions are cumulative and additive, with the fractions appearing in order of decreasing value from left to right. Additionally, the fractions can be used in combinations with integers when transcribing numerical values that are larger than one.

Another peculiarity of the Linear A fractions is the presence of a sub series within the system: the L series. L, L2, L3, L4, L6 are all formed by the same basic shape and a number of horizontal lines around it. This suggests that the values in this sub series are somehow related, which is a fact that is used extensively in our investigation. Other fraction signs also seem to share similar shapes, perhaps indicating a progression, like the
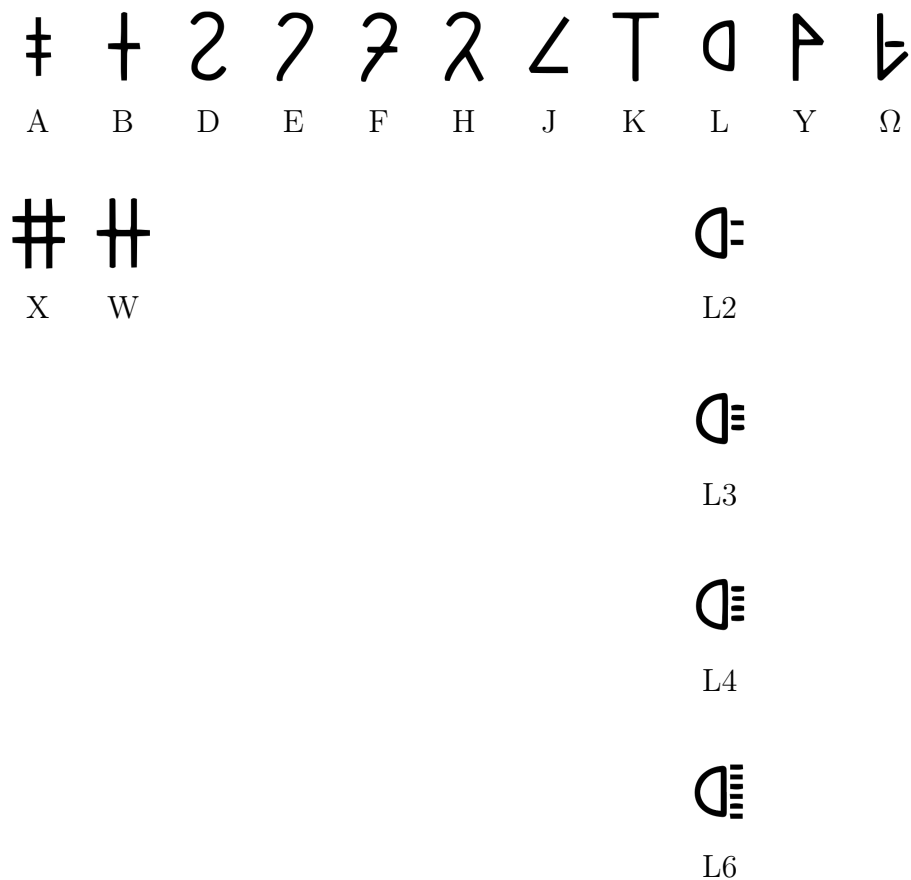
Figure 4.2: The Linear A fraction signs

signs J, E, F and H. The signs A,X and B,W, instead, might form pairs, where X=AA and W=BB.

With regards to their usage, the fractions are mainly found in combination with logograms, which are used to indicate different types of commodities. For this reason, these documents are considered to be mainly used for accounting, recording the amount, encoded as a vertical or horizontal sequence of fractions, and type for each commodity. This structure is the main motivation for the belief that the fraction system, like the integers, are cumulative and attive in nature, with the larger valued fractions appearing before the lower valued ones. Interestingly, unlike Linear B, Linear A does not appear to include units of measurements. Instead, the units of measurement are implicitly conveyed

by using the logogram associated with each specific commodity. For this reason, it is not clear whether dry and liquid products were measured using the same unit or if different measures were used (Schoep, 2002). Nevertheless, it is reasonable to assume the values of the fractions would not change depending on the type of commodity, since the shape of the fraction signs does not change for different types of products.

## 4.2    The sample

In order to produce meaningful results, it is crucial to select a coherent epigraphic sample for the Linear A fraction signs. For this reason, we decided to focus on a specific period, which is the one with the most abundant presence of fraction signs, since their presence is not consistent in volume across all the archaeological sites and periods. In particular, our study focuses on the Late Minoan I period (approximately from 1600 to 1450 BC) and it comprised documents that are more frequently found in the west, central and eastern portions of Crete. Due to the cumulative-additive nature of the fractions, it is crucial to identify, within our sample, all combinations of signs that are attested, as well as the number of attestations for each fraction sign (see Table 4.1). By examining these frequencies, we can already spot some interesting trends in the usage of the fractions by the Minoans. In particular, we have some signs, namely J and E, that constitute alone approximately 50% of the total sample. Additionally, most signs are not found in combination with themselves, with the exception of D, B and Y. DDDD is also the only combination of four fractions that is attested in the sample.

Another aspect of the fraction system in Linear A is the fact that we have no single document that contains numerical calculations without errors or damaged portions (Montecchi, 2009, 2013). For this reason, multiple attempts at decipherment result in diverging solutions, since very few values can be established with certainty. In Bennett (1950, 1980, 1999) the author hypothesizes that most fraction signs in Linear A were probably characterized by a numerator of 1, by using a comparison with the contemporary Egyptian script. Using their frequencies and their combinations, then, he produced a set of possible assignments for the values of the fractions. Since J is the most frequent sign, both when considering isolated instances and combinations, Bennett assigned to

| Total attestations | | Attestations of combinations | |
| --- | --- | --- | --- |
| J | 137 | JE | 24 |
| E | 105 | DD | 14 |
| D | 68 | BB | 5 |
| B | 36 | EF | 5 |
| K | 32 | KL2 | 5 |
| L2 | 26 | EL2 | 3 |
| F | 23 | JB | 3 |
| H | 19 | JEL2 | 3 |
| A | 15 | JL2 | 2 |
| W | 4 | AA | 1 |
| Y | 4 | DDDD | 1 |
| L3 | 3 | EB | 1 |
| L4 | 2 | EL4 | 1 |
| L6 | 2 | EL6 | 1 |
| X | 2 | FK | 1 |
| L | 1 | HHH | 1 |
| Ω | 1 | JA | 1 |
| | | JEB | 1 |
| | | JF | 1 |
| | | JH | 1 |
| | | JK | 1 |
| | | LL2 | 1 |
| | | L2L4 | 1 |
| | | YYY | 1 |

Table 4.1: Attestations of signs and combinations of signs in documents with no scribal errors, erasures or doubtful readings.

it the value of $\frac{1}{2}$. Similarly, it could be inferred that the value of E was probably $\frac{1}{4}$, since it is the second most frequent sign and it appears in combinations after J, indicating a value smaller than $\frac{1}{2}$ according to Bennett. Finally, the value of F was set to $\frac{1}{8}$ using similar considerations. Interestingly, the values of these three fraction signs are supported by the sums found in three different documents. Clay tablet HT 104 contains a sum and a total: $45 + J + 20 + J + 29 = 95$. By solving this equation, we obtain that $2J = 1$, which confirms the fact that $J = \frac{1}{2}$. Another example can be found on clay

tablet PE 1, where a sign group containing the VIR/MUL logogram representing people can be found, in addition to another pair of logograms GRA+PA, specifying grain and its variety. In the second entry, 72 people and 36 GRA+PA are registered, suggesting that each person is assigned $\frac{1}{2}$ of some unit of measurement for grain. In the first entry, then, we see a value of 50 people with a damaged integer beside it, as well as a value of 26J grain. Supposing that the missing integer beside 50 is three, then, the proportion would be consistent with the one found in the second entry, suggesting once again a value of $\frac{1}{2}$ for J. Finally, a progression is inscribed on the graffito on stucco HT Zd 156, where we find $1, 1J, 2E, 3EF$. If we apply the values suggested by Bennet, we can see that the series is a progression where each number is 1.5 times larger than the last one: $1, 1 + \frac{1}{2}, 2 + \frac{1}{4}, 3 + \frac{1}{4}, 3 + \frac{1}{4} + \frac{1}{8}$, as discussed in Pope (1960). We can therefore safely assume that the values for J, E, F are $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ respectively.

Beside Bennett, other authors have proposed possible values, as shown in Table 4.2. By examining this table, we can already see some trends in previous attempts at decipherment. The first regards the fact that the more frequent a sign is, the more authors tend to propose readings for it. In a similar observation, the most commonly attested fractions tend to have higher values and they show a higher level of agreement between authors. Another interesting observation is that for some values no value has been proposed: for L we only have a tentative value from Cash and Cash, while for $\Omega$ no attempt at decipherment has been made. This is of course expected, since any attempt at decipherment is only possible when there are combinations of fractions that can be used to extract some information about individual fractions. The fact that L has no proposed value, however, seems problematic, as it is clearly the base for the L series containing L2, L3, L4, L6. In fact, while Bennett (1950) did not consider L as a self-standing sign, both the standard corpus (Godart et al., 1996) and other authors (Perna, 1990; Facchetti, 1994; Cash and Cash, 2012) treated it as a regular fraction sign. It is clear, then, that no conclusive agreement could be reached between different decipherment attempts.

| Fraction | Bennett (1950) | Stoltenberg (1955) | Was (1971) | (Facchetti, 1994) | Younger (2022) | Cash and Cash (2012) | Schrijver (2014) |
|---|---|---|---|---|---|---|---|
| J | 1/2 | 1/2 | 1/4 | 1/4 | 1/2 | 1/2 | 1/2 |
| E | 1/4 | 1/4 | 1/2 | 1/2 | 1/4 | 1/4 | 1/4 |
| D | 2/3 | 1/5 | | | 1/5? | 1/6 | 1/5 |
| B | 1/12? | 1/6 | 1/6 | 1/10 | 1/3? | 1/5 | 1/6 |
| K | < 1/8 | 1/10 | | | 1/16 | 1/16? | 1/16 |
| L2 | < 1/4 | 1/9 | | | | 3/20? | 1/24 |
| F | 1/8 | 1/8 | 3/8 | 3/8 | 1/8 | 1/8 | 1/8 |
| H | 1/3 | 1/3 | | | 1/6? | 3/10 | 1/3 |
| A | < 1/4 | 1/12 | | | 1/6 | 1/20 | 1/12 |
| L3 | | | | | | 1/10 or 3/40? | 1/36? |
| L4 | | | | | | 3/40 or 1/40? | 1/48? |
| L6 | | | | | | 1/80? | 1/64? |
| X | | | | | | 9/20 | |
| L | | | | | | > 3/20 (17/40?) | |
| Ω | | | | | | | |
| JE | 3/4 | 3/4 | 3/4 | 3/4 | 3/4 | 3/4 | 3/4 |
| DD | 1/6 | 2/5 | | | 2/5? | 1/3 | 2/5 |
| BB | < 2/4 | | 1/12 | 1/12 | 2/3? | 2/5 | 2/6 = 1/3 |
| EF | 3/8 | 3/8 | 7/8 | 7/8 | 3/8 | 3/8 | 3/8 |
| KL2 | < 2/4 | | | | | 17/80? | 5/48 |
| EL2 | 2/4 | | | | | 2/5? | 7/24? |
| JB | < 3/4 | 2/3 | 5/12 | 7/20 | 5/6? | 7/10 | 4/6 |
| JEL2 | < 1 | | | | | 9/10? | 19/24 |
| JL2 | < 3/4 | 11/18 | | | | 13/20? | 13/24 |
| AA | < 2/4 | | | | 2/6? | 1/10 | 1/6 |
| DDDD | | 4/5 | | | 4/5? | 2/3 | 4/5 |
| EB | < 2/4 | | 2/3 | 3/5 | 7/12? | 9/20 | 5/12 |
| EK | < 2/4 | 9/40 | | | 3/16 | 3/16? | 3/16 |
| EL4 | < 2/4 | | | | | 13/40 or 11/40? | 13/48 |
| EL6 | 2/4 | | | | | 21/80? | 17/64? |
| FK | < 3/8 | 7/20 | | | 3/16 | 3/16? | 3/16 |
| HHH | 1 | 1 | | | 1/2? | 9/10 | 1 |
| JA | < 3/4 | 7/12 | | | 4/6? | 11/20 | 7/12 |
| JEB | < 1 | 11/12 | 11/12 | 17/20 | 1+1/12? | 19/20 | 11/12 |
| JF | 5/8 | 5/8 | 5/8 | 5/8 | 5/8 | 5/8 | 5/8 |
| JH | 5/6 | 5/6 | | | 4/6? | 4/5 | 5/6 |
| JK | < 3/4 | 3/5 | | | 9/16 | 9/16? | 9/16 |
| L2L4 | < 2/4 | | | | | 9/40 or 7/40? | 3/48 |

Table 4.2: Proposed value assignments for Linear A fractions and combinations from various authors

## 4.3    Formalizing the problem: constraints and variables

Since no agreement can be found on the numerical values of fractions in Linear A, beside the more obvious cases of J, E and F, it can be beneficial to treat this problem with a new approach, instead of relying on manually compiled tables of relations between signs. The obvious solution for this task is to use a constraint programming approach, which, as described in Section 2.1, allows us to create a set of constraint that are the bare minimum needed to consider a decipherment as possible. Unlike other approaches based on linear programming or integer programming, constraint programming allows the specification of non-linear constraints, as well as the usage of floating points. Furthermore, since we are interested in producing all valid solutions given the constraints, the benefits of algorithms for linear optimization are limited, since they require the usage of an optimization function. In our experiments, we use the programming language for constraint programming Minizinc (Nethercote et al., 2007) and the solver Gecode (Gecode Team, 2005), which allow us to quickly specify constraint-based programs which use floating points in addition to integers. In fact, we can already hypothesize various aspect of the script, which will then be used to formalize our constraints. In particular:

1. The values of J, E and F are known and they are $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ respectively;

2. The combinations of fraction signs in sequence indicate that an addition is being performed between them.Therefore the fraction system must be economical: if a value is represented by a sign, it cannot share the same value with another sign or combination of signs. This is motivated by the idea that writing a value that can be represented by a single sign in any other way is inefficient. However, it is still possible for a value to be represented by two different combinations of two of more signs.

3. Any fraction or sequence of fractions must yield a value that is less than one. This is motivated by the fact that, since the Minoans used both a system for denoting integers and one for fraction values in combinations, it would be pointless to express a value larger than the unit with fractions only, as it is more economical to express it as a combination of integers and fractions.

4. The fractions are written in decreasing order. This fact is derived from the similarities between the integer system and the fraction system, which also transcribes integers in a decreasing order of values. It is also clear that there is an order between fractions in the available material. As an example, while there are only two doubtful attestations of EJ in total (which were excluded from our sample), the sequence JE is the most attested in the entire corpus, appearing 24 times. This would be very improbable unless a standardized order for signs existed.

5. The signs belonging to the L series must be part of some mathematical progression. By applying the previous constraints to this rule, we can then try to establish which kind of relation this can be. In particular, we know that $L2 > L4$ and that the problematic sign L is also greater than L2. This suggests that by increasing the value of $i$ in $L_i$ (which denotes the nth sign in the series) we obtain signs that are decreasing in value. This excludes any additive relation involving positive values, as that would violate the order of values between signs in the L series. Similarly, any multiplication with a positive integer is impossible. An arbitrary subtractive mechanism also seems improbable, especially since the horizontal line added to the basic L shape seems similar to the sign for 10 in Linear A, which would immediately lead to negative values. We are then left with only one possible solution: a divisive progression. What is left to determine, then, is which base to use for this kind of divisive series. The only two seemingly reasonable solutions seem to be considering $L_i = \frac{L}{i}$ or $L_i = \frac{L}{10i}$. While both of these progressions are possible, the base 10 progression is problematic for multiple reasons. First of all, whatever the value of L is, the series would quickly produce very high denominator values. Secondly, the absence of L1, which would correspond to $\frac{L}{10}$ also suggests to use a simpler divisive relation of the form $L_i = \frac{L}{i}$ for the L series.

From these ideas, we can then formalize a series of mathematical constraints for the values of the fractions. First, we define our set of variables as $\mathcal{F}=\{$A, B, D, E, F, H, J, K, L2, L3, L4, L6$\}$ and the set of attested combinations as $\mathcal{S}$. In this set, W and X are excluded, since they appear to be formed by the combinations of BB and AA respectively. Y and $\Omega$ were also excluded, as they appear in few attestations and never in combination with any other sign. This means that they would be bound by very

few constraints and that any attempt at using values obtained by computational means to decipher them would be arbitrary. Furthermore, their inclusion in the computation would increase the number of valid solutions, since they could take almost any possible value, which would in turn complicate any further analysis and dramatically increase the time needed to obtain solutions, to an intractable extent.

Having informally defined the types of constraints that can be applied to the fraction values, we can then formalize them:

$$J = \frac{1}{2} \quad E = \frac{1}{4} \quad F = \frac{1}{8} \tag{4.1}$$

$$\forall x \in \mathcal{F} \quad \forall y \in \mathcal{S} \quad x \neq \sum_{y_i \in y} y_i \tag{4.2}$$

$$\forall x \in \mathcal{F} \quad x < 1 \qquad \forall x \in \mathcal{S} \quad \sum_{x_i \in x} x_i < 1; \quad \forall x, y \in \mathcal{F} \quad x \neq y \tag{4.3}$$

$$J > E > F > K > L2 > L4; J > A; J > H; E > B; L > L2 \tag{4.4}$$

$$L_i = \frac{L}{i} \tag{4.5}$$

As mentioned in section 2.1, constraint programming programs are defined by a set of variables, constraints and the domains for the variables. While we already formalized the first two aspects of the program, what's missing is the domain of our variables, which represents the possible values that they can have. By using constraint number 2, one might be tempted to consider as a possibility all fractions in the range $]0, 1[$. This approach, however, is impossible, since the set of all rational numbers $\mathbb{Q}$ is not only infinite but dense, which means that between any two rational numbers $p, q$ we can construct a third number $r$ such that $p < r < q$ by using the arithmetic mean between the two numbers: $r = \frac{p+q}{2}$. Intuitively, this means that, with constraints involving only inequalities between our variables (beside J,E,F), we would always obtain an infinite number of solutions for our problem, which makes any attempt at computing all valid solutions impossible. Furthermore, since the constraint propagation approach used to solve problems involving floating point values is expensive and needs to try many possible assignments before obtaining all solutions, it is crucial to limit the number of possible values for the variables, without excluding sensible ones.

## 4.4   Formalizing the problem: possible fraction values

Since no formal method can be used in order to determine the set of values to allow for our variables, our approach is to use typological knowledge and examine similar systems of numerical notation for fractions attested around the world and determine which values were used in such systems. In particular, we considered four Egyptian systems, Mesopotamian cuneiform, Greek Alphabetical, Coptic alphabetical, North African Fez/Rumi, the vulgar Arabic system and Indian Grantha. The first finding of this typological investigation is the fact that the lowest attested value for fractions is $\frac{1}{320}$, found in the Grantha system. Furthermore, only two of the examined systems use values that are lower than $\frac{1}{10}$. In summary, the attested fraction values are:

$$\frac{5}{6} \quad \frac{2}{3} \quad \frac{1}{5} \quad \frac{1}{6} \quad \frac{3}{20} \quad \frac{3}{20} \quad \frac{1}{10} \quad \frac{1}{16} \quad \frac{1}{20} \quad \frac{1}{32} \quad \frac{1}{40} \quad \frac{1}{64} \quad \frac{1}{80} \quad \frac{1}{160} \quad \frac{1}{320}$$

However, since Linear A is the system with the highest number of fractions among those we examined and most other writing systems do not encode values lower than $\frac{1}{10}$, it is necessary to consider a wider range of values. In particular, by observing systems of numeric notation in general, including integers, we can observe that all numerical systems in the world use a base 10 system or some multiple of 10, like base 20 and base 60. A base two subsystem is used in the Eye of Horus Egyptian system, while Grantha uses a base 20. For this reason, we extended the typologically attested values with the fractions having numerator 1 and the following denominators:

- All multiples of 10 up to 100 and all multiples of 100 up to 300;

- All products of 10 with numbers that are multiples of 2 and 5, up to 320;

- Sexagesimal values up to 240;

- Multiples of 12 up to 120;

- 7 and 9, whose odd denominator has been hypothesized to explain the fact that D is only ever found in combination with itself.

Finally, by applying the previous additions, we obtain the final domains of the variables, which include all the values that are considered possible for the fractions (since $J = \frac{1}{2}$, $E = \frac{1}{4}$, $F = \frac{1}{8}$ these are excluded from the possible fraction values):

$$\frac{5}{6} \quad \frac{2}{3} \quad \frac{1}{3} \quad \frac{1}{5} \quad \frac{1}{6} \quad \frac{3}{20} \quad \frac{1}{7} \quad \frac{1}{9} \quad \frac{1}{10} \quad \frac{1}{12} \quad \frac{1}{15} \quad \frac{1}{16} \quad \frac{1}{20} \quad \frac{1}{24} \quad \frac{1}{30} \quad \frac{1}{32} \quad \frac{1}{36} \quad \frac{1}{40}$$

$$\frac{1}{48} \quad \frac{1}{50} \quad \frac{1}{60} \quad \frac{1}{64} \quad \frac{1}{70} \quad \frac{1}{72} \quad \frac{1}{80} \quad \frac{1}{84} \quad \frac{1}{90} \quad \frac{1}{96} \quad \frac{1}{108} \quad \frac{1}{100} \quad \frac{1}{120} \quad \frac{1}{160} \quad \frac{1}{200} \quad \frac{1}{240} \quad \frac{1}{300} \quad \frac{1}{320}$$

This set of values, however, might not be sufficient to capture possible values for the L series, since this series is divisive in nature and will produce lower values than the rest of the fractions. Additionally, the only constraint that applies to the L series with respect to other fractions is the fact that L4 < L2 < K < F = $\frac{1}{8}$. This suggests that, if the numerators of the L series are all one, the value of L2 must be smaller than $\frac{1}{9}$, which in turn implies that this series contains many fractions with low values, in proportions not seen otherwise in the literature. For this reason, the domain for the L series is defined as the set of all fractions with numerator one and denominator between 3 and 320, excluding denominators of the fractions with known values (2 for J, 4 for E, 8 for F).

## 4.5   L is a self-standing sign?

As mentioned in the previous sections, the L series constitutes a peculiar characteristic of the Minoan fraction system, but the status of the basic sign L is debated and the fact that we only have one safe attestation of it suggests that this might not be a self-standing sign. This doubt stems from multiple factors, including the fact that L is the base for the whole L series, but it is attested only once, despite being the higher-valued fraction in the series due to the constraint concerning its divisive nature, as well as the fact that it appears before L2. Bennett notes that within the Minoan fraction system, more frequently used fractions should tend to encode higher values overall, and vice versa. This idea, which can be defended even intuitively, also has a basis that is more grounded in the property of such fractions when used in an additive cumulative system. This idea has to do with the denominators resulting from the combinations of fractions. In order to sum two fractions, one has to compute the lowest common denominator between the two, which corresponds to the least common multiple of the two denominators in the sum. With higher values fractions, which have a lower denominator, we obtain many instances

where a common factor exists between the two denominators, resulting in lower values for the lowest common denominator, which entails a resulting value which has a lower denominator. In fact, if we have two fractions $\frac{x}{n}$ and $\frac{y}{m}$, where $n \leq m$, the probability that $n$ is a divisor of $m$ is exactly $\frac{1}{n}$, which would result in $m$ being the least common multiple between the two. This means, for example, that one in two integers is a divisor of two, while one in three is a divisor of three and so on. Even if no common factors can be found between the two denominators, the combinations of lower denominators always tend to produce a lower common multiple. This means that, in a sense, lower-valued fractions are more useful in a cumulative additive fraction system, since they can be combined with other fractions without producing very high denominators, which are impractical and in turn do not combine gracefully with other fractions. Another, more informal way to think of this concerns the practical application of these fractions to commodities. Using modern measuring units, we can intuitively say that it rarely makes sense to combine $\frac{1}{2}$ of a kilogram with $\frac{1}{100}$ of it, as the second portion of the addition almost disappears in terms of orders of magnitude beside the first one and even a weight scale might not accurately detect such a small variation in weight.

In order to assess whether L is a self-standing sign, then, we first used all the possible solutions obtained from the application of the constraints to the fractions. This first simulation yields 2,172,836 solutions. Then, for each solution, we sorted the signs by number of attestations and computed, for each, the number of fractions that are their divisors (whose denominator is a multiple of theirs). By manually observing some solutions, it is already clear that, while other fractions such as D, B, F and A can yield anomalous ranks in terms of number of divisors, L is always an outlier. In order to measure the effect of this observation, we used the Spearman Rank's Correlation Coefficient (s) in order to measure the correlation between the ranking resulting from sorting signs by frequency ($f$) and by number of common divisors ($d$). This measure is defined as:

$$g(f, d) = 1 - s(f, d) \tag{4.6}$$

In order to assess the impact of L over the measure, we applied it to another simulation obtained by excluding L as a variable from our system. This simulation yielded 3,794,740 solutions. In simulation 1, where L was still present, the lowest possible value for $g(f, d)$ is 0.14284. When we apply the same metric to the second simulation excluding L we

obtain 394,905 possible assignments with $g(f, d) < 0.14284$. Moreover, the lowest value attested for $g$ is 0.00881, which is 16 times smaller than the one from simulation 1.

Although this result is already a good indication that L might not be a self-standing sign, another approach can be used, in order to more accurately assess how much of an outlier L really is. In order to produce this result, we need to focus on the L series in isolation. Due to the divisive nature of the series, we can infer some information about the number of divisors within the series independently of the specific values of the fractions. In particular, we define $d_{Li}$ as the number of fractions distinct from Li which are its divisors. Considering $d_{L2}$, then, we already know that $d_L \geq d_{L2} + 2$, since $d_L$ also includes L2 and L3, which are not counted in $d_{L2}$. Due to the five constraints over the fraction values, only five signs could be divisors of L or L2: A, B, H, W and X. Additionally, it seems improbable that these signs are divisors of L, since they do not feature the same semi-circle shape present in the L series. These values would be equivalent to the hypothetical signs L8, L9, L10 or so on, but they are clearly not a part of the same progression.

Once established that signs in the L series can be treated in isolation, independent of their values, with respect to their divisors, we can then apply Pearson's $\chi^2$ goodness-of-fit statistical test, to measure whether the difference between the observed and expected frequencies of L is indeed statistically significant, under the hypothesis that the frequency of a sign is directly proportional with the number of its divisors. We use the frequencies of L and L2, as well as the frequencies of L2 and L6 for this analysis. Considering the number of divisors, the expected frequency of L is $\frac{2}{3}$, while the expected frequency of L2 is $\frac{1}{3}$. Our evaluation then uses the number of attestations of two pairs of signs, namely L and L2, L6 and L2. Only the number of attestations and relative frequencies of the two pairs of signs are considered, namely L and L2, L6 and L2 for each of the two experiment independently. As a threshold for statistical significance, we chose a p-value of 0.01 in order to avoid any false positive that might result from the relatively low number of attestations for these fractions. In Table 4.3 we show the p-values obtained by the application of the statistical test both to the real number of attestations and to hypothetically increased values of attestations for both L and L6. The results of this evaluation show that, while L6 is also considered to have a low number of attestation

| | Attestations of L | P-value | Attestations of L6 | P-value |
|---|---|---|---|---|
| Real attestations | 1 | $4 * 10^{-12}$ (***) | 2 | 0.029 (-) |
| Hypothetical attestations | 2 | $2 * 10^{-11}$ (***) | 3 | 0.068 (-) |
| | 5 | $2 * 10^{-9}$ (***) | 5 | 0.254(-) |
| | 10 | $7 * 10^{-7}$ (***) | | |
| | 15 | $4 * 10^{-5}$ (***) | | |
| | 20 | $8 * 10^{-5}$ (***) | | |
| | 26 | 0.010 (*) | | |
| | 30 | 0.037 (-) | | |
| | 35 | 0.124 (-) | | |

Table 4.3: P-values for Pearson's $\chi^2$ goodness of fit statistical test comparing expected frequency and real frequency for L1 and L6, for both real and hypothetical numbers of attestations. Between parenthesis, the significance values:

(***) $p < 0.0001$;

(**) $0.0001 \leq 0.001$;

(*) $0.001 \leq p < 0.01$;

(-) $p \geq 0.01$, no statistical significance.

when considering its number of divisors, the resulting probability value is not so low as to be considered statistically significant ($p = 0.02 > 0.01$). Furthermore, if some new attestations of L6 were to be found, this probability number would rapidly increase, with a p-value of 0.25 with only three new attestations found. The same cannot be said for L, which has a very low p-value for the real number of attestations, and which would require 25 new attestations to be found in order to be barely above the threshold for statistical significance. While the possibility of finding some new attestations for L must be considered, it is very improbable that the real frequency of L is so different from the one observed in our sample.

With this new evidence, it appears clear that L is an outlier with respect to the Minoan fraction system and that the only attestation of L in the sequence L L2 (see Figure 4.3) must be considered an anomaly. For this reason, we opted to exclude L from

Figure 4.3: The only safe attestation of L in the sequence LL2. Drawing from the SigLA database (Salgarella and Castellan, 2020).

the set of variables in our experiments.

## 4.6    Evaluating the solutions

Due to the large number of possible assignments for the values of the fractions (approximately 3.8 million in simulation 2, the one excluding L) it is impossible to proceed with any manual observation over the fraction values without first applying some sort of heuristic in order to reduce the number of solutions to a more manageable number. For this reason, we propose 4 different metrics, which aim to capture different desirable aspects of any cumulative additive system of notation for fractions.

The first measure is the correlation between the frequency of a sign and the number of divisors found in the system. This metric has already been defined in equation 4.6. The rationale behind it is the idea that frequent signs should combine in a simple way with other signs, therefore they should have relatively many divisors within the fraction system.

The second measure is a measure of attested ambiguity. While our second constraint prevents a fraction value from being expressed both using a single sign and with another sign or combination of signs, no constraint prevents solutions where two attested combinations express the same value. While it is possible for a fraction system to be ambiguous, we expect this kind ambiguity to be relatively rare, since a standard way to express each sum should emerge in any cumulative additive system for fractions such as the Linear A one. Given $\mathcal{S}$, which is defined as the set of sequences of signs that are

attested in inscriptions, this metric is defined as:

$$a = \frac{|\{(x,y) \in \mathcal{S} \times \mathcal{S} \ : \ x \neq y \wedge \sum_{x_i \in x} x_i = \sum_{y_i \in y} y\}|}{|\{(x,y) \in \mathcal{S} \times \mathcal{S} \ : \ x \neq y\}|} \qquad (4.7)$$

The formula counts the number of attested combinations that share the same value and then normalizes the result by dividing this number by the total number of distinct pairs of combinations, in order to obtain a value between 0 and 1.

The third metric in our analysis has to do with the typological analysis of section 4.4. During this analysis, a list of all the values attested in other systems of notations for fractions was used to construct the domains of our variables. In addition to this usage of our typological analysis, however, it is also possible to consider the fact that we expect most values that were expressed in other fraction systems to be also present in Linear A. For this reason, the third metric measures how many values attested typologically are assigned to a variable in each solution. Given the set of fractions $\mathcal{F}$ and the set of typologically attested signs $(T)$, we measure:

$$t = \frac{|\{t \in \mathcal{T} \ : \ \exists f \in \mathcal{F} \ t = s\}|}{|\mathcal{T}|} \qquad (4.8)$$

Like the other metrics, $t$ is normalized from 0 to 1 and it measures the relative amount of typologically attested signs that are assigned in the Linear A system according to each solution.

The final metric, which we called "optimality", is involved with the overall evaluation of the expressive power and ambiguity of the fraction system. With respect to expressive power, it measures how well the system is able to represent values that are in its own fractional base, which is the range of values between $\frac{1}{2}$ and $\frac{m_d-1}{m_d}$, where $m_d$ is the highest denominator in the system. As an example, if $m_d = 3$ the fractional base is $\frac{1}{2}, \frac{1}{3}, \frac{2}{3}$. In order to make this metric computationally tractable and realistic, we use all possible combinations of four or less fractions, since we do not have any evidence for combinations of more than four signs. However, this approach also becomes problematic for very high denominators, since many values would be missing when the denominator is relatively high due to the restriction of using only sequences of four or less signs. For this reason, we weigh the metric so that a value missing from a low value denominator is deemed more severe than one missing with a high denominator. In order to do this, we

| Denominators | Numerators | | | | Incompleteness |
|:---:|:---:|:---:|:---:|:---:|:---:|
| | 1 | 2 | 3 | 4 | |
| 2 | R | | | | $0 * \frac{1}{1}$ |
| 3 | R | U | | | $1 * \frac{1}{2}$ |
| 4 | R | R | R | | $0 * \frac{1}{3}$ |
| 5 | R | R | R | U | $1 * \frac{1}{4}$ |
| Total | $(0 + \frac{1}{2} + 0 + \frac{1}{4})/4 = 18.6\%$ | | | | |

Table 4.4: Example application of the optimality component for expressive power $o_p$ for a small sample, with $m_d = 5$. R=Represented, U=Unrepresented.

normalize the value of the metric for each denominator, so that each one has the same weight. First, we define the set of all possible combinations of four or less signs $\mathcal{C}$. For convenience, we can also define a series of sets $\mathcal{A}_i$ such that $\mathcal{A}_i$ contains all fractions with denominator $i$:

$$\mathcal{A}_i = \left\{ \frac{x}{i} \; : \; i \leq m_d \wedge x < y - 1 \right\} \tag{4.9}$$

Finally, we define the component of the metric tasked with the expressive power of the system as follows:

$$o_p = \left( \sum_{i=2}^{m_d} \frac{|\{a \in A_i \; : \; \nexists c \in \mathcal{C} \quad \sum c = a\}|}{i - 1} \right) \Big/ (m_d - 1) \tag{4.10}$$

This formula measures the number of values within the fractional base that cannot be expressed with combinations of at most four fractions. The values are normalized so that every denominator shares the same weight. In order to clarify this concept further, we show in Table 4.4 a small example of how this metric is computed.

Another component of the optimality metric regards the ambiguity of the system. Using all possible combinations of four or less signs as discussed previously, it is possible to compute the number of values that can be represented by two or more combinations

| Denominators | Numerators | | | | Incompleteness |
|:---:|:---:|:---:|:---:|:---:|:---:|
| | 1 | 2 | 3 | 4 | |
| 2 | U | | | | $0 * \frac{1}{1}$ |
| 3 | U | | | | $0 * \frac{1}{2}$ |
| 4 | U | U | U | | $0 * \frac{1}{3}$ |
| 5 | U | A | A | | $2 * \frac{1}{4}$ |
| Total | $(0 + 0 + 0 + \frac{2}{4})/4 = 12.5\%$ | | | | |

Table 4.5: Example application of the optimality component for ambiguity $o_a$ for a small sample, with $m_d = 5$. U=unambiguous, A=ambiguous.

of signs. We also use the same adjustment to the metric, in order to reduce the impact of fractions with lower denominator in the total metric. This adjustment is motivated by the fact that the fractions with a lower denominators result from more combinations of signs, so they are less severe with respect to the problem of ambiguity. In fact, every even denominator up to 8 can produce $\frac{1}{2}$ with at most four combinations of itself, and every denominator divisible by three up to 12 can do the same for 3, and so on. Similarly to the previous definition of $o_p$, we define the ambiguity component of the optimality measure as:

$$o_a = \left( \sum_{i=2}^{m_d} \frac{|\{a \in \mathcal{A}_i \ : \ \exists x, y \in \mathcal{C} \quad x \neq y \wedge \sum_{x_i \in x} x_i = \sum_{y_i \in y} y_i\}|}{y - 1} \right) \Big/ (m_d - 1) \quad (4.11)$$

We also provide an example of a possible application of $o_a$ in Table 4.5. In order to be coherent with the previous example, notice that some of the cells of this table are grey, since those values are marked as unrepresented in table 4.4.

Having defined the expressive power and ambiguity components of the optimality measure, we simply combine them to produce the overall optimality measure by summing over them:

$$o = o_p + o_a \quad (4.12)$$

One interesting aspect of the optimality metric is that, unlike the other metrics described in this section, it is not normalized to produce a value between 0 and 1, instead producing values between 0 and 2. The reason behind this choice is not related to any mathematical property but it has to do with practical constraints that prevent the application of this metric to a large number of solutions. In fact, this metric is very expensive to compute as it needs to produce, for each solution, a large number of comparisons between values. In particular, we need to compute all possible combinations of four or less signs, which is given by:

$$\sum_{i=1}^{4} \binom{12}{i} = 793 \tag{4.13}$$

Once all these values have been computed, then, various other procedures need to be applied in order to obtain the final measure. This procedure needs to be applied to all the 3.8 million solutions in simulation 2. For this reason, the application of the optimality metric can happen only on a smaller subset of the solution, so our first step was to apply the sum of the first three metrics to all solutions first, and then restrict them further using the optimality metric. Therefore, the optimality metric does not need any normalization as it is considered only in isolation.

## 4.7   Sign Assignments

After the application of the metrics for the correlation between frequency and divisors, ambiguity, typology, we selected the lower valued solutions and restricted any further analysis to the first 5000. Then, we were able to apply the optimality metric $o$ to this restricted set of solutions. We then proceeded to manually examine the solutions and the assignments that characterized them, in order to produce an iterative process where we could assign some fraction values each time, in turn reducing the number of solutions left for the other fractions.

The first step of this process was the observation that in terms of optimality only two solutions had the lowest value of 0.67094. Interestingly, these solutions always presented the same assignments for seven fractions:

$$B = \frac{1}{5} \quad D = \frac{1}{6} \quad K = \frac{1}{10} \quad L2 = \frac{1}{20} \quad L3 = \frac{1}{30} \quad L4 = \frac{1}{40} \quad L6 = \frac{1}{60}$$

Two other fractions also have a very restricted set of possible values, which is shared between the two. Those are H and A, which can only be $\frac{1}{32}$ and $\frac{1}{36}$. While these patterns are interesting, they are not sufficient to claim that any of these assignments is correct, since there might be some other possible solutions with a slightly lower value of optimality that present different and sensible values for these fractions. For this reason, it is useful to examine the pervasiveness of these assignment across a wider range of solutions that present a high optimality, in the form of a low value for $o$.

Examining the most optimal solutions, we noticed that assignments for K and the L series, which suggest a decimal-sexagesimal nature for these fractions, appear in the 48 most optimal solutions. In total, they appear in 100 of the 5000 total solutions, but they clearly show a high value of optimality according to our metric. The decimal-sexagesimal system that stems from these metrics would entail a value of $\frac{3}{20}$ for K L2, which is frequently attested in our sample. This result seems coherent with the idea that frequently attested combinations of signs should yield values that are relatively simple, with a denominator which is one of the two involved in the addition and relatively low numerator values. This is precisely what we observe for K L2 and the existence of a decimal-sexagesimal subsystem within the Minoan fractions also seems to produce results that are more expressive than alternative assignments, where the system is able to succinctly express many values within its own fractional base. The fact that the L series combines with J and E as well is also a good indicator that the hypothesized assignments are correct. Since there is good evidence for a decimal-sexagesimal series for the L series and K, we assign the following values:

$$K = \frac{1}{10} \quad L2 = \frac{1}{20} \quad L3 = \frac{1}{30} \quad L4 = \frac{1}{40} \quad L6 = \frac{1}{60}$$

Having assigned the values for the L series and K, it is now possible to observe what characterizes the remaining 100 possible solutions. In particular, 86 out of 100 of those have the following assignments:

$$B = \frac{1}{5} \quad D = \frac{1}{6}$$

In addition to being present in the vast majority of the remaining solutions, these assignments are also consistently used in the 28 best solutions in terms of optimality ($o$ in the range 0.67094-72676).

| Den. 20 Fraction | Den. 10 fraction | $B = \frac{1}{5}$, $D = \frac{1}{6}$ | $B = \frac{1}{6}$, $D = \frac{1}{5}$ |
|:---:|:---:|:---:|:---:|
| $\frac{1}{20}$ | | L2 | L2 |
| $\frac{2}{20}$ | $\frac{1}{10}$ | K | K |
| $\frac{3}{20}$ | | K L2 | K L2 |
| $\frac{4}{20}$ | $\frac{2}{10}$ | B | D |
| $\frac{5}{20}$ | | E | E |
| $\frac{6}{20}$ | $\frac{3}{10}$ | E L2 | E L2 |
| $\frac{7}{20}$ | | E K | E K |
| $\frac{8}{20}$ | $\frac{4}{10}$ | B B | D D |
| $\frac{9}{20}$ | | E B | |
| $\frac{10}{20}$ | $\frac{5}{10}$ | J | J |
| $\frac{11}{20}$ | | J L2 | J L2 |
| $\frac{12}{20}$ | $\frac{6}{10}$ | J K B | J K |
| $\frac{13}{20}$ | | | |
| $\frac{14}{20}$ | $\frac{7}{10}$ | J B | |
| $\frac{15}{20}$ | | JE | JE |
| $\frac{16}{20}$ | $\frac{8}{10}$ | J E L2 | J E L2 |
| $\frac{17}{20}$ | | | |
| $\frac{18}{20}$ | $\frac{9}{10}$ | | |
| $\frac{19}{20}$ | | J E B | |

Table 4.6: Attested combinations for signs having denominators that are multiples of 20. Values with denominator 10 shown where available for readability.

There are only three alternative assignments that are possible for D and B:

- $B = \frac{1}{6}$, $D = \frac{1}{5}$ (12 solutions, $o \geq 0.71724$);

- $B = \frac{1}{5}$, $D = \frac{1}{12}$ (1 solution $o = 0.79323$);

- $B = \frac{1}{9}$, $D = \frac{1}{6}$ (1 solution, $o = 0.80510$);

These alternatives are not as optimal per our metric, which reflects that the best assignment of $B = \frac{1}{5}$, $D = \frac{1}{6}$ presents structural advantages in combination with the usage of a decimal-sexagesimal subsystem such as the one used by K and the L series. The assignments for B and D, in fact, produce a system where most fraction values with denominators 10, 20 and 60 respectively are attested, which would only make sense in the context of the previous sign assignments of L and K. In order to illustrate the value of this procedure, the represented values for the denominator 20 and 10 fractions, obtained by using the two most optimal assignments are shown in Table 4.6. In both situations with denominators 10 and 20, we see an advantage when using $B = \frac{1}{5}$, $D = \frac{1}{6}$.

| Fractional base | $B = \frac{1}{5}$, $D = \frac{1}{6}$ | $B = \frac{1}{6}$, $D = \frac{1}{5}$ |
|:---:|:---:|:---:|
| 10 | **1** | 2 |
| 20 | **3** | 6 |
| 60 | **35** | 36 |

Table 4.7: Number of signs not attested with the two most optimal assignments for D and B. In bold, the lower (better) values.

A more synthetic evaluation in the form of the number of missing values, which are not attested as signs or combinations of signs with the two possible assignments for D and B is shown in Table 4.7. These results show that the assignments of $B = \frac{1}{5}$, $D = \frac{1}{6}$ produce less unattested values across all fractional bases, even if the degree to which they are represented with attested combinations is of course dependent on the denominator.

Another factor supporting the assignment of D as $\frac{1}{6}$ is the fact that $D$ is only found in combination with itself and, crucially, it is only repeated two and four times (DD and DDDD). This would perfectly make sense if D was $\frac{1}{6}$, since that would imply that the

missing DDD $= \frac{3}{6} = \frac{1}{2} = $ J. On the other hand, DD and DDDD would be equal to $\frac{1}{3}$ and $\frac{2}{3}$, two values that appear useful in a fraction system, especially when a decimal-sexagesimal sub-base is used within it. While a comparison between the Linear A fractions and Linear B units of measure also seems to support the assignment of D as $\frac{1}{6}$, this consideration is left for a post-hoc evaluation of our assignments. Due to the aforementioned factors, we assigned $B = \frac{1}{5}$, $D = \frac{1}{6}$ and further restricted the remaining solutions.

Next, we examine the values of H and A. In the remaining solutions, we observe the following possibilities for H and A:

$$
\text{H:} \quad \frac{1}{16} \quad \frac{1}{24} \quad \frac{1}{32} \quad \frac{1}{36} \quad \frac{1}{48} \quad \frac{1}{64} \quad \frac{1}{72} \quad \frac{1}{84}
$$
$$
\text{A:} \quad \frac{1}{24} \quad \frac{1}{32} \quad \frac{1}{36} \quad \frac{1}{48} \quad \frac{1}{64} \quad \frac{1}{72} \quad \frac{1}{84}
$$

Within these possible assignments, optimality appears to be less helpful than in cases involving other signs. In particular, the assignments with the lower values for the metric involve $H = \frac{1}{36}$, $A = \frac{1}{32}$ and $H = \frac{1}{32}$, $A = \frac{1}{36}$ respectively. These values are not as consistent as other assignments with respect to the already established values. Furthermore, some other evidence, which is not involved in the optimality of the assignments, needs to be considered.



$$
\begin{array}{cccc}
\text{J} & \text{E} & \text{F} & \text{H} \\
\frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \frac{1}{16}?
\end{array}
$$

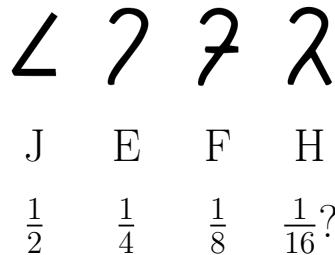Figure 4.4: The graphical progression for the series involving J,E,F,H.

With respect to H, some paleographic considerations need to be carefully examined. In particular, the sign contains the same crooked shape which is part of J, E and F. Given the fact that these signs are characterized by a denominator which is a power of 2, it would make sense for H to be a part of this progression and have value $\frac{1}{16}$. This

would result in a progression which is both graphical and numerical in nature, as shown in Figure 4.4.

In addition to these considerations, it must be noted that $\frac{1}{16}$ is the only value that is only possible for H and not for A. Since we already discussed how fractions with a lower denominator generally combine more seamlessly with other fractions, this seems to further support our idea that H= $\frac{1}{16}$.

Other considerations that support this more tentative assignment of H involve the value of A. In particular, if H=$\frac{1}{16}$, the assignment of A=$\frac{1}{24}$ presents some interesting advantages. In particular, if we consider that the sign X appears to be the result of a combination of two A signs, AA would be $\frac{2}{24} = \frac{1}{12}$ and this value would be consistent with the decimal-sexagesimal sub-component of the fraction system, since $\frac{1}{12} = \frac{5}{60}$. The values proposed for H and A are more tentative than the other ones resulting from considerations that are purely concerned with their expressive power in the fraction system and their ambiguity, but they still present the following advantages:

- They have the lowest possible denominators for H and A, which results in a system where signs are more easily used in combinations;

- With these assignments, all denominators in the system are divisors of 240, which fits with a system of fractions involving a decimal-sexagesimal component, since $240 = 4 * 6 * 10$.

In conclusion, by using metrics designed to evaluate a cumulative, additive fraction system such as the one in Linear A, we managed to assign 7 values, while two other assignments are more tentative. These assignments result in the following system:

$$B = \frac{1}{5} \quad D = \frac{1}{6} \quad K = \frac{1}{10} \quad L2 = \frac{1}{20} \quad L3 = \frac{1}{30} \quad L4 = \frac{1}{40} \quad L6 = \frac{1}{60} \quad A = \frac{1}{24}? \quad H = \frac{1}{16}?$$

## 4.8 Discussion of the results

As discussed in the previous section, our sign assignments are based on a series of premises for the fractions in Linear A:

- Values represented in the system should have a significant overlap with values attested in other systems for fractional notations.

- The system should show signs of standardization, meaning that the presence of two or more different combinations to express a single value is possible, but it should be kept to a minimum.

- The system should be able to represent a wide range of values within its fractional base with short sequences composed of only a handful of signs.

Given these premises, the sign assignments discussed in the previous sections for B, D, K, L2, L3, L4, L6 are the most optimal, while the assignments for H and A are more tentative, as they involve considerations that do not stem directly from the property of the system itself.

While no author proposed our decimal-sexagesimal assignments for the L series, Schrijver (2014) discussed this possibility, but instead assigned the value of $\frac{1}{24}$ to L2 due to his reading of tablet HT 123b. This tablet, however, contains scribal mistakes and doubtful readings (Montecchi, 2009), so it was not included in our sample per our inclusion criteria. Interestingly, the reasoning behind the first assignment, later revised, is the idea that the horizontal line, which is used in conjunction with the basic shape to produce the fraction signs in the L series, might be related to the integer "10" in Linear A. The repetition of this horizontal line, then, would denote a division, so basic shape + two lines would mean "L" divided by 20 and so on.

Regarding the proposed assignments $D = \frac{1}{6}$, $B = \frac{1}{5}$, they were already proposed in Cash and Cash (2012) and Montecchi (2013). In the latter, the possible proportion $B : 4 = J : 10$ from tablet KH 7a was used to infer the value of B. With regards to the value of D, some observations were used to derive its value. First, the fact that D is attested only in combination with itself suggests that it must have an odd value, which is not particularly useful when used in combination with other signs. Furthermore, the presence of DD and DDDD in the sample, combined with the absence of DDD, was used as further evidence that the value of this fraction is $\frac{1}{3}$. This is motivated by the fact that while $\frac{3}{5}$ is not a problematic value, $\frac{3}{6}$ is already represented in the system by J. In addition to these factors, which were used in the literature to motivate the value of D, one must consider the fact that DDDD is the only four sign combination in the sample and it is attested only once. If $D = \frac{1}{5}$, then $DDDD = \frac{4}{5}$ which is already expressed by J E L2 four times in the sample. Instead, it is more probable that D was used four

times to express a value which could not be easily transcribed otherwise by using other combinations of fractions, such as $\frac{2}{3}$.

| Linear A | | | Linear B | | |
|---|---|---|---|---|---|
| Fraction | Shape | Value | Fraction | Shape | Value |
| DD | ⸮ | $\frac{1}{3}$ | *117/M | ⸮ | $\frac{1}{3}$ of LANA |
| K | T | $\frac{1}{10}$ | *112/T | T | $\frac{1}{10}$ of highest dry unit |
| Ln | ◖◀◁ | $\frac{1}{60}$ | *111/V | ◀◁▷Ɛ | $\frac{1}{60}$ of highest dry unit |
| X(=AA?) | ♯ | $\frac{1}{12}$ | *116/N | ♯ | $\frac{1}{12}$ of LANA |

Table 4.8: Shapes and values for some Linear A fractions when compared with Linear B measuring units.

Other considerations involve the relation between the signs DD, X, K and the L series when they are compared with the units of measurements found in Linear B. These measures, just like our modern counterparts, used fractions of the main unit for a given commodity when they were more convenient, and these values were represented using specific logograms whose values are known. The shapes and values for both systems are shown in Table 4.8. From the table, we can observe a few similarities between their shapes and values:

- The shape of DD is identical to *117/M, used in Linear B to denote $\frac{1}{3}$ of the the largest weight measure for wool, named *145/LANA. This value would be shared between the two shapes.

- The shape of K is identical to *112/T, used in Linear B to denote $\frac{1}{10}$ of the highest measure of capacity for dry commodities. This value would be shared between the two shapes.

- The basic shape from the L series is similar to *111/V, used in Linear B to denote $\frac{1}{60}$ of the highest measure of capacity for dry commodities. This value of $\frac{1}{60}$ is shared with the lower valued member of the L series, L6.

- The shape of X is identical to the one used in *116/N, which denotes $\frac{1}{12}$ of the largest weight measure for wool. Since X is probably obtained by combining A two times to obtain AA= $\frac{2}{24} = \frac{1}{12}$, they would also share the same value.

While these strong similarities cannot be used a priori to speculate on the values of the fractions in Linear A, they seem to match our decipherment which derives from properties of the fraction system itself for most signs, except for the more tentative assignments of A and H. This result is comforting, as Linear B itself was largely adapted from Linear A and the two systems used the same notation for integers, so the fact that signs with a similar or identical shape share in some way a similar value is predictable and, if anything, strengthens the results of our decipherment. In Linear A, the combination of a logogram and integers and/or fractions was probably used to denote implicitly a multiple or fraction of a unit of measure. The fact that the Mycenaeans adapted these fractional values to directly define units of measurements, then, is just a further development, where instead of using a single unit of measurement for each commodity, multiple units with different sizes are used.

## 4.9    Conclusions

The results of our proposal for the decipherment of the Linear A fractions shows that a synergistic approach involving multiple disciplines, that also considers the specific requirement of any scientific enquiry on undeciphered writing systems can produce meaningful results which advance the state-of-the-art. In particular, these results show how using well known computational techniques can be beneficial to at least some open questions in the field.

From a computational standpoint, the Linear A fractions and any attempt at deciphering systems of mathematical notation constitute a very peculiar class of problems, since they generally have well defined rules and and the values transcribed by signs are numerical in nature. These characteristics suggest the usage of very strict, formal methods, which operate on numerical values. Its application guarantees that even complex interactions between the variables do not violate any of the requirements, providing a solution that is correct, as far as the specifications go. The time save obtained by using

such a system cannot be overstated, as it can provide scholars with all possible solutions, without requiring a manual check for the constraints of the problem for each sign tentative assignment. Constraint programming, then, is a perfect fit to restrict the number of possible solutions as much as possible.

Despite its great potential, Constraint programming also presents a great limit, which is shared by most computational approaches. Namely, it cannot operate over infinite sets of values, so the choice of the starting values is really important, as it is a trade-off between the time needed to compute solutions and the risk of excluding values that are perfectly valid for the system of numerical notation. This situation requires careful consideration, and the help of experts and prior information about similar systems of notations can be used, in order to produce a set of values that does not exclude plausible ones, but that also limits the time needed to obtain solutions.

While Constraint Programming is a useful tool, any attempt at decipherment for systems of numeric notation is under-constrained, meaning that the available constraints are not sufficiently strict to produce a single possible solution. In fact, problems with a single possible solution can generally be solved manually by humans. Therefore, the definition of specific hard requirements for the solutions is not sufficient to produce meaningful results. In our case, after the application of our constraint-based model, we obtained almost four million solutions for this problem. The next step, then, is to define meaningful measures, which can help in reducing the number of possibilities further and define a more manageable set of solutions, which can then be tackled with an iterative process until almost all signs are deciphered. These measures need to be based on ideas derived from experts in the field, since not all measures can be thought of from a purely mathematical point of view. By applying these metrics, followed by an iterative process where signs were assigned sequentially, it was possible to provide an almost complete decipherment for Linear A fractions.

While the considerations on the process we used to provide a decipherment proposal for the Linear A fractions is one aspect of this study, the main contribution is surely the fact that it advances the state-of-the-art on this specific subject. In fact, we were able to deduce most of the values in this fraction system by considering the specifics of the system itself in relation to other such systems and, crucially, considering its cumulative-additive

nature. In particular, the safer proposed values of B, D, K, L2, L4, L6 in addition to the known values of J, E, F produce a system which has some desirable characteristics for any system of numeric notation:

- Low ambiguity;

- High representative power;

- Typological consistency.

In addition to this, paleographic considerations can be used to tentatively propose more values for A and H, by producing a second sub-base in the signs J, E, F. By comparing the resulting system to Linear B, finally, it is possible to see that similarities between signs are correlated with the usage of a similar value for Linear A as the fraction of the base measuring unit used in Linear B. This suggests an adaptation of the Linear A fraction signs into Linear B units of measurement.

# Chapter 5

# Preliminary experiments on the Cypriot Greek Syllabary

In this Chapter, I describe the preliminary experiments that led to the development of a deep learning model that aims to learn good quality vector representations for signs of undeciphered scripts, with the goal of applying this technique to Cypro-Minoan. In particular, the goal is to create a system that is able to investigate whether two glyphs are different, distinct, graphemes or whether they are the same grapheme, in other words allographs, sharing the same phonetic value.

Due to the peculiar nature of the problem, this model has very specific requirements. First of all, since no ground truth is available for Cypro-Minoan, the system needs to use an unsupervised approach, where the only available information stems from the raw data itself. In this kind of setting the model needs to learn vector representations from signs, without using any prior information, allowing it to categorize the data without favouring any interpretation of the sign inventory, producing an unbiased evaluation.

As a second requirement, the system needs to consider the shape of signs, since the absence of completely agreed upon labels does not allow them to be represented in a symbolic way. The aforementioned requirements clearly suggest the usage of models from the field of computer vision, where unsupervised deep learning models can be applied to images, in order to cluster them in different categories. When signs of an undeciphered writing system are the subjects of the analysis, however, another important aspect needs

to be considered. Since these signs encode a language, they form words and sequences of words, so the prominence of signs in specific sequences must be considered and it is one of the crucial aspects that the experts consider when attempting a rationalization of signs in an undeciphered writing system. In fact, it was thanks to observations about the position of signs in words that Kober was able to first hypothesize the inflectional nature of Linear B first, allowing Ventris to tentatively assign the phonetic value "a" to a grapheme, by observing its prominence at the beginning of words, a characteristic of Greek (see Section 1.1.3).

Having established the requirements for a model that aims to investigate the signs of Cypro-Minoan, however, one issue remains unsolved. In particular, due to the unsupervised nature of the model, it is not possible to evaluate its performance directly on Cypro-Minoan signs themselves, as we would have no way of validating the capabilities of the model. The most sensible course of action, then, is to evaluate the performance of the model on another ancient script, which should be as similar as possible to Cypro-Minoan.

For this reason, our approach involves the usage of the Cypriot Greek Syllabary, which is the ideal candidate for our preliminary evaluation, since:

- It is deciphered and we know it encodes a dialect of ancient Greek, so we can evaluate the performance of the model with certainty;

- Like Cypro-Minoan, it is a syllabic writing system, with open syllables and vowels. Therefore, the same order of magnitude in terms of number of signs is expected;

- As discussed in Chapter 1, it is known that the Cypriot Greek Syllabary is related to the other writing systems in the Aegean and to Cypro-Minoan. In particular, graphemes share a similar shape to corresponding ones in Cypro-Minoan.

- The Cypriot Greek Syllabary corpus is larger than the Cypro-Minoan one, so it is possible to select a portion of it to roughly match the size of the Cypro-Minoan one.

These factors make the Cypriot Greek Syllabary the obvious choice for this preliminary evaluation. However, as it will be discussed later, this is only a preliminary step and a careful evaluation of the distinctions between Cypro-Minoan and the Cypriot Greek

Syllabary needs to be performed, in order to adapt the model that stems from the the experiments. The result of this evaluation has been presented in Corazza et al. (2022a), while in Corazza (2022) we focus on the process of starting from the Cypriot Greek Syllabary in order to build a model for the undeciphered Cypro-Minoan.

## 5.1 The Dataset

In order to construct a dataset for the Cypriot Greek Syllabary, the first consideration is the fact that, while it would be ideal to start from the real artifacts, it is undesirable to use photographs of the documents directly. Provided that high quality photographs of the documents are available, their usage proves problematic for a machine learning system. First, because the type of material that the glyphs are inscribed on is a variable, a perfectly reasonable categorization of the signs involves simply dividing them by material or color of the document. While this is an interesting subject by itself, these characteristics of the corpus are not interesting for the purposes of our analysis. Another issue with images is the fact that it is often difficult to read the signs even for humans, since it is not possible to perceive the depth of a stroke on an object by using a two dimensional photo. Therefore, while efforts are undergoing to produce high quality three dimensional models for documents in Linear A and Cretan Hieroglyphic (Ravanelli et al., 2022), we are still forced to use drawings of the signs in the documents.

In order to build a sufficiently large dataset for Cypriot Greek Syllabary signs, our team used scans from various sources (Casabonne et al., 2002; Egetmeyer, 2010; Masson, 1983; Mitford, 1981; Karageorghis and Karageorghis, 1956; Karageorghis, 1976; Karnava, 2019; Masson and Mitford, 1986; Mitford, 1971; Masson and Olivier, 1983; Mitford, 1958; Mitford et al., 1961; Olivier, 2007; Mitford et al., 1983). The drawing contained in these scans were cropped and categorized manually, in order to obtain a dataset that is comparable in size to the available data for Cypro-Minoan. The result, then, is a dataset of 100x100 black and white images, containing 2995 glyphs from 164 inscriptions. In order to evaluate the model, we used readings from the sources, except for a few cases when revised readings for glyphs are provided in successive publications (included in the list of sources). Each image was named following a specific scheme, annotated with

the name of the document and its position in the sequence. As an example, the name
`CG_ICS.092.r03.18_NE_.png` is structured as follows:

- `CG` is shared among all images from the Cypriot Greek Syllabary dataset;

- `ICS.092` denotes the name of the document;

- `r03` denotes the fact that the glyph belongs to the third line in the document;

- `18` denotes the fact that the glyph is the 18th in the third line;

- `NE_` the first part of this string denotes the reading of the sign as syllabogram "NE".
  The underscore `_` beside the syllable denotes the fact that the reading is doubtful.
  Of course, this is an optional component of the naming scheme as not all glyphs
  have doubtful readings.

Once the naming convention was applied to all images, the resulting data was carefully
checked for errors, by examining the sequences resulting from the naming scheme of the
files. The next step was to determine which images resulted from drawings of damaged
signs. The exclusion of damaged glyphs was performed for multiple reasons. In particu-
lar, the drawing of a damaged sign is surely more tentative and it might not correspond
to the true shape of the glyph. Additionally, some drawings are missing one or more
of the components of the sign and present the drawing of a fracture in the document
(see Figure 5.1) or dots representing damage in the inscription, making it harder for any
neural model to distinguish between the graphical features representing damage and the
actual shape of the glyph.

The number of damaged glyphs which were excluded is 322, so the resulting dataset
contains 2673 drawings from 164 documents. The number of distinct graphemes repre-
sented in this sample is 64, including syllabograms, punctuation and "SPACE", represent-
ing the situation where two sequences are clearly separated by a space in the document.
Crucially, the Cypriot Greek Syllabary is found in two different variants: one used mainly
in the area of Paphos, in West Cyprus (called "Paphian") and another attested on the
rest of the island ("Common"). Since the shape of some signs in these variants (5 in our
dataset) is significantly different, we opted to treat them as separate categories for our

Figure 5.1: On the left, an instance of the syllabogram "TO" that was kept in the dataset. On the right, a damaged instance that was tentatively read as "TO", which was discarded. Notice how the right drawing contains a diagonal line representing a fracture in the document.

evaluation. Out of the 56 syllabograms that are part of the Cypriot Greek Syllabary, only one is missing from our dataset, the rare "XA".

Before describing the procedure we used to develop our model, it is useful to describe some salient statistics about our Cypriot Greek Syllabary dataset. In Figure 5.2 the number of attestations for each grapheme in the Cypriot Greek Syllabary are shown. The first observation stemming from the plot is the fact that, like many linguistic features, the distribution of graphemes results in a Zipf distribution, where some graphemes are relatively frequent, while some are exceedingly rare. In particular, one interesting observation is the fact that the most commonly attested grapheme "DIV" is not even a syllabogram. Instead, this is what is called sequence divisor, used to divide words and sequences in the documents. Its prominence in the dataset has some very interesting implication, since this sign always constitutes a word boundary, despite not always being used to separate every word. This means that, even with no assumption on the script, it is sometimes possible to know with certainty that a given glyph is found in word initial or word final position. Crucially, the presence of the sequence separator is not limited to the Cypriot Greek Syllabary, since it is also prominently featured in Cypro-Minoan. The presence of the word divider will be consistently used for developing our model, since its status is not debated and it is consistently present on both Cypriot writing systems that we are interested in.

Since our goal is to leverage the structured nature of writing, where the sequences attested in the documents matter for the final results, it is also crucial to analyze the relative frequency of sequences. For this reason, in Figure 5.3 the 50 most attested
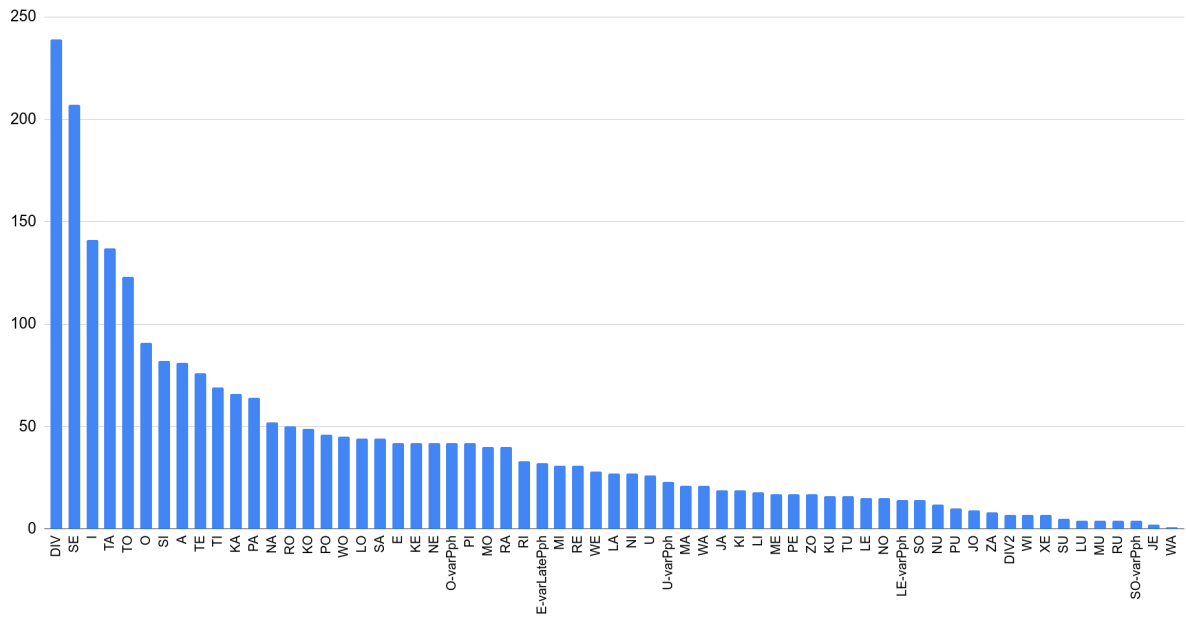
Figure 5.2: Number of attestations of each grapheme in the Cypriot Greek dataset.
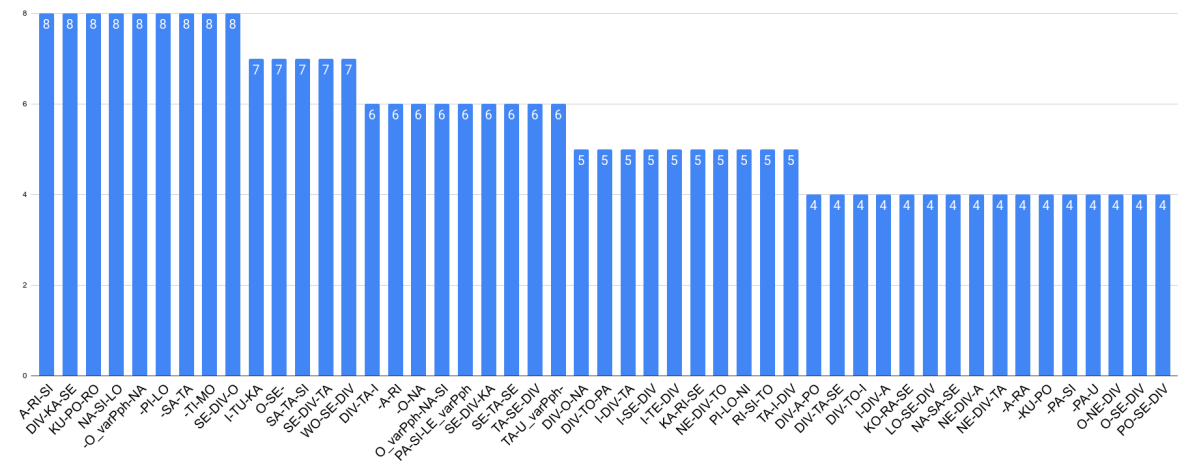


Figure 5.3: Number of attestations for the 50 most frequent glyph trigrams in the Cypriot Greek Syllabary

glyph trigrams are shown. It is very interesting to notice the prominence of the sequence divider (DIV) in this sample, as well as the presence of some instances where the plot shows the number of attestations even for pairs of signs, preceded or followed by a dash with no sign beside it. The latter are situations where the two signs are found at the beginning or at the end of inscriptions. In this case, we do consider the end and beginning of a document as a special character in our model, so it makes sense to consider these situations as trigrams, despite the absence of a third glyph in the sequence. This choice stems from the idea that the prominence of a grapheme at the beginning and end of words is important in the context of undeciphered writing systems, as it can allow experts to spot allography even when the shapes of individual attestations are different. Overall, many sequences of three signs that are prominent in the dataset appear either at the beginning or end of documents or are preceded or followed by a sequence separator, denoting a word boundary in both situations. In particular, 24 glyph trigrams contain a sequence separator, while 10 are found at the beginning of documents and two at the end. In other words, out of the 50 most frequent sequences, 36 denote the beginning or the end of a word, showing the importance of including word boundaries when considering sequences.
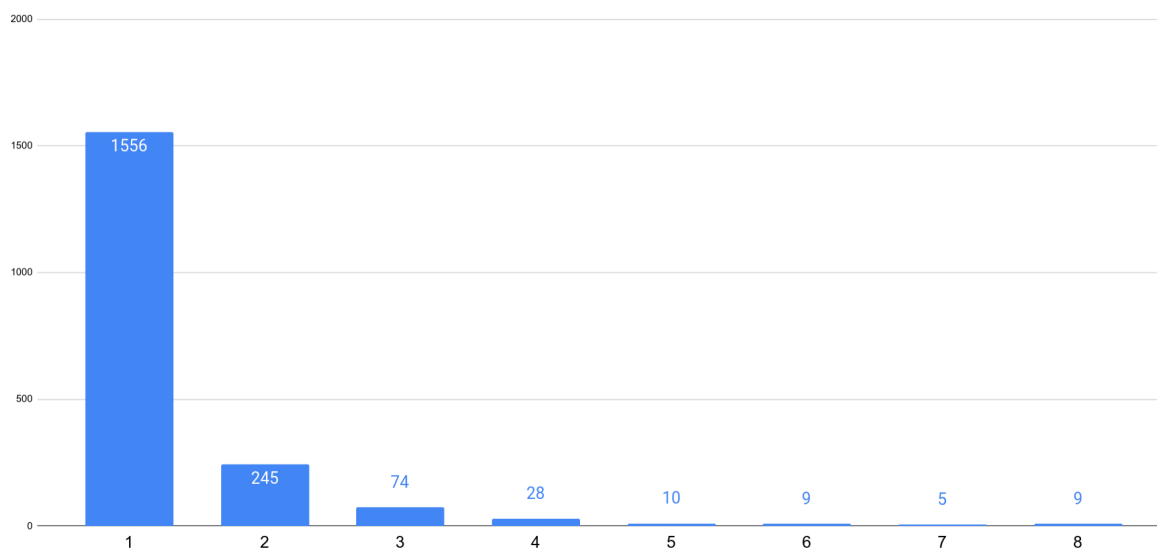


Figure 5.4: Number of grapheme trigrams having a given number of attestations.

It is clear from the figure that even the most frequent glyph trigrams are relatively
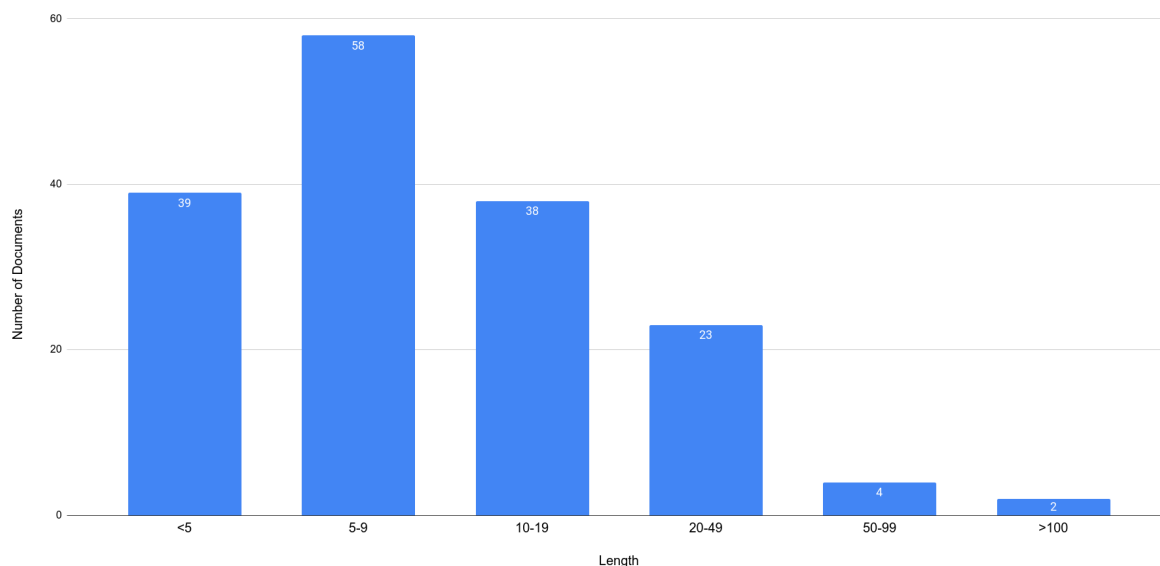
Figure 5.5: Histogram of the length of documents in the Cypriot Greek Syllabary.

infrequent in the corpus. In fact, most sequences of three signs are attested only once in the corpus. In order to show this fact, another image (Figure 5.4) shows the number of graphemes that are attested a given number of times. From this figure alone, we can see that most sequences of three signs are attested only once in the corpus, while 379 are found more than once in the dataset. While this result might suggest that no meaningful information can be extracted from trigrams, there exist a significant portion of the dataset for which multiple attestations of the same trigram exist. Additionally, the most frequent bigrams are attested hundreds of times, suggesting that some information can be extracted from short sequences of glyphs. What is evident is that there is very little advantage in considering longer sequences, as the frequency of the most frequent sequences of three glyphs is already quite low.

One final aspect of note for the dataset is the average length of the documents. Since our aim is to use a model that leverages the information contained in sequences, it is useful to observe that most documents in our dataset contain less than 25 signs overall, while only a handful have more than 75. This result shows one of the challenges of some ancient writing systems: since many documents exist with only a handful of signs inscribed on them, statistical analyses are difficult, unless a large number of documents

can be used. In our small sample of Cypriot Greek, the longest document has 584 signs, while the shortest one only contains 2 signs. On average, the length of a document is approximately 9.

## 5.2 Model selection

In order to perform an analysis of glyphs for undeciphered languages, a strict set of requirements needs to be fulfilled in order to select a suitable model for this task. During our preliminary evaluation, we considered multiple types of models. Our first candidate was derived from an interesting dataset containing glyphs from multiple different languages called Omniglot (Lake et al., 2015). In particular, we thought that it could be useful to use a transfer learning approach based on Siamese Networks (Koch et al., 2015), starting from Omniglot and then proceeding by applying the resulting model to Cypro-Minoan. However, this approach is problematic when applied to undeciphered scripts, as we are in fact learning the behaviour that emerges from different writing systems, which might not be coherent with the specific script that is the goal of our investigation. Additionally, the incorporation of any contextual information to this kind of models can be difficult, as it is not possible to use sequences of glyphs in other writing systems to derive any information on the Cypriot Greek Syllabary.

In fact, we quickly realized that that our model should be completely unsupervised, in order to avoid any assumption on the script. Therefore, another possibility was the usage of convolutional autoencoders in order to learn good quality vector representations for each of our signs, without requiring any type of supervision. Providing contextual information, then, would be possible, for example one could ask the model to reconstruct the image from the ones appearing on its left and on its right. This approach, however, is also problematic, as using autoencoders on glyphs would entail teaching them to be very sensitive about the specific shape and position of the strokes inside the image. This is precisely the opposite of what is demanded from a model that needs to investigate allography, since it is important that the model learns to distinguish what are the diagnostic trait and the invariants for each of the graphemes in the dataset, instead of focusing on specific traits of any glyph. The inclusion of contextual information, then, further

complicates things, as it is unreasonable for a model to be able to reconstruct the precise shape of a glyph from its context. Finally, the usage of autoencoders in computer vision is generally limited to noise reduction or generative models, since they are not suited to create good quality vectors that can be used for clustering or other classification tasks.
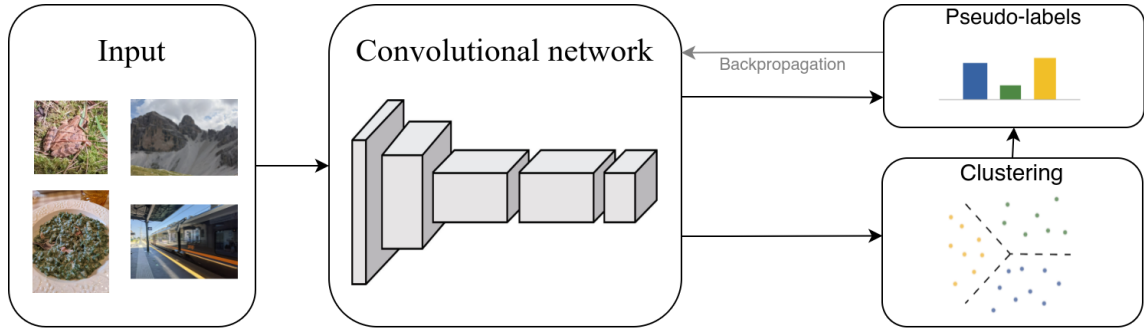


Figure 5.6: DeepCluster. Image source: Corazza et al. (2022b)

The last class of models that we considered is the one producing the most successful attempts at unsupervised classification or clustering of images. In particular, our focus was on unsupervised computer vision models that explicitly use clustering as an objective within their structure. Among those, the one with the most promise a priori appeared to be DeepCluster (see Figure 5.6), and its successor DeepClusterv2. DeepCluster, in particular, starts from a randomly initialized ResNet50 convolutional model, producing vectors for each image in the dataset. Then, these vectors are clustered using K-Means and the resulting categories are used as pseudo-labels for the backpropagation step that updates the weights of the network. At the end of each epoch, then, the model is applied to all the images, and K-Means is used again to obtain a pseudo-label for each image. The model uses the cosine distance as the metric for the application of K-Means, and it produces normalized vectors with unit norm.

DeepClusterv2, then, is an improvement of this approach, which incorporates a crucial data augmentation technique for situations where the amount of data available is quite limited, like in the case of undeciphered scripts. Instead of simply providing the model with each image in the dataset, DeepClusterv2 uses random transformations of the image, consisting of random crops, color distortions and random horizontal flips. Each image is then transformed multiple times and these augmented versions of the image are all fed

to the network. The model, then, is required to produce vectors belonging to the same K-Means cluster for all augmented versions of the same image. Other improvements upon the original DeepCluster include a cosine learning rate schedule and the usage of a multi-layer perceptron as the final projection of the model.

Interestingly, while the model performs a classification task based on the categories found during the application of K-Means, the task is performed by incorporating the centroids inside the network itself. In particular, the last layer of the model contains the centroids found during the previous application of K-Means. As mentioned previously, the distance metric used in the model is the cosine-based. Given two vectors of the same dimension $v, w \in \mathbb{R}^n$, we can obtain their cosine similarity as follows:

$$S_{\cos} = \frac{v \cdot w}{||v|| \ ||w||} \tag{5.1}$$

DeepClusterv2, however, uses only unit-norm vectors, which means that:

$$||v|| = ||w|| = 1 \tag{5.2}$$

So the denominator in Equation 5.1 is always one, and the dot product between the two vectors corresponds to their cosine similarity. Then, given a vector produced by the DeepClusterv2 model $v$ and its last layer, which uses a matrix populated with the centroids $C = c_1, \ldots, c_m$ obtained from K-Means, we obtain:

$$Cv = o = [o_1, \ldots, o_m] \quad o_i = c_i \cdot v \tag{5.3}$$

By applying the last layer, then, we obtain the dot product between the vector and each centroid $c_i$ and this value is stored in a cell of the output vector. Then, it is possible to apply a softmax function to the output, to make it so the sum of all values is one, and treat this problem as a classification task by applying a categorical cross entropy loss. This structure allows the model to avoid re-initializing the last layer after each epoch, when the centroids and clusters change following the application of K-Means.

Crucially, the source code for DeepClusterv2 is publicly available [1], so it can be used as the basis for our experiments. The model is implemented in Python using Pytorch

---

[1] `https://github.com/facebookresearch/swav`

(Paszke et al., 2019) as the deep learning framework. Having selected a promising candidate for our model, it is now necessary to create a context aware variant of this model, which we named Sign2Vec$_c$.

## 5.3    A context-aware model: Sign2Vec$_c$

In order to incorporate a context-aware component for our Sign2Vec$_c$ model, it's useful to define formally the behaviour of the non contextual component of our model. In particular, this component uses the function $M_s$:

$$M_s(X_i) = MLP_s(R_{18}(X_i)) \tag{5.4}$$

Where:

- $X_i \in \mathbb{R}^{i_w \times i_h}$ is the i-th image in the training data. $i_w$ and $i_h$ represent the width and height of the input images, in pixels;

- $R_{18}(X_i) \in \mathbb{R}^{r_s}$ represents the output of a ResNet18, used in our experiments to replace the original ResNet50 in an effort to reduce the number of parameters of the model. $r_s$ represents the size of the output of the ResNet18 model;

- $MLP_s$, is a multi-layer perceptron that is applied to the output of the ResNet. It produces vectors of size $v_s$, where $v_s$ is a hyperparameter regulating the size of the vectors obtained from the model.

Finally, the loss is the categorical cross entropy $H_c$ between the product of the normalized $M_s(X_i)$ and $C$ with respect to $Y_i$, which represents the one-hot encoded centroids obtained from the application of K-Means:

$$H_c\left(C\frac{M_s(X_i)}{||M_s(X_i)||}, Y_i\right) \tag{5.5}$$

Notice that the product between the centroids $C$ and the normalized vectors $\frac{M_s(X_i)}{||M_s(X_i)||}$ produces the cosine similarities between each vector in the batch and each centroid, as shown in equation 5.1.

$$(1 - \lambda_c) \, H_c \left( C \frac{M_s(X_i)}{||M_s(X_i)||}, Y_i \right) \qquad \lambda_c \, H_c \left( C \frac{M_c(L_i, R_i)}{||M_c(L_i, R_i)||}, Y_i \right)$$
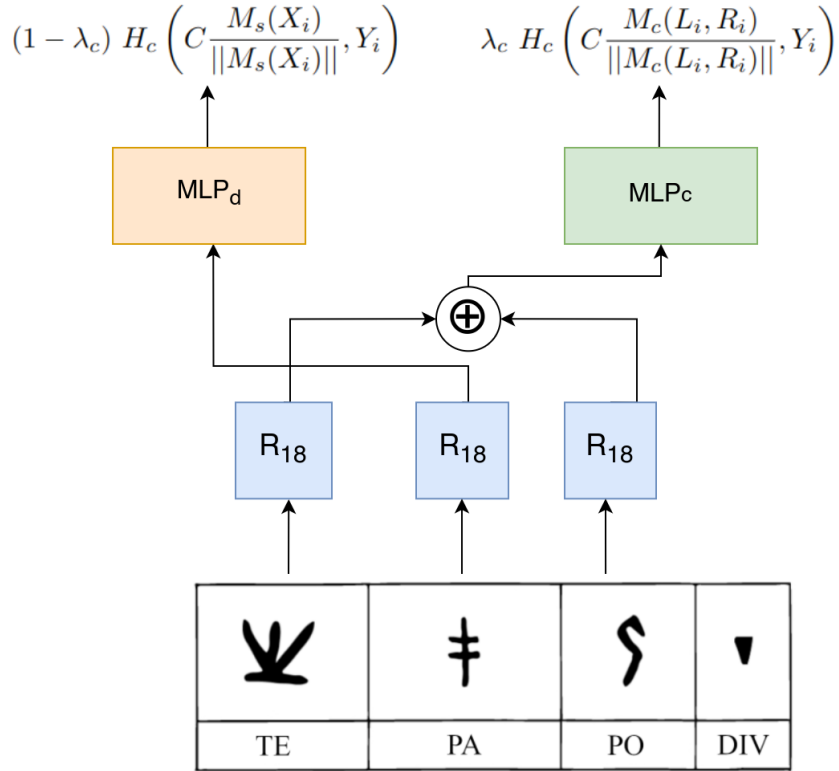
Figure 5.7: Our contextual model for signs, called Sign2Vec$_c$. Layers with the same colors highlight shared weights. On the bottom, a sample sequence of Cypriot Greek Syllabary signs.

In order to incorporate a contextual component in the model, then, we opted to consider sequences of three signs and to teach the model to predict the cluster that a given sign belongs to using the signs on its left and right. This approach is inspired by the CBOW variant of word2vec (Mikolov et al., 2013b) proposed for creating word embeddings. In order to perform this task, we add a component to the model which computes a joint vector representation for the context of a given sign:

$$M_c(L_i, R_i) = MLP_c(R_{18}(L_i) \oplus R_{18}(R_i)) \tag{5.6}$$

Where:

- $L_i, R_i$ are the signs found on the left and right of the current sign $X_i$;

- $\oplus$ denotes the concatenation of two vectors;

- $MLP_c$ is a multi-layer perceptron. Crucially, it does not share weights with $MLP_s$, as $MLP_c$ operates on a vector obtained from $R_{18}(L_i) \oplus R_{18}(R_i)$ which has size $2v_s$. The ResNet $R_{18}$ is instead shared.

Once we obtained this vector representation for a pair of images, they can be used to calculate the contextual component of our loss using cross entropy as follows:

$$H_c\left(C\frac{M_c(L_i, R_i)}{||M_c(L_i, R_i)||}, Y_i\right) \tag{5.7}$$

Like in equation 5.5, the output of the model is normalized and its product with the centroids $C$ produces the matrix of cosine similarities (dot products) between each vector and each centroid obtained from K-Means.

Using the non contextual and contextual components of the model, it is possible to define the complete context-aware model, which we called Sign2Vec$_c$ (see Figure 5.7). The loss function of the complete model is:

$$\mathcal{L}(C, X_i, L_i, R_i, Y_i) = (1-\lambda_c)\, H_c\left(C\frac{M_s(X_i)}{||M_s(X_i)||}, Y_i\right) + \lambda_c\, H_c\left(C\frac{M_c(L_i, R_i)}{||M_c(L_i, R_i)||}, Y_i\right) \tag{5.8}$$

Where $\lambda_c \in [0, 1]$ is a hyperparameter used to determine the relative weight of the two different components of the model, which now is trained using information derived from each sign $X_i$ and its context $(L_i, R_i)$.

## 5.4 Sequences and Damaged inscriptions

Having defined the behaviour of our model, which incorporates information about the left and right context of each sign in its formulation, it is necessary to discuss some corner cases where complete sequences are not available. In particular, we can distinguish two different situations where we cannot construct a proper context for a given sign $x_i$:

- $x_i$ is found at the beginning or at the end of a document, so there is no sign preceding or following it;

- Before or after $x_i$ the document is either completely broken or only a damaged sign, which is excluded from the dataset, can be found.

It is clear that these situations have the potential to be disruptive when using contextual information in our model, since many frequent sequences of three signs include the beginning or the end of documents (see Figure 5.3) and there are many damaged sequences, which were not shown in the trigrams plot. These two situations are also very different, since the beginning and end of a document always constitute word boundaries, while damaged portions of the document can in principle contain damaged signs or no signs at all and we have no way of reconstructing with certainty the missing content.

In order to construct meaningful sequences for signs found at the beginning or end of documents, then, it is possible to leverage an interesting feature of both the Cypriot Greek Syllabary and Cypro-Minoan: the presence of sequence separators. While, especially in the Cypriot Greek Syllabary, they are not always used to separate each word, they always constitute a word boundary, since they are used to separate sequences of words. Additionally, their status is not debated even in Cypro-Minoan and they are clearly not syllabograms. For this reason, we use this peculiar feature to produce the left and right contexts of signs that appear at the beginning or at the end of documents, respectively. In particular, in order to avoid an arbitrary choice of of a sequence divider for a specific sign, we choose them at random during training. This random selection avoids an over-reliance of the model on a specific artificial word separator for signs that appear at the beginning or end of sequences, while using only prior knowledge about sequence separators, which are not syllabograms and they are not the subject of our analysis.

With respect to damaged signs and broken inscriptions, then, we need to find a meaningful substitute for missing data. Crucially, since no information can in principle be inferred from damage, this artificial image should be constructed so that no two images are the same, as otherwise the model might rely on it as an invariant component of the contexts for some signs. The solution we adopted for this problem can be found in standardized representations of damaged portions of signs in the literature, namely the usage of a dotted pattern to indicate damage. For this reason, then, we adopt the same convention to represent damaged signs. Since this pattern needs to vary at each iteration of the algorithm, it is crucial to select an adequate algorithm that can quickly produce dots in a two dimensional image.

Figure 5.8: Two 100x100 pixel images with 122 random dots. On the left, the dots are selected from two random uniform distributions (one for each coordinate). On the right, the result from the application of the Bridson algorithm.

For this task, the usage of a random uniform distribution for the two coordinates of each point tends to produce situations where many points are found in a small area, while other areas of the image are empty. Since this might result, due to random variation, in small shapes being formed from multiple, randomly distributed points, we opted to use a more sophisticated distribution in order to generate our images. This has the added advantage of producing dotted patterns that are generally more similar to what a human would produce. For these reasons, a Poisson disk sampling can instead be used for the points, in the form of the Bridson algorithm (Bridson, 2007). This algorithm uses a the following procedure:

- **Step 1**: select a random point $x_0$, uniformly chosen in the two dimensional space and insert it in the active list;

- **Step 2**: select a random point $x_i$ from the active list and generate up to $k$ point in the annulus between radius $r$ and $2r$ around $x_i$. Check whether the new point is within distance $r$ of any of the other points. If no point is within distance $r$ of $x_i$, insert $x_i$ in the active list. If, after $k$ attempts, no such point can be found, remove $x_i$ from the list.

In addition to the randomness of the position of the dots, we also use dot sizes that are slightly variable, using a random uniform distribution for their radius. In Figure 5.8, the application of the Bridson algorithm and a uniform distribution of dots are shown. It is clear, then, that the Bridson algorithm produces points that are more evenly spaced and resemble what a human would draw, especially when contrasted with the image containing uniformly distributed dots.

Having defined the two corner cases concerning signs at the beginning or end of documents and signs surrounded by damaged portions of the inscription, we are now able to produce a left and right image for each sign in our corpus. Crucially, this allows us to leverage even information about the word-initial or word-final position of signs even in very short inscriptions, which are frequent in our dataset.

## 5.5    Experimental Setting

The output of Sign2Vec$_c$ is a vector space, representing all the glyphs in our Cypriot Greek Syllabary dataset. As a first step, in order to perform a visual evaluation of this space, we applied a dimensionality reduction to the vectors, to obtain a three dimensional representation for each sign. For the dimensionality reduction, we applied t-distributed stochastic neighbor embedding (t-SNE), as it operates by minimizing Kullback-Leibler divergence between the distances between vectors in the original space and in the reduced space (Van der Maaten and Hinton, 2008). Since the distance between vectors and centroids is precisely the criterion used in the application of K-Means in both Deep-Clusterv2 and Sign2Vec$_c$, the preservation of distances is a desirable criterion when visualizing the outputs of these models. Using these three dimensional vectors, we built a three dimensional visualization in WebGL, based on the Scatter-gl[2] project, in turn derived from the embedding projector of TensorFlow. The visualizations obtained from the three models used in our evaluation for the Cypriot Greek Syllabary can be found at `https://corpora.ficlit.unibo.it/INSCRIBE/PaperCG/`. These visualizations were very useful in our experiments, as they allow experts to spot mistakes in the transcription of signs, as well as a qualitative evaluation of the algorithms. Furthermore, they allow us to visually detect overarching patterns in the corpus, which involve large groups of signs.

While using three dimensional visualizations of glyphs can be useful, in order to assess the impact of the inclusion of contextual information in our Sign2Vec$_c$ model, two more tasks need to be performed. First, a best-case scenario evaluation can be used in order to choose sensible hyperparameters for the model. In these experiments, we assume that

---

[2]`https://github.com/PAIR-code/scatter-gl`

the number of clusters is known a priori, in order to assess the capabilities of the model itself. Then, a more rigorous worse-case procedure must be devised for the evaluation of Sign2Vec$_c$ in comparison with DeepClusterv2 in a completely unsupervised setting, where the number of clusters is unknown. This worse-case evaluationa uses 10 runs for each model, in order to reduce the impact of the random initialization of the parameters in the evaluation of the results and to test any difference in performance for statistical significance.

For both sets of experiments, we opted to exclude word dividers and numerals from the clustering procedure, while they are still used during training. This choice is motivated by the fact that we are mostly interested in the allography between glyphs, while the numerals and sequence separators constitute different phenomena. Their inclusion in the training set, then, is motivated by the idea that they are still part of contexts of other signs, so it is useful to teach the model about their contexts, as well as their presence in other signs contexts.

Since DeepClusterv2 and Sign2Vec$_c$ are used to partition images in different categories, a clustering task is an obvious choice to evaluate the models, since this approach allows us to determine how well a model separates the different graphemes in the dataset. However, since our goal is to prove that the usage of contextual information in Sign2Vec$_c$ is an advantage over DeepClusterv2 in the context of undeciphered writing systems, our starting point for comparing Sign2Vec$_c$ and DeepClusterv2 is one where no prior knowledge of the script is available. The only possible usage of the gold standard transcriptions of the Cypriot Greek Syllabary is the final evaluation of the model through clustering metrics, while no knowledge can be used during the training phase.

A crucial aspect of our evaluation is the choice of a clustering algorithm to apply to the data. While K-Means is part of the training procedure of Sign2Vec$_c$ itself and it is used in the selection of the hyperparameters, its usage in our worse-case setting, where no ground truth can be established about the exact number of clusters is problematic, since this would force K-Means to severely over-cluster the dataset, impacting the overall performance of the model. For this reason, our evaluation is conducted using only a density based clustering algorithm, the Density-Based Spatial Clustering of Applications with Noise (Ester et al., 1996), also known as DBSCAN. This algorithm does not require

the specification of the number of clusters, but instead it requires to specify a minimum number of points to produce a neighborhood $n$ and, most importantly, the value $\epsilon$, which determines the maximum distance between two points according to the chosen metric in order to consider two points as neighboring. Crucially, the output of DBSCAN produces, in addition to a number of clusters, a set of points that are considered as outliers or "noise" and for which no cluster can be determined.

The procedure used by DBSCAN to produce clusters starts from examining the points in the sample and defining relations between them on the basis of $n$ and $\epsilon$. In particular:

- A point $p$ is a *core point* if it has at least $n$ points within distance $\epsilon$;

- A point $q$ is *directly reachable* from $p$ if it is within distance $\epsilon$ from it;

- A point $q$ is *reachable* from $p$ if a sequence of points $p, p_1 \ldots, p_n, q$ exists such that for each pair of successive points $m, n$ in the sequence, $n$ is directly reachable from $m$.

Using these definitions, the procedure used by DBSCAN in order to determine which points belong to the same clusters and how many clusters are in the sample is the following:

1. For each points, determine its $\epsilon$ neighborhood. Determine which points have at least $n$ within distance $\epsilon$ (*core points*);

2. For each *core point*, determine which other points are *reachable* from it, ignoring non-core points. These belong to the same cluster;

3. Assign non-core points that are in the $\epsilon$ neighborhood of any point that belongs to a cluster to that cluster, consider others as noise.

From these principles, DBSCAN partitions the data in a number of clusters and a number of points that are considered as outliers or noise. While this allows us to use the algorithm without specifying the number of clusters as an input, but it is clear from the informal description of the algorithm that the choice of the two parameters $m$ and $\epsilon$ is crucial in order to produce meaningful results. In the context of an undeciphered writing system,
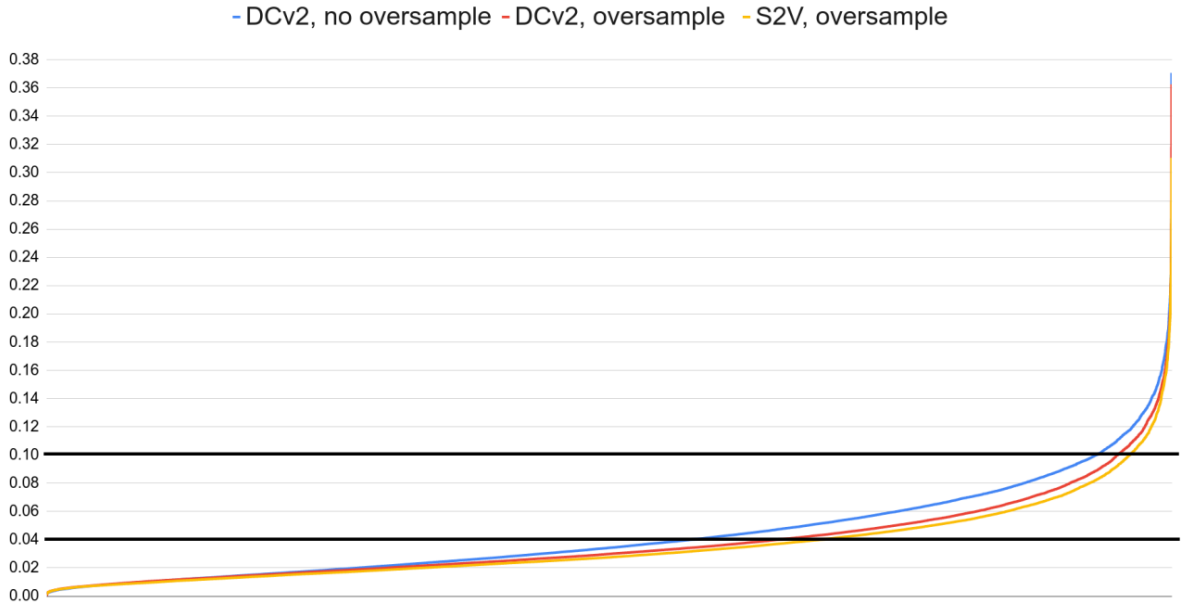
Figure 5.9: The elbow plots obtained from DeepClusterv2 with no oversampling, Deep-Clusterv2 with oversampling and Sign2vec$_c$ with oversampling. The two horizontal lines represent the range of values considered in the experiments.

the choice of $m$ is less problematic: since it is not known whether less frequent graphemes are present in the sample, the minimum value of 2 can be safely chosen, so that the algorithm can produce arbitrarily small neighborhoods (and, in turn, clusters). The choice of $\epsilon$, however, is more problematic, as it directly impacts the number of clusters in the final result. One possibility to determine the value of $\epsilon$ is the usage of a heuristic called the elbow method (Rahmah and Sitanggang, 2016). This methods operates by calculating the furthest distance of each point from its two nearest neighbors. Then, these values are plotted in ascending order and the $\epsilon$ value is chosen to match the elbow in the plot. This value, then, is a point of diminishing returns, where increasing $\epsilon$ does not result in a large number of points that were considered as noise becoming part of clusters.

While the application of the elbow method can be attempted, in our experiments the resulting plot was always ambiguous and choosing an $\epsilon$ value always involved an arbitrary decision. In Figure 5.9, the plots for the three different models that we used in

our completely unsupervised evaluation are shown. Since our aim was to choose a single $\epsilon$ for each model, all vectors resulting from the 10 runs are used in order to produce the plot. It is clear from observing the figure that it can be very hard to distinguish the elbow of a plot with so many points, so the application of the elbow for our evaluation is problematic. For this reason, we chose to evaluate each model using two different approaches. On one hand, we plot the behaviour of the model across a range of $\epsilon$ values (the range is shown with black horizontal lines in Figure 5.9). On the other hand, we also show the mean and standard deviation of our 10 models across multiple metrics for the best value of $\epsilon$ and test these values for statistical significance.

While I mentioned the fact that metrics are used to show the behaviour of the models across a wide range of $\epsilon$ values, as well as their means and standard deviations for the best performing $\epsilon$, our selection of metrics is yet to be discussed. Since our goal is to use clustering as the task to evaluate our model, we selected the Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI) and V-Measure, which are explicitly devised for evaluating clustering algorithms, using their implementations from Scikit-learn (Pedregosa et al., 2011). For a formal description of these metrics, see Section 2.2.7.

## 5.6 Hyperparameters selection

As discussed in the previous section, one of the goals of the experiments using the Cypriot Greek Syllabary is the choice of sensible hyperparameters, which will be used for the eventual evaluation of the model. For this preliminary evaluation, since our DBSCAN-based protocol for the evaluation of the system in a completely unsupervised setting was yet to be developed, we started from a best-case scenario, where the number of clusters is known and K-Means is applied to the outputs of the model in order to cluster the glyphs. Before the selection of other hyperparameters, we performed the following alteration to the DeepClusterv2 model:

- We selected the size of the crops relative to the total image manually, by observing the result of crops. This was performed in order to make sure that the resulting crops contain enough information for the signs to be discernible;

- As a further measure, we discard images containing too few black pixels. As a threshold, we use the minimum between 50% of the non white pixels of the original image and 50 total pixels.

- We replaced the original ResNet50 used in DeepClusterv2 with a smaller ResNet18 model in an effort to reduce training time and the number of parameters.

- The number of clusters used within the models is set to 70, in order to create a slight over-clustering which can help improve the behaviour of the model with respect to rarer graphemes.

Our exploration of the impact of hyperparameters, then, focuses mainly on the data augmentation steps that are used during the training, and it concerns the following parameters:

- $\lambda_c$: this is the parameter that regulates the impact of context on the overall loss, as shown in Equation 5.8;

- Augmentation steps: we investigated the role of the various types of transformation on performance. In particular, we tested the model excluding some transformations (color distortion, random horizontal flip);

- Number of crops: the number of crops to produce for each group of crops. It is represented as a list, since multiple groups of crops can be specified, with different cardinalities and sizes;

- Crops for assign: this list encodes the index of the crop to use during the application of K-Means, in order to obtain the pseudo-labels.

In order to start our evaluation, we started from a baseline using the parameters shown in Table 5.1. Since we use multiple metrics to evaluate each run, and since it was not feasible to explore all parameters using multiple runs for each model, we only implement a parameter change when there is agreement between all metrics.

The first step, then, was to determine the impact of our $\lambda_c$ constant, determining the weight of context on the overall loss. For this reason, we compared a vanilla DeepClus-

| Hyper-parameter | Value |
|---|---|
| Augmentation steps | Crop, random horizontal flip, color distortion |
| Base Learning Rate | 4.8 |
| Batch size | 16 |
| Crops for assign | 0 |
| Epochs | 100 |
| Feature dimensions | 128 |
| Final learning rate | 0.0048 |
| Number iterations before freeze | 300000 |
| $\lambda_c$ | 0.2 |
| Hidden MLP size | 2048 |
| Max scale crops | [1.0, 0.6] |
| Min scale crops | [0.6, 0.4] |
| Number of crops | [2, 6] |
| Number of prototypes | [70,70,70] |
| Size of the crops | [80, 60] |
| Start warmup | 0.3 |
| Temperature | 0.1 |
| Warmup Epochs | 10 |
| Weight decay | $1 \times 10^{-6}$ |

Table 5.1: Baseline Hyper-parameters for Sign2Vec$_c$

terv2 loss with $\lambda_c = 0$ with models using different values of $\lambda_c$, as shown in Table 5.2.

The performance of the model over the selected values of $\lambda_c$ shows that all the three metrics are maximized when $\lambda_c$ is set to 0.2, so we selected that value and continued the evaluation for other parameters.

The next step in our evaluation involved the exclusion of color distortions and random horizontal flips from the data augmentation steps, in order to determine whether these steps have a positive impact on the performance of the model. In terms of horizontal flip, this augmentation step simply mirrors the image horizontally 50% of the time at random. The color distortion procedure, instead, randomly changes the brightness,

| $\lambda_c$ | ARI | AMI | V-Measure |
|---|---|---|---|
| 0 | 0.3372 | 0.5650 | 0.6461 |
| 0.1 | 0.3081 | 0.5616 | 0.6437 |
| 0.2 | **0.4022** | **0.6322** | **0.7005** |
| 0.3 | 0.3695 | 0.5998 | 0.6742 |

Table 5.2: Performance for various models using different $\lambda_c$ values. In bold, the highest values.

contrast, saturation and hue of each image 80% of the time. In Table 5.3 we show the impact of the exclusion of color distortion and random horizontal flips when compared with the contextual model found previously, with $\lambda_c = 0.2$. While both the version excluding color distortion and the version excluding random flips show an improvement in terms of ARI, the other metrics are not affected by their exclusion. For this reason, we decided to include these steps, as well as crops, in our experiments.

| Augmentation steps | ARI | AMI | V-Measure |
|---|---|---|---|
| Crop, random horizontal flip, color distortion | 0.4022 | **0.6322** | **0.7005** |
| Crop, random horizontal flip | 0.4153 | 0.6149 | 0.6868 |
| Crop, color distortion | **0.4258** | 0.6308 | 0.6996 |

Table 5.3: Performance for various models using different augmentation steps. In bold, the highest values.

Next, we focused on the cropping procedure, and we investigated what the impact of the number of crops is on the performance of our contextual model. In particular, we attempted to increase the number of crops for the bigger (between 60 and 100% of the image) and smaller crops (between 40 and 60% of the image) separately. The parameters for the crops are used in groups, where each element in each list refers to the n-th group of crops. As a remainder, our baseline model uses:

- Min scale crops: [0.6,0.4];

- Max scale crops: [1.0, 0.6];

- Number of crops: [2,6];

- Size of crops: [80, 60].

These parameters produce two groups of crops. The first group contains 2 bigger crops, each representing 60 to 100% of the original image, of size $80 \times 80$ pixels. The second group contains 6 smaller crops, representing 40 to 60% of the original and with size $60 \times 60$ pixels.

For the number of crops, we started from the best model obtained from the previous steps, and compared it against multiple different variants of it, where this parameter is changed. First, we focused on the optimal number of bigger crops.

| Number of crops | ARI | AMI | V-Measure |
|:---:|:---:|:---:|:---:|
| [2,6] | 0.4022 | 0.6322 | 0.7005 |
| [6,6] | **0.5250** | 0.7048 | 0.7591 |
| [10,6] | 0.5064 | **0.7173** | **0.7694** |

Table 5.4: Performance for various models using a different number of bigger crops. In bold, the highest values.

As shown in Table 5.4, the best performing models when altering the number of big crops are found when this parameter is set to 6 or 10. The difference in terms of performance between these models, however, is less marked, since not all metrics show the same trend. In particular, while AMI and V-Measure seem to favour a higher number of big crops, ARI shows an opposite result. Since the usage of more crops increases the time needed to train our model, we selected the lower value of 6 for the number of bigger crops in our model.

Then, we considered the number of smaller crops and used a similar procedure, gradually incrementing the number of small crops in order to find a reasonable value for this parameter. The results of this procedure, shown in Table 5.5, mirror to an extent the ones obtained when evaluating the smaller crops. In particular, we see a slight improvement across all metrics when increasing the number of smaller crops from 6 to 10. A further increase in the number of crops, from 10 to 14, however, does not yield such a result. In particular, once again, AMI and V-Measure favour a larger number of 14 bigger

| Number of crops | ARI | AMI | V-Measure |
|:---:|:---:|:---:|:---:|
| [6,6] | 0.5250 | 0.7048 | 0.7591 |
| [6,10] | **0.5368** | 0.7222 | 0.7734 |
| [6,14] | 0.5169 | **0.7244** | **0.7754** |

Table 5.5: Performance for various models using a different number of bigger crops. In bold, the highest values.

crops, while ARI favours the lower value of 10. For this reason, since any increase in the number of crops is computationally expensive, we opted to choose the more conservative, lower value of 10 smaller crops.

One other parameter of the model is the number of assignments, which regulates which crop to use in order to apply K-Means for each final classification layer. The parameter is specified as a list, and it works in combination with the number of prototypes. As an example, we could use the following parameters:

- Number of prototypes [70,70,70];

- Crops for assign [0,1];

In this case, the model uses three output layers, so the final output of the model is a list of three vectors of 70 elements each (each element is a cluster or a centroid). K-Means is also applied three times, one for each layer. The applications of K-Means for each output layers, then, is derived from the crops by following the list in order and restarting once it's over. Therefore, in this example, the three layers would use centroids and pseudo-labels derived from applications of K-Means to images obtained from crop 0, crop 1 and crop 0 respectively. We were interested in knowing whether using multiple crops for different output layers could yield an improvement in performance, so we compared our baseline using one crop for all output layers to one using two different crops.

As shown in Table 5.6, the application of K-Means to multiple crops results in a marked decrease in performance in our experiments, so we decided to use only one crop from the first set of bigger ones.

While this evaluation is by no means exhaustive, it allows us to choose sensible hyperparameters for the model, which will be used when assessing its behaviour in a

| Crops for assign | ARI | AMI | V-Measure |
|:---:|:---:|:---:|:---:|
| [0] | **0.5368** | **0.7222** | **0.7734** |
| [0,1] | 0.3595 | 0.6102 | 0.6834 |

Table 5.6: Performance for various models using a different values for the "crops for assign" parameter. In bold, the highest values.

completely unsupervised setting. Furthermore, these preliminary results show promise for Sign2Vec$_c$, since in these experiments the usage of a context-aware component for the loss seems to improve performance. However, a more rigorous evaluation is required.

## 5.7 Results and Analysis

The first challenge in the application of Sign2Vec$_c$ in a completely unsupervised setting is the impossibility to determine the exact number of syllabograms that are present in the dataset. For this reason, we set as an hyperparameter for the number of clusters in the system to 100, since this is an over-estimate of the number of syllabograms present a syllabic system. In these experiments, then, the hyperparameters of the model are the ones discussed in the previous section, with the number of clusters set to 100 (see Table 5.7).

An effect of the application the over-clustering of our dataset, however, is the fact that many small classes are created, which in turn can have a detrimental effect on the performance of the model. This effect, combined with the zipfian distribution of graphemes in the dataset (see Figure 5.2), is problematic when working in a completely unsupervised setting. For this reason, we implemented a simple oversampling mechanism, in order to mitigate the effect of less attested graphemes and over-clustering on DBSCAN. In particular, we artificially double the size of the dataset, by replicating each sign in the dataset exactly twice. Thanks to the augmentation steps used by the model during training, two augmented versions of an image are always different. Therefore, we can provide K-Means with the same sign twice without the two images coinciding. In a sense, we are forcing the model to overfit on the data, in order for it to be able to separate even less frequent graphemes, which is especially useful when over-clustering. While this

| Hyper-parameter | Value |
|---|---|
| Architecture | Resnet18 |
| Base Learning Rate | 4.8 |
| Batch size ($b_s$) | 16 |
| Crops for assign | 0 |
| Epochs | 100 |
| Feature dimensions ($v_s$) | 128 |
| Final learning rate | 0.0048 |
| Number iterations before freeze | 300000 |
| $\lambda_c$ | 0.2 |
| Hidden MLP size | 2048 |
| Max scale crops | [1.0, 0.6] |
| Min scale crops | [0.6, 0.4] |
| Number of crops | [6, 10] |
| Number of prototypes | [100,100,100] |
| Size of the crops | [80, 60] |
| Start warmup | 0.3 |
| Temperature | 0.1 |
| Warmup Epochs | 10 |
| Weight decay | $1 \times 10^{-6}$ |

Table 5.7: Hyper-parameters for Sign2Vec$_c$

would be problematic in any other setting, we are treating the Cypriot Greek Syllabary as an undeciphered script and our dataset as the only available data in this writing system. For this reason, while it would be great to produce a model that is able to generalize on new data, the hypothesis is that such data is simply not available. This procedure, however, produces a trade-off. In our three dimensional visualizations, in particular, we observed a more marked separation between groups of signs, but the model seems less able to retain the relation between different groups of images (see Figure 5.10), which might hinder some tasks that do not use clustering.

As discussed in the previous section, the arbitrary choice of an $\epsilon$ value for DBSCAN is problematic and even the elbow heuristic is not useful when many runs of the same model
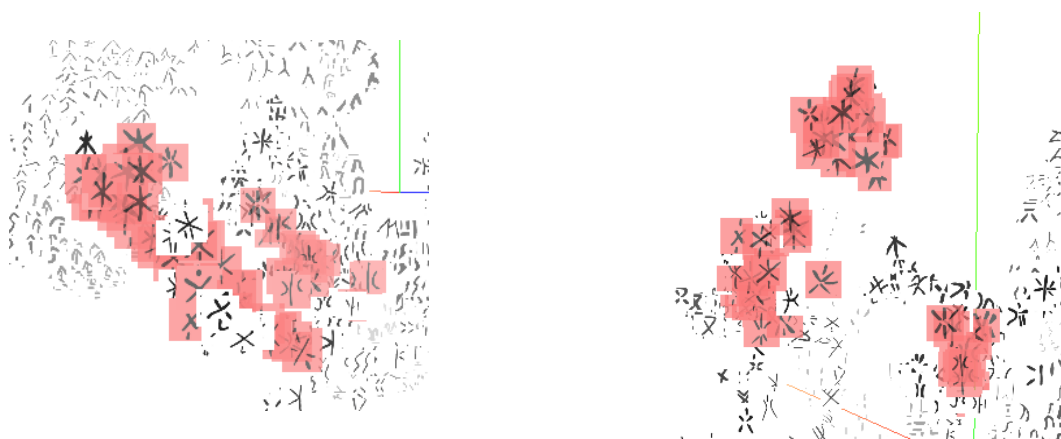
Figure 5.10: Syllabogram "A" in two DeepClusterv2 models. On the left, a model using no oversampling, on the right one using oversampling. Notice how the signs on the right are separated from other syllabograms, but they are also further apart from other similarly shaped graphemes (E, WA, etc).

or many points are involved. As the first step of our evaluation, then, the mean values of the clustering metrics across 10 randomly initialized runs for each model are plotted for a wide range of $\epsilon$ values for DBSCAN. The three variants of the model used are: DeepClusterv2, DeepClusterv2 with oversampling and Sign2Vec$_c$ with oversampling. The results for ARI, AMI and V-Measure are shown in Figures 5.11, 5.12, 5.13 respectively.

This evaluation, then, shows some interesting trends when comparing the three variants of models. First, the impact of our oversampling approach is evident, as the two models using oversampling show a marked improvement in performance when compared with the non-oversampled variant of DeepClusterv2. Furthermore, Sign2Vec$_c$ with oversampling shows the highest performance when compared with the two DeepClusterv2 models, which suggests that even in a worse-case unsupervised setting, the usage of context is beneficial to an unsupervised system for clustering graphemes of ancient scripts.

Another interesting pattern emerges from these plots, which is one of the reasons why using a single value of $\epsilon$ for any evaluation can be deceiving. In particular, Sign2Vec is not only more performant, but it shows a more stable behaviour across a wider range of $\epsilon$ values. The same difference can be spotted between the oversampled variant of DeepClusterv2 and its non-oversampled counterpart. This property is an interesting
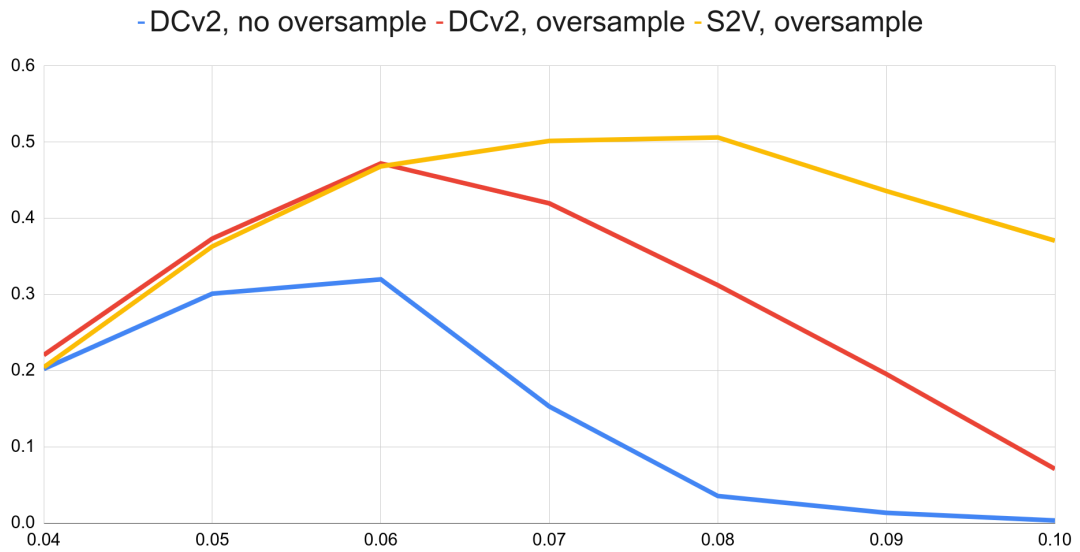
Figure 5.11: Mean ARI for all three models, for a range of $\epsilon$ value.
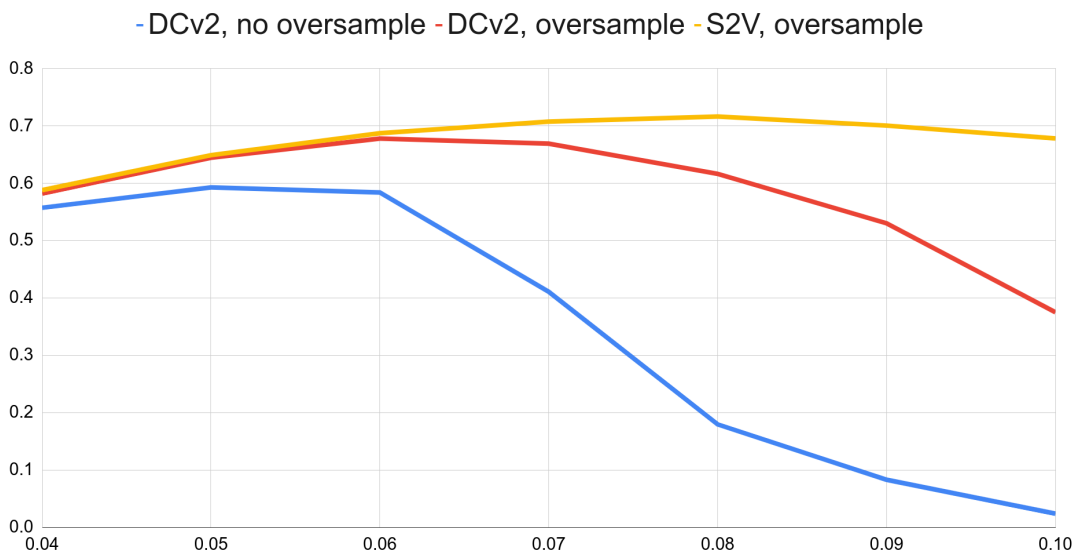


Figure 5.12: Mean AMI for all three models, for a range of $\epsilon$ value

one, as in a completely unsupervised setting it can be difficult to choose the best value of $\epsilon$ a priori, so the fact that our model can tolerate a wider range of values can simplify any downstream analysis. Another interesting note is the fact that, while the range of $\epsilon$ values shown always produces the highest values for the three models across all metrics, it is not clear when applying the elbow heuristic (see Figure 5.9) that the elbow is close
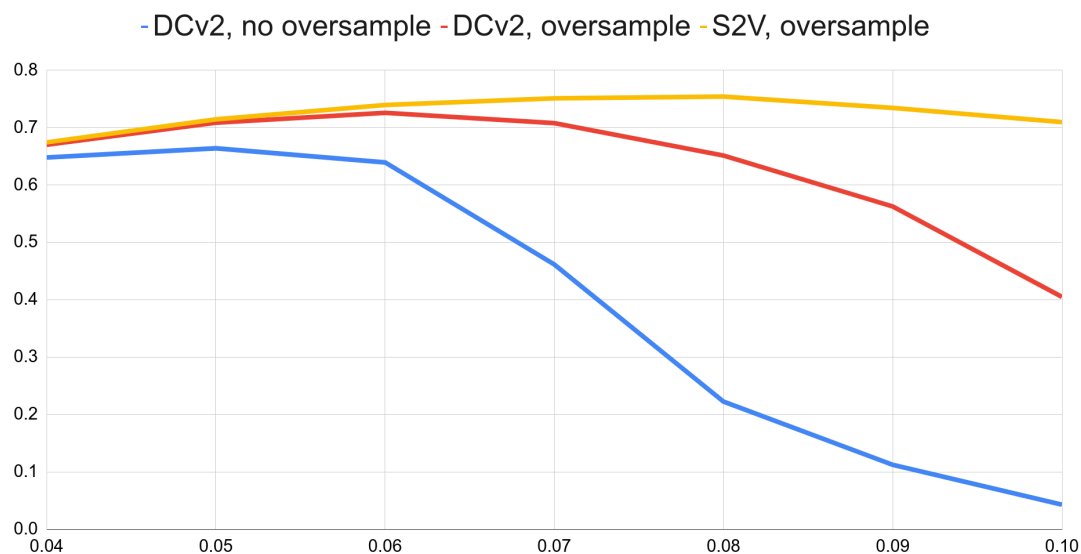
Figure 5.13: Mean V-Measure for all three models, for a range of $\epsilon$ value

to the best possible value. This shows why the choice of $\epsilon$ is so problematic when relying on such a heuristic.

| Model | $\epsilon$ value | Adjusted Rand Index | Adjusted Mutual Information | V-measure |
|---|---|---|---|---|
| DC2, no oversample | 0.05 | $0.30 \pm 0.02$ | $0.59 \pm 0.02$ | $0.66 \pm 0.03$ |
| DC2, oversample | 0.06 | $0.47 \pm 0.02$ | $0.68 \pm 0.02$ | $0.73 \pm 0.01$ |
| S2V, oversample | 0.08 | $\mathbf{0.51 \pm 0.04}$ | $\mathbf{0.72 \pm 0.02}$ | $\mathbf{0.75 \pm 0.01}$ |

Table 5.8: Means and standard deviations for all the clustering metrics of the three models, for the best choice of $\epsilon$. In bold, the highest means.

While observations about the overarching trends of the models over a range of $\epsilon$ values are useful to assess the stability and usefulness of Sign2Vec$_c$, another approach in the comparison of the three models regards the metrics that can be obtained by selecting the best value of $\epsilon$. In Table 5.8 the mean and standard deviations of all metrics are reported for all three models across 10 runs, for the best possible selection of $\epsilon$. Using these values, then, it is possible to perform a one-tailed t-test comparing them, in order to determine whether the difference between the means of the metrics is statistically significant ($p < 0.05$).

| Models | Adjusted Rand Index | Adjusted Mutual Information | V-measure |
|---|---|---|---|
| DC2 oversample, DC2 no oversample | $1.92 * 10^{-5}$ | $1.21 * 10^{-4}$ | $1.71 * 10^{-4}$ |
| S2V oversample, DC2 oversample | $0.01$ | $3.60 * 10^{-5}$ | $1.17 * 10^{-4}$ |

Table 5.9: One tailed t-tests comparing the metrics obtained from the models.

In Table 5.9 the values of two comparisons are shown, one involving DeepClusterv2 with and without oversampling, while the other involves Sign2Vec$_c$ and DeepClusterv2, both with oversampling. Comparing the two DeepClusterv2 models, then, shows that the usage of oversampling produces clusters that have higher mean ARI, AMI and V-Measure and that the difference between the two is statistically significant. When comparing Sign2Vec$_c$ and DeepClusterv2, then, we can conclude that Sign2Vec produces better quality clusters according to the metrics, and that the difference between the mean values of the metric obtained from the two models is also statistically significant.

In conclusion, this analysis proves the effectiveness of Sign2Vec$_c$, which is not only more stable over a wider range of $\epsilon$ values when using DBSCAN, but it also produces measurably higher quality clusters for our Cypriot Greek Syllabary dataset. Crucially, this model is able to leverage contextual information, and it produces vectors that better separate graphemes for the Cypriot Greek Syllabary. In addition to this result, the selection of the hyperparameters described previously allows us to apply the same models to undeciphered writing systems in a real unsupervised setting having selected the values for the most important parameters. This combination of factors, then, allows us to start considering the usage of a context-aware model to an open question: investigating the inventory of signs of Cypro-Minoan. While these results are encouraging, however, each writing system is different and its peculiarities need to be considered when adapting a model such as Sign2Vec$_c$.

# Chapter 6

# Cypro-Minoan

After the experiments on the Cypriot Greek Syllabary, it is now possible to focus on the application of our unsupervised deep learning model to Cypro-Minoan. This undeciphered writing system is characterized by the fact that there is no agreement on the inventory of signs that compose it. Additionally, the debate is still open on whether Cypro-Minoan is in fact a single writing system or whether multiple, similar, writing systems are instead present. For this reason, the usage of a novel approach based on machine learning can be useful, as the unsupervised nature of our model allows us to investigate these questions with no prior information on the script, meaning that our approach is not biased on any specific hypothesis on Cypro-Minoan.

In order to investigate Cypro-Minoan, then, the first step is the construction of an adequate dataset, containing drawings from all the available documents. Then, the model that was built and tested on the Cypriot Greek Syllabary must be adapted to the peculiarities of Cypro-Minoan. Finally, a new methodology to analyze instances of possible allography must be devised and validated. The results of this analysis have been published in Corazza et al. (2022b).

## 6.1   The Dataset

In order to create a dataset of Cypro-Minoan signs, we used a similar procedure as the one used for the Cypriot Greek Syllabary. In particular, we used scans of sign drawings

published in the two corpora (Olivier, 2007; Ferrara, 2013a) and in other publications (Hirschfeld, 1999, 2012; Davis et al., 2014; Valério, 2014; Egetmeyer, 2016; Valério, 2016). The result of this first step is a dataset composed of 3499 black and white images of size 100x100 pixels, from 230 inscriptions. From this number, we excluded 18 inscriptions for which no usable drawing was published, 5 single-sign inscriptions which are often considered as marks, as well as 5 object that are unepigraphic or whose status as proper written documents is doubtful. The next step was the exclusion of signs that are damaged, which cannot be included in the final dataset as their shape is incomplete due to fractures or damage on the document. In addition to these exclusions, some other documents were discarded from the dataset:

- The inscription ENKO Atab 001 (##001) that uses the archaic variant of Cypro-Minoan, also known as CM0. This inscription differs from the rest of the corpus in terms of chronology, and the 23 signs attested on it do not appear anywhere else in the corpus. Since the status of this archaic variant as a distinct script from the rest of the Cypro-Minoan inscriptions is not debated, it can safely be excluded from our analysis;

- 6 inscription were excluded as they are thought to be inscribed in the Cypriot Greek Syllabary and not in Cypro-Minoan. Crucially, some were used in the previous chapter as part of the Cypriot Greek Syllabary dataset. The Cypriot Greek Syllabary documents are ATHI Adis 001 (##092), discussed in Egetmeyer (2010), PPAP Mins 001-003 (##170-172), PPAP Pblo 001-002 (##189-190) discussed in Valério (2016); Ferrara (2012); Duhoux (2012) respectively.

- The last two signs in inscription HALA Abou 001 (##088) whose segmentation is debated.

These exclusions amount to 600 total signs, resulting in a final dataset composed of 2899 images from 213 inscriptions. As for the Cypriot Greek dataset, we used a naming scheme that encodes the exact position of the sign within a Cypro-Minoan document. As an example, the name `CM_ENKO.Atab.002B.r10022d.1_102_.png` encodes the following information:

- `CM` is shared among all signs from the Cypro-Minoan dataset;

- `ENKO.Atab.002B` denotes that the sign is from the second (`002`) clay tablet (`Atab`) from Enkomi (`ENKO`). The letter B in `002B` denotes the fact that the document is found on side B of the tablet.

- `r10022d.1` denotes the position of the sign in the document. In particular, some longer documents in the Cypro-Minoan corpus have multiple portions of text, separated by lines inscribed in the documents. In this example, the sign is in the first fragment (`r1`) and on the 22nd line of that fragment (`22`). The two components of the dimensions are separated by two zeros. Finally, in an effort to simplify the correspondence between the image and the source material, we encode the position of the sign in the line with respect to the page it is found on in the corpus, so the string `d.01` means that the document is the first one on the right page representing this line (we use g for *gauche* in French, meaning left, d for *droite*, meaning right).

- `102_` denotes the reading of the sign according to the reference material where this is found. Notice that, since Cypro-Minoan is undeciphered, all syllabograms are transcribed with a numerical code, while for the Cypriot Greek Syllabary it was possible to use syllables. The `_` at the end of the code for the grapheme is used to denote a doubtful transcription and it is derived from the reference material.

Our dataset for Cypro-Minoan contains all categories of signs that are known for the writing system: syllabograms, two alleged logograms, numerical signs and punctuation. Out of the 96 signs identified in Olivier, 95 are present in our dataset, with the sole exclusion being a doubtful instance of the hapax sign 083. This dataset, therefore, is representative of the whole Cypro-Minoan corpus.

It is useful, then, to highlight some of the salient characteristics of the dataset beyond the number of documents and images. In particular, an interesting aspect is the number of individual glyphs attested in each archaeological site, shown in Figure 6.1. Crucially, this figure is very different from a count of the number of documents in each site, since in Cypro-Minoan very few documents are very long and constitute a large portion of the dataset. In particular, the figure shows a very high number of signs attested in the

Cypriot site of Enkomi, which is also the site where most of the longer clay tablets were found. Beside Enkomi, the second most prominent site is Ugarit, on the coast of Syria. Another significant contribution in terms of number of signs is from the Kalavasos-Ayios Dhimitrios site, also found on Cyprus. Finally, the rest of the dataset is from other Cypriot sites.
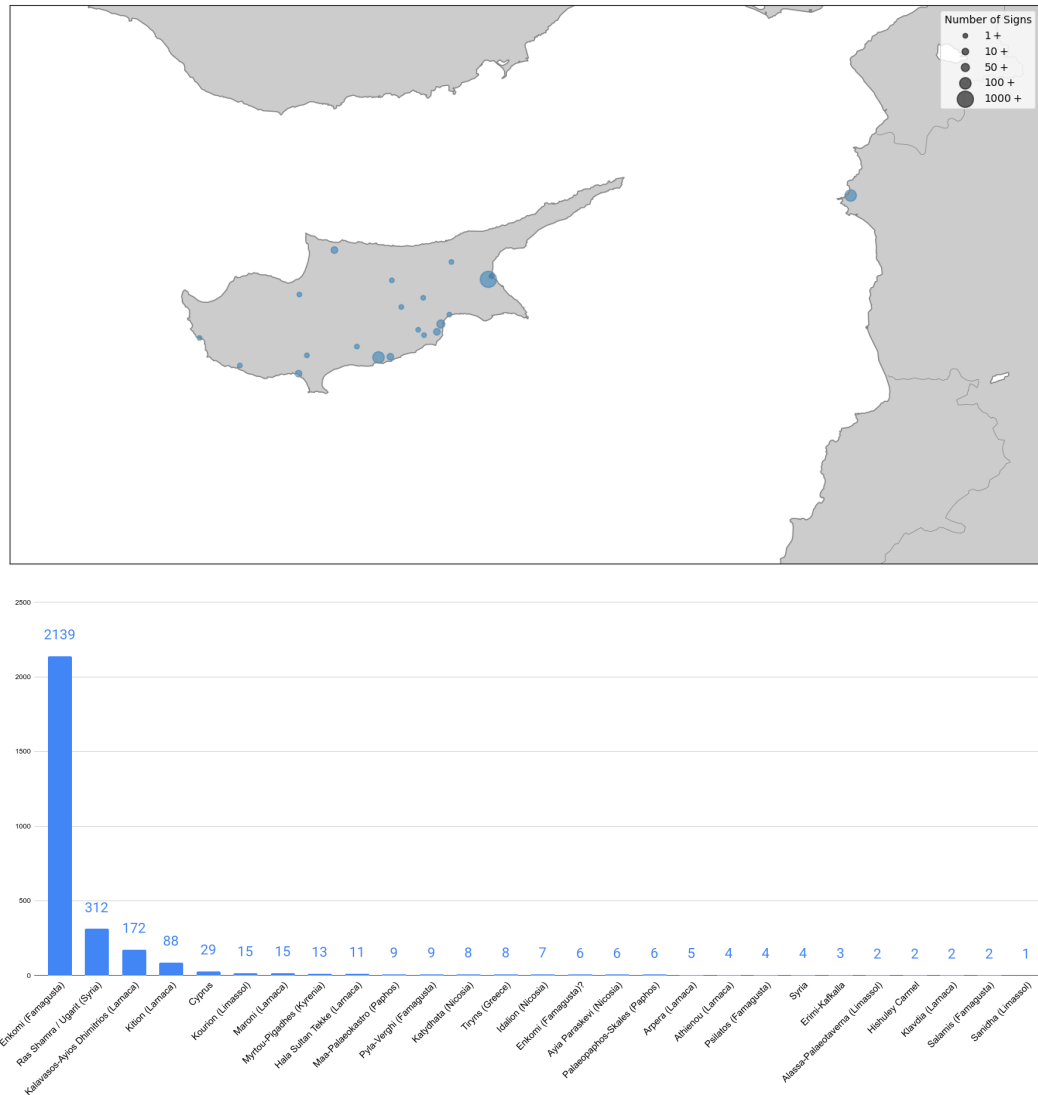


Figure 6.1: Distribution of CM signs in our dataset according to archaeological site.

Using the published readings of signs, it is possible to plot the frequency of the most

frequent alleged grapheme in the dataset, shown in Figure 6.2. From this Figure, an important aspect of the Cypro-Minoan dataset is evident, which is the prominence of sign "DIV", the sequence divisor. Similarly to what was discussed for the Cypriot Greek Syllabary, this sign is used to separate sequences and it always denotes word boundaries. Furthermore, by comparing the prominence of "DIV" in Cypro-Minoan with its usage in the Cypriot Greek Syllabary, it is possible to note an important distinction between the two scripts. In the Cypriot Greek Syllabary the sequence separator "DIV" is attested 239 times, which is comparable with the second most frequent sign "SE", attested 207 times. Meanwhile in Cypro-Minoan, the sequence separator is by far the most frequent sign, with 466 attestations, compared with the 109 attestations for the second most frequent grapheme "023". This discrepancy implies that the sequence separator was used more frequently in Cypro-Minoan than in the Cypriot Greek Syllabary, suggesting that its role as a separator between words is more prominent in the earlier script. Therefore, more documents use "DIV" to separate each individual word in Cypro-Minoan than in the Cypriot Greek Syllabary. Another interesting aspect is the presence of the graphemes "P" and "et", which are also punctuation signs.
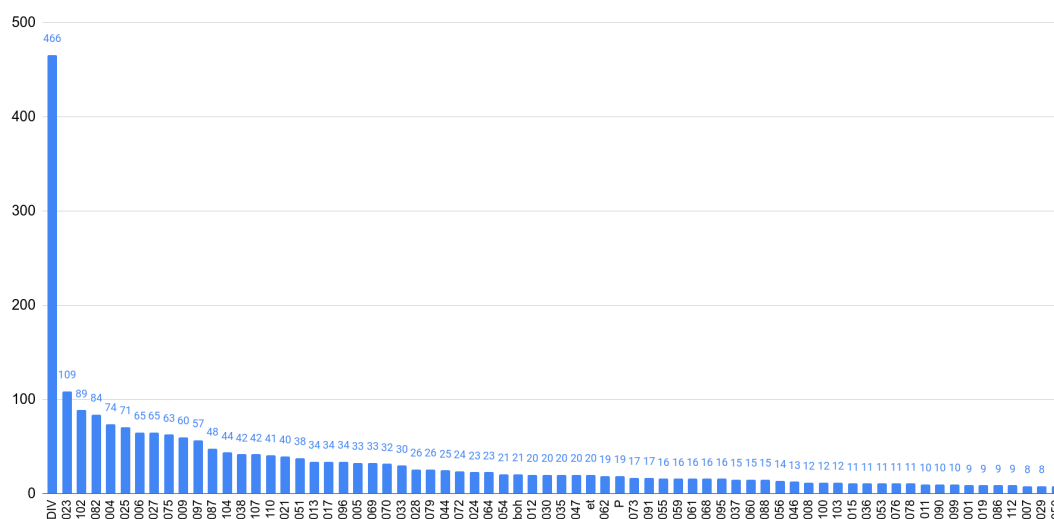


Figure 6.2: Number of attestations of the most frequent 70 alleged graphemes in the Cypro-Minoan dataset.

In order to further examine the statistical patterns emerging from the alleged attes-

tations of each grapheme, it is possible to observe the most frequent grapheme trigrams in the dataset, shown in Figure 6.3.
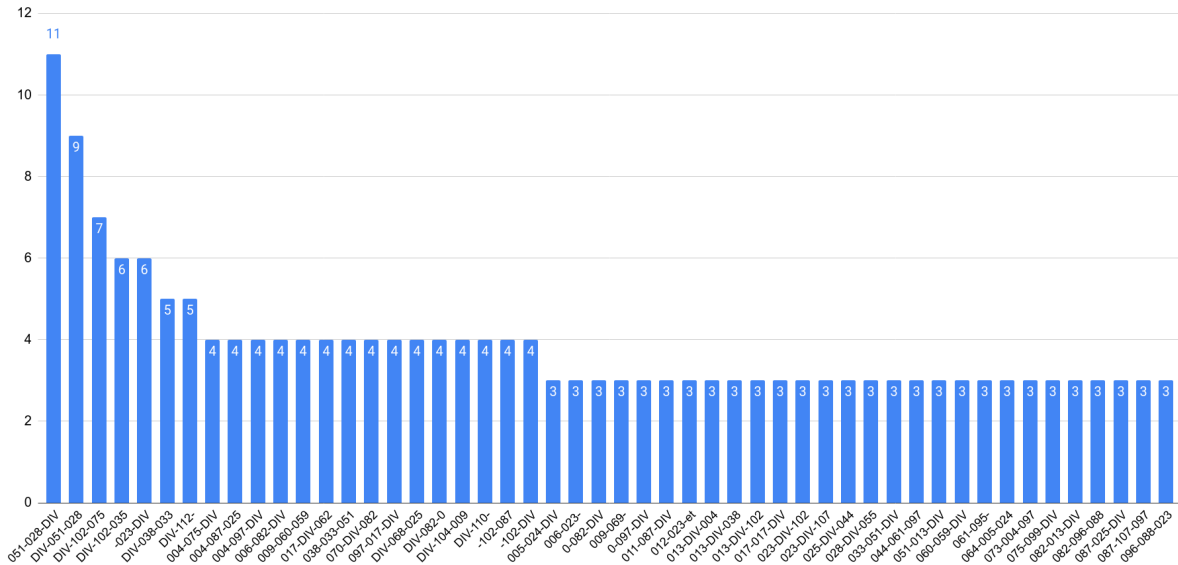


Figure 6.3: Number of attestations of the 50 most attested trigrams of graphemes in the Cypro-Minoan dataset.

Like in the case of the Cypriot Greek Syllabary, the prominence of the word separator and of sequences at the beginning and end of document is also present in the Cypro-Minoan dataset. However, as expected from the high number of attestations of "DIV" in the dataset, the number of three sign sequences containing the sequence divider is even higher in Cypro-Minoan and it constitutes 36 of the 50 most frequent sign trigrams. Including sequences that contain the beginning and end of documents (denoted with a dash with nothing preceding or following it in the Figure), as well as the punctuation sign "et", only 9 out of 50 sequences of three signs are composed uniquely of syllabograms. In Figure 6.4, then, the number of grapheme trigrams having a given number of attestations in the dataset is shown. The situation highlighted by this figure is similar to the one observed for the Cypriot Greek Syllabary, where not many sequences are attested more than once. Furthermore, due to the presence of many damaged signs (excluded in the count of trigrams) the number of complete sequences are even more scarce than for the Cypriot Greek Syllabary. This situation, combined with the prevalence of the sequence

divider in the most frequent trigrams, suggests that the contextual information available from sequences of three graphemes in Cypro-Minoan is very limited. On the other hand, the prominence of the sequence separator allows us to leverage this information, which is useful to determine the prevalence of each grapheme in word initial and word final position.
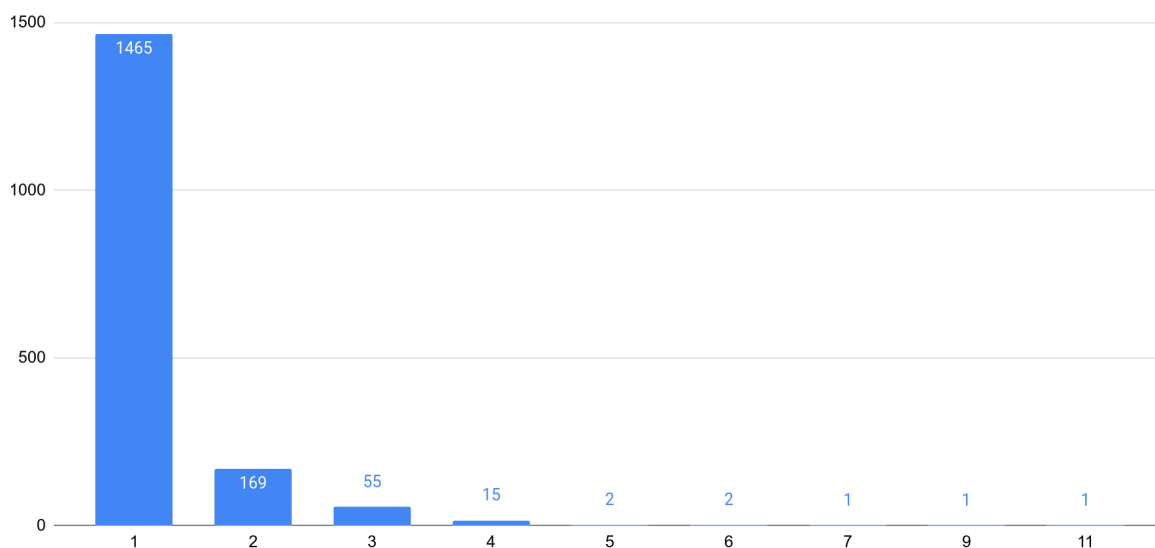


Figure 6.4: Number of grapheme trigrams having a given number of attestations.s
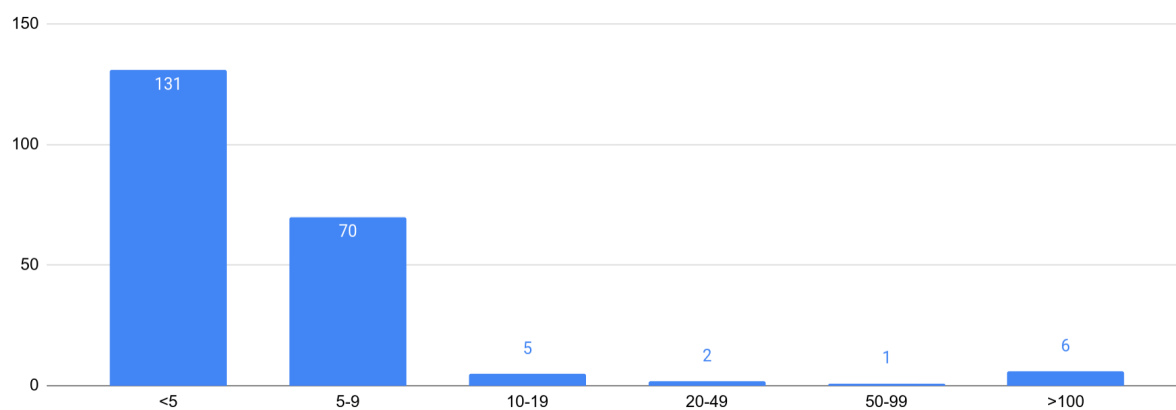


Figure 6.5: Number of documents containing a certain number of glyphs.

One final statistic about the dataset regards the length of the documents, which is a

crucial property to consider when examining the corpus any undeciphered writing system. The presence of longer documents, as a matter of fact, can help decipherment efforts, as it allows a more thorough examination of the patterns in the transcribed language. We show this property in Figure 6.5, where the number of documents within a given range of non damaged signs is shown. Crucially, the number of documents with a high number of attestations is higher in Cypro-Minoan when comparing it with the Cypriot Greek Syllabary. However, the number of inscriptions with less than 5 signs is also higher.

## 6.2   The tripartite division of Cypro-Minoan

One of the more debated aspects of Cypro-Minoan is the alleged division of it in three different subscript, named CM1, CM2 and CM3, which were supposedly used in order to transcribe different languages. This division, first proposed in Masson (1974), is also adopted in Olivier (2007). The three subscripts are defined as follows:

- **CM1**: considered to be the main writing system used in Cyprus throughout most of the Bronze age (ca. 1525-1050 BCE). It is ought to be the most prominent of the three subscript.

- **CM2**: considered to be a variant of CM1, attested only on four clay tablet fragments from Enkomi. Due to the alleged derivative nature of this sub-script, its usage is supposed to be more restricted.

- **CM3**: it was described by Masson as a subscript of CM1, used on the port town of Ugarit, located on the coast of modern Syria. The distinction from the other variants is motivated by the idea that it transcribes the Ugaritic language. Later, Olivier partially altered this definition to include all the Cypro-Minoan documents found in Syria.

The main motivation for the division of Cypro-Minoan in three different subscripts is the presence of signs that are attested in only some of the groups, suggesting that new graphemes were created in order to adapt the sounds of different languages.

| | CM 1 | CM 2 | « CM 3 » | | CM 1 | CM 2 | « CM 3 » | | CM 1 | CM 2 | « CM 3 » |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 001 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 040 | ... | ... | ⟨sign⟩ | 079 | ... | ⟨sign⟩ | ... |
| 002 | ⟨sign⟩ | ... | ⟨sign⟩ | 041 | ⟨sign⟩ | ... | ... | 080 | ... | ⟨sign⟩ | ... |
| 004 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 044 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 081 | ⟨sign⟩ | ⟨sign⟩ | ... |
| 005 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 046 | ⟨sign⟩ | ... | ... | 082 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 006 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 047 | ... | ⟨sign⟩ | ... | 083 | ⟨sign⟩ | ... | ... |
| 007 | ⟨sign⟩ | ... | ⟨sign⟩ | 049 | ... | ⟨sign⟩ | ... | 084 | ⟨sign⟩ | ... | ... |
| 008 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 050 | ⟨sign⟩ | ... | ⟨sign⟩ | 085 | ⟨sign⟩ | ... | ... |
| 009 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 051 | ... | ⟨sign⟩ | ⟨sign⟩ | 086 | ⟨sign⟩ | ... | ... |
| 010 | ... | ⟨sign⟩ | ... | 052 | ... | ⟨sign⟩ | ... | 087 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 011 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 053 | ⟨sign⟩ | ... | ⟨sign⟩ | 088 | ⟨sign⟩ | ... | ... |
| 012 | ⟨sign⟩ | ⟨sign⟩ | ... | 054 | ... | ⟨sign⟩ | ... | 089 | ... | ⟨sign⟩ | ... |
| 012b | ⟨sign⟩ | ... | ... | 055 | ⟨sign⟩ | ... | ⟨sign⟩ | 090 | ... | ⟨sign⟩ | ... |
| 013 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 056 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 091 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 015 | ⟨sign⟩ | ... | ... | 058 | ... | ... | ⟨sign⟩ | 092 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 017 | ⟨sign⟩ | ⟨sign⟩ | ... | 059 | ⟨sign⟩ | ⟨sign⟩ | ... | 094 | ... | ... | ⟨sign⟩ |
| 019 | ⟨sign⟩ | ... | ⟨sign⟩ | 060 | ... | ⟨sign⟩ | ... | 095 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 021 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 061 | ⟨sign⟩ | ⟨sign⟩ | ... | 096 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 023 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 063 | ⟨sign⟩ | ... | ... | 097 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 024 | ⟨sign⟩ | ⟨sign⟩ | ... | 062 | ... | ⟨sign⟩ | ... | 098 | ... | ... | ⟨sign⟩ |
| 025 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 064 | ⟨sign⟩ | ⟨sign⟩ | ... | 099 | ⟨sign⟩ | ... | ⟨sign⟩ |
| 026 | ⟨sign⟩ | ... | ... | 066 | ... | ⟨sign⟩ | ... | 100 | ... | ... | ⟨sign⟩ |
| 027 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 067 | ⟨sign⟩ | ... | ... | 101 | ⟨sign⟩ | ... | ... |
| 028 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 068 | ⟨sign⟩ | ⟨sign⟩ | ... | 102 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 029 | ... | ⟨sign⟩ | ... | 069 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 103 | ⟨sign⟩ | ... | ⟨sign⟩ |
| 030 | ⟨sign⟩ | ⟨sign⟩ | ... | 070 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 104 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 033 | ⟨sign⟩ | ⟨sign⟩ | ... | 071 | ... | ... | ⟨sign⟩ | 105 | ... | ... | ⟨sign⟩ |
| 034 | ⟨sign⟩ | ... | ... | 072 | ⟨sign⟩ | ⟨sign⟩ | ... | 107 | ⟨sign⟩ | ⟨sign⟩ | ... |
| 035 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 073 | ⟨sign⟩ | ... | ⟨sign⟩ | 108 | ⟨sign⟩ | ... | ... |
| 036 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 074 | ... | ⟨sign⟩ | ⟨sign⟩ | 109 | ⟨sign⟩ | ... | ... |
| 037 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 075 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 110 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ |
| 038 | ⟨sign⟩ | ⟨sign⟩ | ⟨sign⟩ | 076 | ... | ⟨sign⟩ | ... | 112 | ⟨sign⟩ | ... | ... |
| 039 | ⟨sign⟩ | ... | ... | 078 | ... | ⟨sign⟩ | ... | 114 | ⟨sign⟩ | ... | ... |

Figure 6.6: The repertoire of Cypro-Minoan signs across the three sub-scripts, as proposed by Olivier (2007).

In particular, Olivier proposed a classification of the signs in all Cypro-Minoan variants that is composed of 96 total syllabograms, of which only 32 are shared between all the three variants of Cypro-Minoan (see Figure 6.6). Other graphemes, then, are not shared across all variants, and some are even unique to one of the three alleged subscripts.

This hypothesized subdivision of Cypro-Minoan in three different variants, however, is not universally agreed upon among scholars and it has been criticized in multiple works (Davis et al., 2014; Valério, 2016; Valério and Davis, 2017; Palaima, 1989; Ferrara, 2012, 2013b; Valério, 2013). In particular, one of the main criticisms moved at the tripartite division regard the fact that very little consideration is given to the different media and techniques used in the corpus, which can profoundly influence the shape of the signs. While it is trivial to distinguish two shapes representing the same phonetic value in deciphered writing systems (for example, lowercase and uppercase letters in the English alphabet), this task is more complex when the script is undeciphered, like Cypro-Minoan. This situation is exacerbated when dealing with inscribed glyphs, as the techniques used to inscribe signs on hard materials like stone or metal differ significantly with the techniques used on softer materials like clay. The shape and size of the object might also play a role, and so can other factors such as the content of the document.

| Sign no. | CM1 | CM2 | CM3 |
|---|---|---|---|
| 070 | | | |
| 087 | | | |
| 092 | | | |
| 088 | | | |
| 089 and 090 | | | |

Figure 6.7: The alleged inconsistency in the criteria used to classify signs 070,087,092 when compared with 088, 089 and 090. Image source: Corazza et al. (2022a)

In addition to these more abstract considerations, some inconsistencies in the clas-

sification of the signs were also highlighted. As an example, see the situation of signs 070,087,092,088 shown in Figure 6.7. In the cases of signs 070, 087, 092, Olivier uses the same criterion to find the correspondences between the CM1 variant of the grapheme and its CM2/3 counterparts. In particular, diagonal lines become vertical and the overall shape of the glyphs is more similar to a rectangle. In the case of signs 088, 089, 090, however, although a similar mechanism seems to be used, which would suggest that these three glyphs are all allographs, they are considered as three different graphemes by Olivier.

For this reason, one of the open questions in the study of Cypro-Minoan regards the presence of allographs that are not consensual in the literature, as these could serve to reduce the number of graphemes that are allegedly found only in some of the variants of Cypro-Minoan.

One final aspect of the tripartite division of Cypro Minoan is how this division is represented in our dataset, where damaged signs have been excluded. In particular, it is interesting to consider the relative size of the three subscripts in the dataset, as well as the length of the documents within them (see Figure 6.8). It is clear from the pie charts that there is a great discrepancy between the number of signs and the number of documents in the three variants. In particular, CM2 is the largest variant in terms of number of signs, but it is comprised only of three long clay tablets from Enkomi. A similar situation is found for CM3, where the number of documents constitutes only 4.2% of the total, but the number of signs in this variant is 10.9% of the total.
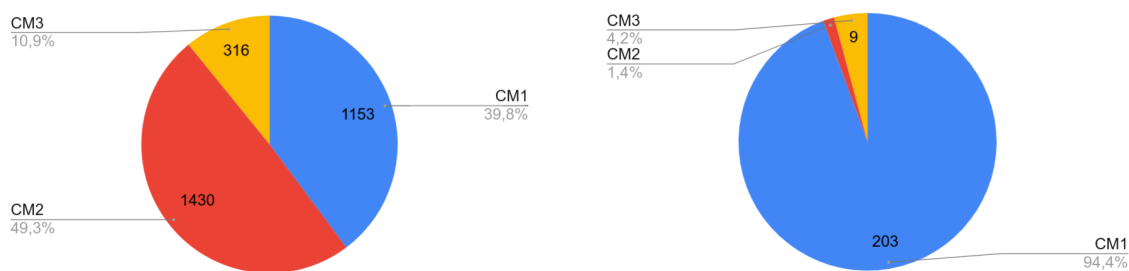


Figure 6.8: The relative size of the three variants of Cypro-Minoan in our datasets. On the left in terms of number of signs. On the right, in terms of number of documents.

This discrepancy constitutes a crucial distinction between these three variants. Fur-

thermore, by analyzing the length of the documents in the three variants (see Figure 6.9) it is clear that while CM1 and CM3 are composed of both small and long documents, CM2 comprises only some of the longest inscriptions in the dataset. This leads to a paradoxical situation, where, considering only the available data, the CM2 variant is the largest one in terms of number of signs, despite the very low number of documents and the fact that it is supposed to be derived from CM1 and to be of limited use. Crucially, the only long inscription in CM3 is also a clay tablet, while the rest of the CM3 documents are relatively short. Finally, the only other two inscriptions that are in CM1 and contain more than 100 signs are two clay cylinders. It is also useful to note that the most attested type of inscription, entirely contained in CM1, is the clay ball. This type of inscription is characterized by a very low number of signs, hence the high number of documents in CM1 with a very low number of signs.
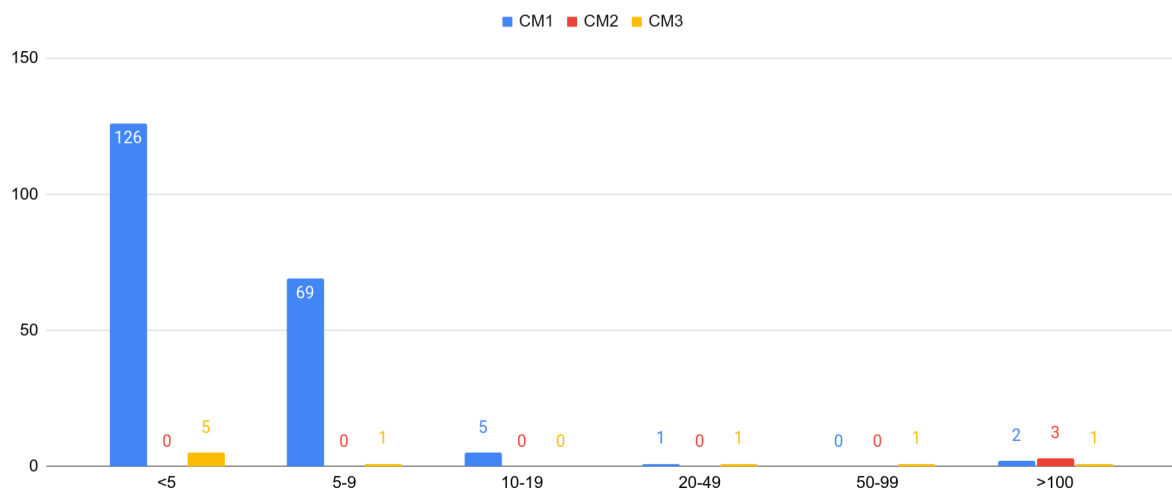


Figure 6.9: Length of the documents belonging to the three variants of Cypro-Minoan

Crucially, the imbalance between number of documents and number of signs in the three alleged variants of Cypro-Minoan mirrors the same imbalance when considering all signs, including those that are damaged. Additionally, despite the exclusion of a number of damaged signs from our dataset, the CM3 portion of the dataset is still significant, as it contains 316 signs, or 10.9% of the total.

## 6.3 From Deciphered to Undeciphered

In the previous chapter, an unsupervised deep learning model for categorizing signs in ancient writing system was developed, incorporating contextual information about the sequences attested in the script. However, this approach was only tested on the Cypriot Greek Syllabary, in order to evaluate the impact of contextual information on the performance of the model using a deciphered writing system. In addition to this, the Cypriot Greek Syllabary was also used to select some hyperparameters for the model, since no such selection can be performed on an undeciphered writing system such as Cypro-Minoan.

Despite the promising results obtained on the Cypriot Greek Syllabary, it is unfortunately not possible to apply the same experimental settings to Cypro-Minoan. The first difference between the two writing systems is also the more obvious one: while the Cypriot Greek Syllabary is completely deciphered and we know it transcribes a dialect of ancient Greek, Cypro-Minoan is undeciphered and we do not know what language it encodes. Furthermore, whether Cypro-Minoan is a single script or a collection of closely related scripts is also a contentious matter. For these reasons, it is not feasible to directly apply a clustering-based approach to this undeciphered script, as any clustering algorithm uses an all-encompassing approach, where all the dataset is categorized. Furthermore, any clustering of the data is bound to contain errors. In such a situation, then, it would be impossible to use the result of a clustering attempt, since in Cypro-Minoan the inventory of signs is not agreed upon, and any errors of this approach would completely invalidate the procedure, since no holistic solution can be adopted as-is.

Another crucial difference between the two scripts is the fact that, while in the Cypriot Greek Syllabary sequences of three syllabograms are relatively more frequent, these are relatively more rare in Cypro-Minoan (see Figures 5.3, 6.3). This suggests that our contextual component of the model might not be adequate to deal with the lower number of complete contexts that are present in Cypro-Minoan. Crucially, the lower number of sequences of three syllabograms is caused, in part, by the extensive use of the sequence separator in Cypro-Minoan, to a higher degree than what is observed in the Cypriot Greek Syllabary (see Figures 5.2, 6.2). This suggests that, while both writing systems sometimes use sequence dividers in order to separate sequences of signs, this practice is

more frequent in Cypro-Minoan. The usage of "DIV", then, can provide us with some information about the position of signs in words, since any sequence separator is surely found between two separate words. Therefore, the signs preceding and following a divisor are in word-final and word-initial position, respectively. As a reminder, the prominence of signs in word-initial and word final positions is a useful tool in order to detect allography, and it has been used in the past as one of the methods to decipher other writing systems such as Linear B.

$$(1 - \lambda_c) \, H_c \left( C \frac{M_s(X_i)}{||M_s(X_i)||}, Y_i \right) \qquad \lambda_c \, H_b(M_s(L_i, R_i), D_i)$$
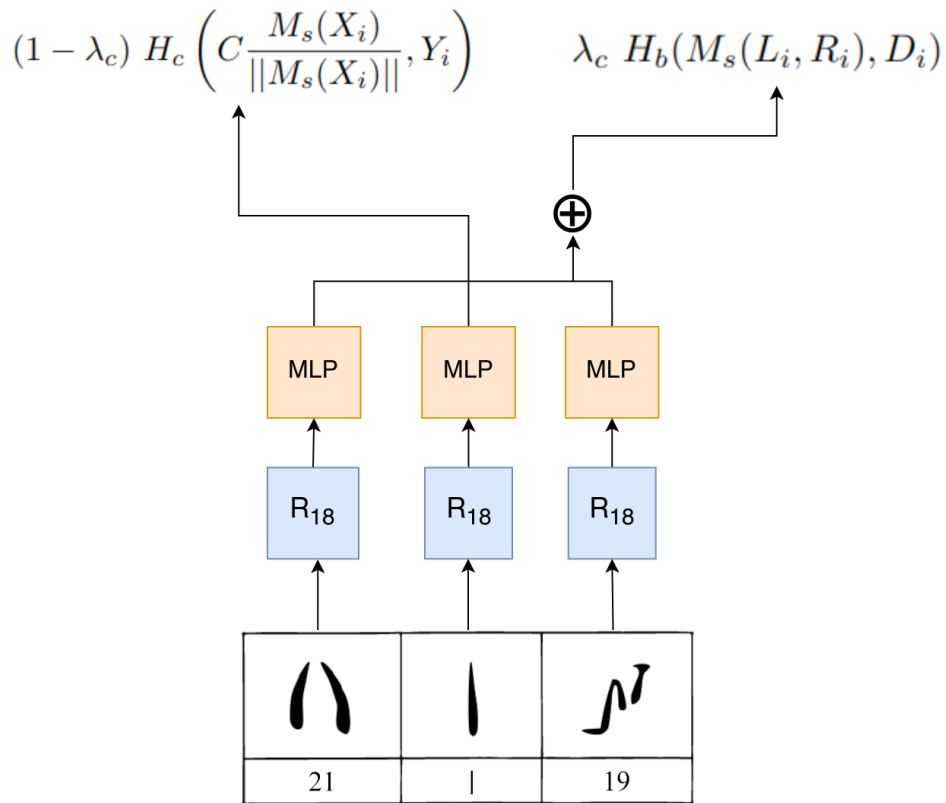


Figure 6.10: Sign2Vec$_d$, our contextual model for Cypro-Minoan. It predicts whether the current sign $X_i$ is a sequence separator, using only the signs to its left and right ($L_i$, $R_i$).

This information about the presence of sequence separators, then, can be used to alter our contextual model Sign2Vec$_c$ in order to apply it to Cypro-Minoan. In order to distinguish the two models, we call this version Sign2Vec$_d$, and we define a new task for the model, which consists in predicting whether the current sign is a sequence separator

using the signs on its left and right (see Figure 6.10). In order to perform this task, the contextual component of the model is defined as follows:

$$M_d(L_i, R_i) = W(M_s(L_i) \oplus M_s(R_i)) + b \qquad (6.1)$$

Where:

- $L_i, R_i$ are the images found on the left and on the right of the i-th sign in the dataset, respectively;

- $M_s$ is the function that applies the ResNet18 and the MLP to the model, defined in equation 5.4;

- $\oplus$ denotes the concatenation between vectors;

- $W \in \mathbb{R}^{2v_s \times 1}$ is a matrix of parameters, used to project the $2v_s$ dimensional vector obtained from the ResNet18 to a single value;

- $b \in \mathbb{R}$ is the bias applied to the final output.

Following the previous equation, then, the model uses the concatenated vector representations of $L_i, R_i$ to output a single value, which in turn predicts whether the current sign $X_i$ is a sequence separator or not. Finally, we can define the contextual component of the loss as follows:

$$H_b(M_d(L_i, R_i), D_i) \qquad (6.2)$$

Where:

- $H_b$ is the binary cross-entropy function;

- $D_i$ is a binary value, indicating whether the i-th sign $X_i$ is a sequence divider.

The complete formula for the model, then, can be expressed as:

$$\mathcal{L}(C, X_i, L_i, R_i, Y_i, D_i) = (1 - \lambda_c) \ H_c\left(C\frac{M_s(X_i)}{||M_d(X_i)||}, Y_i\right) + \lambda_c \ H_b(M_d(L_i, R_i), D_i) \quad (6.3)$$

Where:

- $C$ is the matrix of centroids found during the application of K-Means;

- $Y_i$ is the cluster that was found for the i-th sign $X_i$ during the application of K-Means;

- $H_c$ is the categorical cross entropy;

- $M_s$ is the non contextual component of the model, as defined in equation 5.4

The resulting model, then, aims to leverage information about the attestations of each sign in word initial and word final position in order to determine whether the current sign is a word separator. The only prior information that is used for this procedure is the classification of the sequence separators, which are not debated and they are not part of our analysis, since they are not syllabograms.

In order to teach the model about word initial and word final signs, however, it is crucial to consider also the signs appearing at the beginning and at the end of documents. These situations are common, since the Cypro-Minoan dataset contains a large number of inscriptions that are very short, like the clay balls. When applying $Sign2Vec_c$ to the Cypriot Greek Syllabary we opted to create an artificial context for a sign at the beginning or end of a document using a random sequence separator from the dataset. By applying the same procedure to Cypro-Minoan, we obtain random dividers that represent the beginning or end of documents. However, it is still necessary to provide the model with an image to represent that the document has ended beyond the current sequence separator. For this reason, a completely black image is provided to the model, representing the beginning or end of a document. Crucially, then, the virtual sequence dividers used at the end of documents are considered as normal signs, allowing the model to consider the word boundary found at the beginning or end of inscriptions. Finally, like for the Cypriot Greek Syllabary, we provide random images with dots, populated using the Bridson algorithm for Poisson Sampling. The source code for Sign2Vec and all the experiments that follow can be found at `https://github.com/ashmikuz/sign2vec_d`. The full set of hyperparameters can be found in Table 6.1.

| Hyperparameter | Value |
|---|---|
| Architecture | Resnet18 |
| Base Learning Rate | 4.8 |
| Batch size | 16; |
| Crops for assign | [0] |
| Epochs | 100 |
| Feature dimensions | 128 |
| Final learning rate | 0.0048 |
| Number iterations before prototypes freeze | 300000 |
| Hidden MLP size | 2048 |
| $\lambda_c$ | 0.7 |
| Max scale crops | [1.0, 0.6] |
| Min scale crops | [0.6, 0.4] |
| Number of crops | [6, 10] |
| Number of prototypes | [100,100,100] |
| Size of the crops | [80, 60] |
| Start warmup | 0.3 |
| Temperature | 0.1 |
| Warmup Epochs | 10 |
| Weight decay | $1 \times 10^{-6}$ |

Table 6.1: Sign2Vec$_d$ Hyperparameters

## 6.4 The paleographic separation in the scatter plot

Having described our context-aware model for Cypro-Minoan, the first step in order to assess its effectiveness is to observe the patterns emerging from the three-dimensional scatter plots obtained from Sign2Vec$_d$ and DeepClusterv2. In order to reduce the effect of the random initialization of parameters over the results, all experiments are the result of 20 models trained with randomly initialized weights. The scatter plots, then, are obtained from the concatenation of the vectors from 20 different models, obtaining a vector representation for each sign, of size 2560. Then, t-SNE is applied to the vec-

tors in order to obtain three dimensional representations for each sign. The interactive three-dimensional visualizations for the vectors obtained from both DeepClusterv2 and Sign2Vec$_d$ can be found at `https://corpora.ficlit.unibo.it/INSCRIBE/PaperCM/`.

By observing the two scatter plots, some preliminary analysis can be performed by observing the relative position of signs in the reduced vector space. By examining these visualizations, we discovered that there exists a separation between all signs inscribed on clay tablets (*Tablet*) and signs found on other types of documents (*Other*). While Tablet signs are generally found towards the center of the scatter plot, on the outskirts most of the signs are tablet signs. This result is very peculiar, as it highlights the fact that both of our models produce a degree of separation between the more angular shapes found on tablets, with fewer strokes, and the signs found on other inscriptions, which generally feature more diagonal lines and more strokes. Crucially, while most of the *Tablet* signs are part of the alleged CM2 variant of Cypro-Minoan, CM3 also contains one tablet. The signs found on the CM3 tablet also tends to behave like CM2, showing our model does not appear to favour a tripartite division of Cypro-Minoan, instead featuring a more binary distinction based on the type of support that the signs are inscribed on.

This result might seem detrimental for any further analysis of the graphemes in Cypro-Minoan, since even consensual graphemes are separated in the plot and tend to form either two distinct groups (see Figure 6.12) or "filaments" from the CM2 variants found on the outskirts to the more central *Other* signs. Due to this fact, it is difficult to imagine a successful strategy that manages to group the signs by proximity, in a similar way to what a clustering algorithm would do. This separation between *Tablet* and *Other*, however, results from an unsupervised model, where no prior knowledge about the graphemes is provided to it, with the notable exclusion of the sequence separators. For this reason, this result constitutes independent evidence for a paleographic separation of the signs, where the only factor causing it is a difference in the shape of the signs that depends on the support that they are inscribed on, independently of the tripartite division of the script. Additionally, by following a line that points towards the center of the plot from the outskirts, it is possible to visually reconstruct a relation between the *Tablet* and *Other* variants of at least some of the consensual graphemes, showing that our model does not completely lose the information regarding allography across these
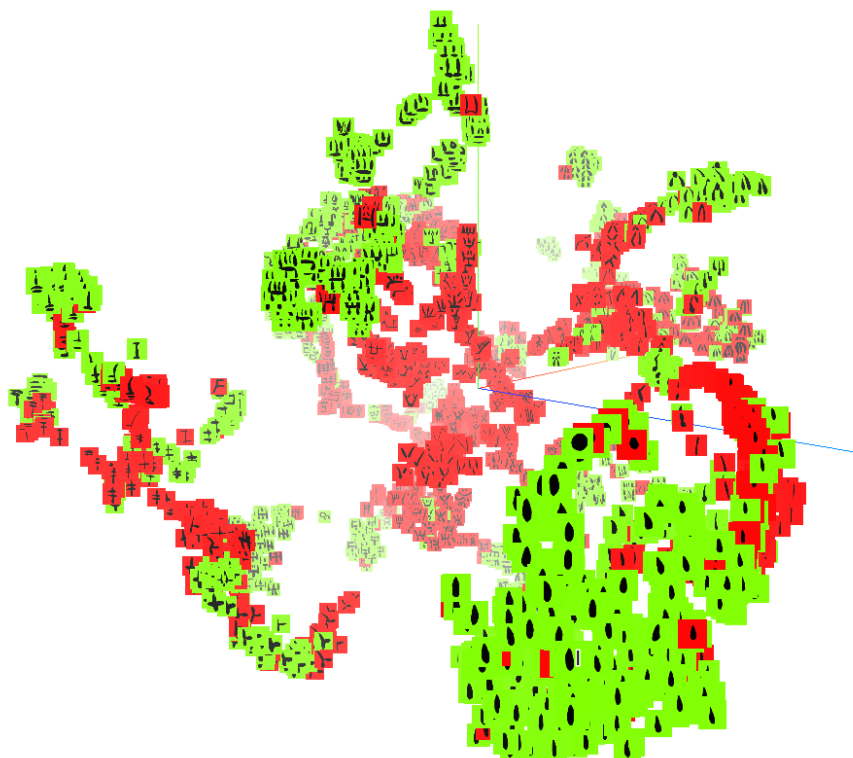
Figure 6.11: The three dimensional scatter plot obtained from Sign2Vec$_\text{d}$, highlighting the apparent separation between signs found on clay tablets (in green) and signs found on other types of documents (in red).

subgroups. If this is indeed the case, it might be possible to devise a system to find correspondences between these subgroups, in order to reconstruct allographs that are not consensually agreed upon in the literature.

## 6.5   Test for possible mistakes in readings

The three-dimensional visualizations created from the models also allow an analysis involving some signs that are miscategorized by the author of the sources of the drawings. By using the interactive visualization that allows the highlighting of each individual grapheme using the published readings, it is possible to spot outliers, which might indicate a mistake in the published reading of some signs. In these instances, through a

Figure 6.12: The separation of grapheme 097 in two distinct groups. On the right, the *Tablet* attestations, on the left the *Other*.

visual inspection, we compiled a list of signs where it is arguable whether the reading of a sign is correct, in addition to a possible revision of said reading. In particular, most of the proposed corrections have already been proposed (Valério, 2016). As an example of an inconsistency within the classification of some signs, see Figure 6.13.
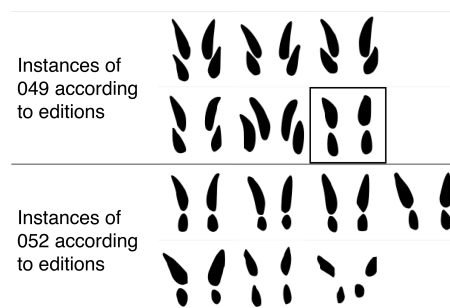


Figure 6.13: An example of an incoherent classification involving sign number 049 and 050. The highlighted sign is categorized as 049 by Olivier, but its shape appears to be more similar to the other 052 in the corpus. Image source: Corazza et al. (2022a).

| Inscription | Sign Position | Published reading | Reference published reading | Proposed correction (Valério 2013, 2016) | Test Result DeepClusterv2 | Test Result Sign2Vec$_d$ |
|---|---|---|---|---|---|---|
| ##011. ENKO Abou 010 | r00.1 | 101 | Olivier (2007) | 102 | 1.0 | 1.0 |
| ##011. ENKO Abou 010 | r00.3 | 013 | Olivier (2007) | 008 | $2.88 \times 10^{-49}$ | $2.49 \times 10^{-45}$ |
| ##014. ENKO Abou 013 | r00.1 | 075 | Olivier (2007) | 073 | $3.41 \times 10^{-153}$ | $8.19 \times 10^{-152}$ |
| ##025. ENKO Abou 022 | r00.3 | 008 | Olivier (2007) | 013 | $2.04 \times 10^{-70}$ | $1.20 \times 10^{-78}$ |
| ##026. ENKO Abou 023 | r00.2 | 075 | Olivier (2007) | 073 | $1.05 \times 10^{-136}$ | $1.05 \times 10^{-126}$ |
| ##026. ENKO Abou 023 | r00.5 | 075 | Olivier (2007) | 073 | $9.71 \times 10^{-131}$ | $6.32 \times 10^{-147}$ |
| ##034. ENKO Abou 031 | r00.2 | 095 | Olivier (2007) | 096 | $1.81 \times 10^{-36}$ | $6.76 \times 10^{-26}$ |
| ##046. ENKO Abou 043 | r00.2 | 068 | Olivier (2007) | 097 | $4.33 \times 10^{-37}1$ | 1.01 |
| ##052. ENKO Abou 049 | r00.2 | 068 | Olivier (2007) | 097 | $6.04 \times 10^{-63}$ | 1.0 |
| ##063. ENKO Abou 060 | r00.2 | 087 | Olivier (2007) | 088 | $4.08 \times 10^{-42}$ | $5.54 \times 10^{-43}$ |
| ##095. ENKO Apes 001 | r00.7 | 064 | Olivier (2007) | 037 | 1.0 | 1.0 |
| ##098. KALA Arou 001 | r15.4 | 070 | Olivier (2007) | 087 | $1.08 \times 10^{-183}$ | $4.82 \times 10^{-166}$ |
| ##108. ENKO Avas 001 | r00.2 | 006 | Olivier (2007) | 009 | $4.14 \times 10^{-97}$ | $7.96 \times 10^{-39}$ |
| ##111. ENKO Avas 004 | r00.4 | 008 | Olivier (2007) | 013 | $3.81 \times 10^{-51}$ | $2.84 \times 10^{-55}$ |
| ##123. IDAL Avas 001 | r00.3 | 068 | Olivier (2007) | 097 | $9.49 \times 10^{-84}$ | 1.0 |
| ##157. MARO Avas 001 | r00.7 | 068 | Olivier (2007) | 097 | $1.35 \times 10^{-69}$ | 0.52 |
| ##179. CYPR Mvas 002 | r00.4 | 008 | Olivier (2007) | 013 | $1.51 \times 10^{-66}$ | $1.60 \times 10^{-47}$ |
| ##194. CYPR? Psce 002 | r00.1 | 068 | Olivier (2007) | 097 | $6.60 \times 10^{-5}$ | 1.0 |
| ##207. ENKO Atab 002.B | r10003d.10 | 072 | Olivier (2007) | 070 | $1.13 \times 10^{-2}$ | 0.97 |
| ##207. ENKO Atab 002.B | r10004d.10 | 049 | Olivier (2007) | 052 | $5.39 \times 10^{-13}$ | $1.40 \times 10^{-5}$ |
| ##208. ENKO Atab 003.A | r05d.1 | 005 | Olivier (2007) | 004 | $9.77 \times 10^{-153}$ | $2.53 \times 10^{-147}$ |
| ##211. RASH Aéti 002 | r00.3 | 064 | Ferrara (2013a) | 086 | $2.47 \times 10^{-25}$ | $1.90 \times 10^{-9}$ |
| ##211. RASH Aéti 002 | r00.3 | 064 | Ferrara (2013a) | 112 | $5.81 \times 10^{-30}$ | $3.10 \times 10^{-7}$ |
| ##211. RASH Aéti 002 | r00.3 | 086 | N/A | 112 | 0.08 | 1.0 |
| ##212. RASH Atab 002 | r02.3 | 102 | Olivier (2007) | 104 | $3.57 \times 10^{-59}$ | $9.01 \times 10^{-61}$ |
| ##215. RASH Atab 004.A | r08.1 | 103 | Olivier (2007) | 102 | 0.20 | $1.99 \times 10^{-13}$ |
| ##215. RASH Atab 004.A | r08.1 | 103 | Olivier (2007) | 102 | $3.32 \times 10^{-37}$ | $6.13 \times 10^{-67}$ |
| ##215. RASH Atab 004.A | r08.1 | 102 | N/A | 024 | N/A | $4.75 \times 10^{-78}$ |
| ##215. RASH Atab 004.A | r10.2 | 008 | Olivier (2007) | 013 | $-3.44 \times 10^{-68}$ | $-3.50 \times 10^{-76}$ |
| ##215. RASH Atab 004.B | r12.3 | 072 | Olivier (2007) | 073 | $1.15 \times 10^{-2}$ | 1.0 |
| ADD##229. ENKO Mins 004 | r00.2 | 027 | Ferrara (2013a) | 025 | 1.0 | 1.0 |
| ADD##233. IDAL Avas 003 | r00.5 | 025 | Ferrara (2013a) | 027 | 1.0 | 1.0 |
| ADD##237. KITI Avas 021 | r00.1 | 006 | Ferrara (2013a) | 102 | 1.0 | 1.0 |
| ADD##237. KITI Avas 021 | r00.2 | 023 | Ferrara (2013a) | 061 | $1.17 \times 10^{-121}$ | $1.42 \times 10^{-102}$ |
| ADD##242. SANI Avas 001 | r00.2 | 082 | Ferrara (2013a) | 053 | $4.48 \times 10^{-61}$ | $1.70 \times 10^{-51}$ |

Table 6.2: Tests for incoherent sign labels on the outputs of DeepClusterv2 and Sign2vec$_d$

In order to test whether our model independently supports these readings, a method involving the cosine distance between signs was used. In particular, given the sign with a possible erroneous reading and the proposed correction, all the cosine distances between

the vector representing the sign and other sign vectors within these two groups are computed. Then, a non-parametric Mann-Whitney U-test is used in order to determine whether the distances between the sign and the proposed correction are generally lower than the ones with the sign and the published reading. Our threshold for statistical significance was set to $p < 0.05$, so if the null hypothesis does not hold we apply the proposed correction to the sign. Finally, since in some instances two corrected readings can be proposed, we first test the current reading against the two corrections. Then, if both those tests are successful, we compare the two readings and apply the one favoured by the statistical test, independently of the p-value.

The results of the application of Mann-Whitney U tests for the correction of possibly mislabeled signs in the dataset are shown in Table 6.2, for both DeepClusterv2 and Sign2Vec$_d$. Since 20 randomly initialized runs of each model were trained, we used all vectors resulting from all runs for the tests. In total, 31 signs were found to be possibly mislabeled, and their published reading was tested against a proposed correction. In total, 26 corrections were applied using DeepClusterv2, while Sign2Vec$_d$ supported the correction of 20 signs. For all subsequent analyses, we used the corrected labels for the aforementioned signs, following the results of each model.

## 6.6   Leveraging the paleographic separation of signs: the paleographic vector

As discussed in section 6.4, one of the findings of the application of both DeepClusterv2 and Sign2vec$_d$ to the Cypro-Minoan dataset was the fact that these models create a separation between the *Tablet* signs and the *Other* signs. However, a relation between signs in these two subgroups still exists in the scatter plot, since it is often possible to reconstruct the complete grapheme by projecting a line from the *Tablet* variant of a sign towards the center of the scatter plot, where the *Other* variant can often be found, and vice versa. For this reason, the next step of our analysis was the intuition that, perhaps, it would be possible to reconstruct this relation in the vector space, using a "Paleographic" vector. This vector would ideally contain the direction that takes an *Other* variant of a

sign to its respective *Tablet* variant. Formally, the paleographic vector $v_p$ is defined as:

$$v_p = \frac{1}{N} \sum_{i=1}^{N} [c(T_i) - c(O_i)] \tag{6.4}$$

$$c(X) = \frac{1}{|X|} \sum_{x \in X} x \tag{6.5}$$

Where:

- $N$ is the number of signs present on both types of documents that are used to calculate the paleographic vector;

- $T_i, O_i$ denote two sets containing the attestations of the i-th sign in tablets and in other documents, respectively;

- $c(X)$ is a function that calculates the centroid (mean) of a set.
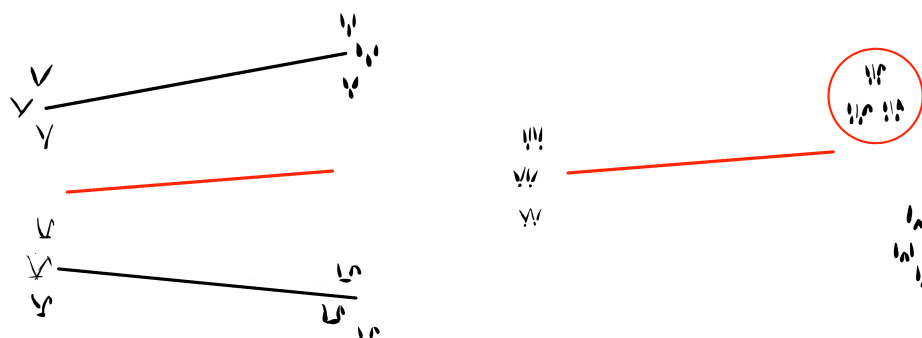


Figure 6.14: On the left: the paleographic vector (red) is first calculated from the difference vector (black) between centroids of signs from clay tablets and centroids of signs from other documents. On the right: the vector is applied to find a correspondence between two variants of the same grapheme in the two subgroups.

As shown in Equation 6.4, the paleographic vector is derived from a set of graphemes that have attestation in both *Tablet* and *Other*, that are used to compute the average *Other→Tablet* direction between the centroids of the two variants of each grapheme in

the two subgroups. If the paleographic vector successfully encoded the *Other* → *Tablet* direction in the vector space, then, we would expect that:

$$\underset{i \in [1, |T|]}{argmin} \; \delta[c(O_j) + v_p, c(T_i)] = i' \quad \implies \quad O_j \approx T_{i'} \tag{6.6}$$

Where:

- $T$ is the set containing all graphemes attested on tablets;

- $\delta$ a distance metric, in this case the cosine distance;

- $\approx$ is used to denote the fact that two sets contain attestations of the same grapheme.

In Equation 6.4, then, by adding the paleographic vector to the centroid of the *Other* attestations of a sign $O_j$, we obtain another vector, which can be compared to the centroid of each of the graphemes that have attestations in *Tablet*, using the cosine distance as a metric. If the application of the paleographic vector is successful, the closest centroid to the vector obtained from the application of the paleographic vector is exactly the one containing the *Tablet* attestations of the same grapheme found in $O_j$. A visual example of the calculation and application of the paleographic vector can be seen in Figure 6.14.

## 6.7  Validation of the paleographic vector

In order to be able to use the paleographic vector, it is necessary to select a set of graphemes, attested in both *Tablet* and *Other*, that are consensual in the literature, in order to calculate the vector from graphemes that are not problematic. Crucially, these graphemes can also be used to validate the approach, as it is possible to try and reconstruct the relation between their *Other* and *Tablet* variants and measure how many of these relations can be reconstructed using the vector.

In order to validate the paleographic vector and to calculate it, then, the following consensual graphemes were used: 001, 004, 005, 006, 008, 009, 011, 012, 017, 021, 023, 024, 025, 027, 028, 033, 036, 037, 038, 044, 061, 069, 070, 075, 082, 087, 096, 097, 102, 104, 107, 110 (see Figure 6.15).

| 001 𐘾 | 004 𐙀 | 005 𐘴 | 006 𐘥 | 008 𐙊 | 009 𐘱 | 011 𐘙 | 012 𐘚 | 017 𐘀 | 021 𐘈 |
| 023 𐘉 | 024 𐘊 | 025 𐘉 | 027 𐘞 | 028 𐘋 | 033 𐘍 | 036 𐘐 | 037 𐘑 | 038 𐘒 | 044 𐘗 |
| 061 𐘢 | 069 𐘪 | 070 𐘫 | 075 𐘰 | 082 𐘶 | 087 𐘹 | 096 𐙁 | 097 𐙂 | 102 𐙅 | 104 𐙆 |
| 107 𐙇 | 110 𐙉 |  |  |  |  |  |  |  |  |

Figure 6.15: The set of 32 signs used to construct and validate the paelographic vector.

This set of consensual signs can then be used to construct the paleographic vector, with the prospect of applying it to non-consensual allographs. Before applying it to non-consensual allographs, however, a validation step is necessary, to assess the ability of the vector to reconstruct *Other → Tablet* relations in the dataset. As a validation task, then, we can proceed by excluding each one of the consensual signs from the construction of the paleographic vector, and then proceed to apply it to its *Other* variant, in order to check whether its *Tablet* variant can be detected. In order to perform a fair assessment of the vector, all conventional graphemes that have at least 2 *Tablet* attestations are candidates to be the closest to the projection obtained using the paleographic vector.

Sign2Vec$_c$ was examined in Chapter 5 and its advantages over DeepClusterv2 have been proven using the Cypriot Greek Syllabary. However, for Cypro-Minoan we use a new method involving the paleographic vector, as well as the new Sign2Vec$_d$ model. For this reason, the validation of the paleographic vector is performed not only for Sign2Vec$_d$, but also for DeepClusterv2, in order to compare their performance on this task. The 2560 dimensional vectors obtained from the concatenation of vectors from all the 20 runs of each model are used during the validation.

| Model | Top-1 Accuracy | Top-2 Accuracy | Top-3 Accuracy | Top-5 accuracy |
|---|---|---|---|---|
| DeepClusterv2 | 0.66 | 0.75 | 0.81 | 0.97 |
| Sign2Vec$_d$ | **0.69** | **0.81** | **0.94** | **1.0** |

Table 6.3: Application of the paleographic vector to consensual graphemes using both DeepClusterv2 and *Sign2Vec$_d$*. In bold, the highest values.

The full results of the validation using consensual graphemes are shown in Tables A.1 and A.2 in the appendix, for DeepClusterv2 and Sign2Vec$_d$ respectively. Addition-

ally, Table 6.3 shows the Top 1,2,3 and 5 accuracy for both models when applying the paleographic vector to reconstruct the *Other* → *Tablet* relations for the 32 consensual graphemes. The table shows that the application of the paleographic vector results in the reconstruction of approximately two thirds of the *Other* → *Tablet* correspondences, while the accuracy jumps to more than 80% and more than 90% when considering the top three candidates, for DeepClusterv2 and Sign2Vec, respectively. Additionally, all the Top 1,2,3 and 5 accuracies favour $Sign2Vec_d$ over the DeepClusterv2 model, showing to a greater extent that incorporating contextual information in the model has a positive effect on the application of the paleographic vector. In order to further evaluate the effectiveness of the method it is possible to use an estimate of how probable it is to obtain the observed results by random chance. Since DeepClusterv2 obtains a worse performance than $Sign2Vec_d$, we use the Top-1 accuracy of the former in this evaluation as a worse-case scenario. This estimate can be achieved by using a binomial distribution over a random variable $X$ as follows:

$$P(X \geq c) = \sum_{k=c}^{n} \binom{n}{k} p^k (1-p)^{n-k} \quad X \sim B(n, p) \tag{6.7}$$

Where:

- $c = 21$ is the number of *Other* → *Tablet* correspondences where the correct sign is the closest to the projection obtained by applying the paleographic vector;

- $n = 32$ is the total number of consensual graphemes, which corresponds to the number of tests performed in the validation;

- $p = \frac{1}{64}$ is the probability of randomly selecting the correct sign from the 64 signs present in *Tablet*.

By applying this evaluation to DeepClusterv2, and considering only instances there the projection obtained using the paleographic vector is closest to the correct *Other* sign of the variant, we obtain $P(X \geq c) = 1.28 * 10^{-30}$. It is also possible to determine the expected value of this distribution, which is given by $n * p = \frac{32}{64}$. Therefore, the expected value of the distribution shows that on average the number of correct correspondences would be just $\frac{1}{2}$; in other words this approach would find a correct correspondence only one half of the times on average.

These probabilistic considerations shows that the positive results from the application of the paleographic vector cannot possibly be due to random chance, since the probability of a model finding 21 out of 32 signs using this method is really low ($p = 1.28 * 10^{-30}$). Through this validation step, then, the application of the paleographic vector has been shown to be viable, since it is able to reconstruct two thirds of the correspondences on the consensual signs. Furthermore, the usage of our contextual Sign2Vec$_d$ model improves performance when applying the paleographic vector. For these reasons, it is now possible to investigate some hypothesized allographs using Sign2Vec$_d$, in order to evaluate whether our model is in agreement with these proposals or not.

## 6.8 Testing allography with the paleographic vector

In Valério (2016) various hypotheses have been put forward questioning the validity of the tripartite division of Cypro-Minoan, by implying that some shapes that are considered as distinct graphemes in the three variants of Cypro-Minoan are actually allographs. In order to test whether the application of the paleographic vector supports these hypothesized mergers, we used two separate sets of tests:

- **Type 1**: test for signs in complementary distribution. In this situation, one of the two hypothesized allographs is present only in CM1, the other is peculiar to CM2;

- **Type 2**: tests for the merger of two signs where one is scarcely attested and present only in CM1, while the other is present in all subcorpora.

For both of these types of tests, then, the goal is to use the paleographic vector to check for correspondences. The tests of type 1 are shown in Figure 6.16. Notice that, despite the fact that our vector uses a correspondence between *Other* and *Tablet*, the type 1 tests consider the three subcorpora as a starting point and also include signs with some tablet CM3 attestations. This is motivated by the fact that the CM3 variant is relatively scarcely attested and it sometime does not mirror the distinction between the shapes of CM1 to the more squarish ones found in CM2. For this reason, we allowed situations where there are some tablet attestations of signs that are otherwise only found on CM1,

| | |
|---|---|
| 013 $\bar{\lambda}$ (CM1) | 078 (CM2) |
| 019 (CM1/3) | 079 (CM2) |
| 034 (CM1) | 056 (CM2) |
| 039 (CM1) | 049 (CM2) |
| 041 (CM1) | 010 (CM2) |
| 046 $\bar{\lambda}$ (CM1) | 047 (CM2) |
| 050 (CM1) | 051 (CM2) |
| 053 (CM1) | 054 (CM2) |
| 055 (CM1) | 054 (CM2) |
| 064$_a$ (CM1) | 062 (CM2) |
| 073 (CM1) | 076 (CM2) |
| 088 (CM1) | 089 + 090 (CM2) |
| 099 (CM1) | 064$_b$ (CM2) + 100 (CM2/3) |

Figure 6.16: Pairs of signs for the type 1 tests for signs in complementary distribution.

in order to test for their alleged CM2 allograph by using the paleographic vector. Some tests include further considerations:

- One of the tests involves 053 (CM1/CM3), 054 (CM2) and 055 (CM3). This situation requires the usage of two different tests using the paleographic vector, in particular 053 → 054 and 053 → 055. Therefore, two possible test results are valid.

- The classification of sign 064 is also debated. In particular, Masson (1974) treated it as two separate graphemes for the two variants, 064 (CM1) and 064 (CM2). Olivier considered them as a single grapheme instead. Based on the position of these signs in words, it has been argued that the two are indeed separate graphemes, with the following proposed allographs: the 064 in CM1 (henceforth 064$_a$) was suggested as an allograph of 062 (CM2); the 064 in CM2 (henceforth 064$_b$) was hypothesizes to be an allograph of 099 (CM1/3). We tested both of these possible mergers, by treating 064$_a$ and 064$_b$ as distinct entities during the application of the paleographic vector.

- 089 (CM1), 090 (CM1) are hypothesized to be allograph and to represent the same grapheme as 088 (CM2). Both tests for 089 → 088 and 090 → 088 were performed.

| Other sign | First 10 Tablet signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 013 | 029 | 074 | 010 | **078** | 040 | 076 | 095 | 079 | 047 | 080 |
| Cosine Distance | 0.32 | 0.59 | 0.61 | **0.65** | 0.81 | 0.81 | 0.82 | 0.89 | 0.93 | 1.05 |
| 019 | 010 | 078 | 029 | **079** | 074 | 090 | 100 | 047 | 062 | 089 |
| Cosine Distance | 0.35 | 0.52 | 0.55 | **0.59** | 0.64 | 0.65 | 0.69 | 0.78 | 0.79 | 0.82 |
| 034 | 049 | 029 | 054 | 010 | 090 | 051 | 080 | 089 | 040 | 060 |
| Cosine Distance | 0.33 | 0.39 | 0.53 | 0.57 | 0.61 | 0.65 | 0.66 | 0.66 | 0.69 | 0.79 |
| 039 | **049** | 040 | 080 | 029 | 054 | 090 | 047 | 079 | 078 | 010 |
| Cosine Distance | **0.41** | 0.64 | 0.65 | 0.67 | 0.70 | 0.72 | 0.85 | 0.91 | 0.92 | 0.93 |
| 041 | **010** | 074 | 029 | 078 | 079 | 076 | 062 | 090 | 100 | 089 |
| Cosine Distance | **0.18** | 0.36 | 0.65 | 0.84 | 0.87 | 0.89 | 0.93 | 0.94 | 0.95 | 0.96 |
| 046 | 079 | **047** | 078 | 080 | 029 | 056 | 040 | 049 | 010 | 062 |
| Cosine Distance | 0.35 | **0.36** | 0.49 | 0.53 | 0.57 | 0.67 | 0.76 | 0.88 | 0.88 | 0.91 |
| 050 | 080 | 056 | **051** | 062 | 049 | 090 | 047 | 054 | 079 | 064$_b$ |
| Cosine Distance | 0.10 | 0.13 | **0.29** | 0.33 | 0.41 | 0.45 | 0.45 | 0.47 | 0.48 | 0.52 |
| 053 | **054** | 051 | 049 | 056 | 080 | 062 | 064$_b$ | 090 | 089 | 060 |
| Cosine Distance | **0.14** | 0.20 | 0.24 | 0.26 | 0.35 | 0.42 | 0.52 | 0.56 | 0.58 | 0.68 |
| 055 | **054** | 049 | 051 | 056 | 080 | 062 | 090 | 064$_b$ | 089 | 010 |
| Cosine Distance | **0.17** | 0.37 | 0.47 | 0.50 | 0.58 | 0.66 | 0.77 | 0.78 | 0.79 | 0.82 |
| 064$_a$ | **062** | 100 | 090 | 060 | 056 | 064$_b$ | 089 | 051 | 054 | 079 |
| Cosine Distance | **0.27** | 0.41 | 0.47 | 0.51 | 0.52 | 0.53 | 0.54 | 0.66 | 0.74 | 0.78 |
| 073 | **076** | 074 | 095 | 062 | 060 | 078 | 064$_b$ | 079 | 056 | 047 |
| Cosine Distance | **0.22** | 0.34 | 0.49 | 0.55 | 0.61 | 0.62 | 0.64 | 0.69 | 0.70 | 0.71 |
| 088 | **090** | **089** | 060 | 100 | 010 | 064$_b$ | 062 | 029 | 076 | 040 |
| Cosine Distance | **0.20** | **0.33** | 0.35 | 0.54 | 0.55 | 0.56 | 0.57 | 0.63 | 0.64 | 0.65 |
| 099 | **064$_b$** | 089 | 062 | 060 | **100** | 090 | 076 | 051 | 056 | 054 |
| Cosine Distance | **0.20** | 0.23 | 0.24 | 0.27 | **0.31** | 0.32 | 0.52 | 0.58 | 0.60 | 0.73 |

Table 6.4: Results for tests of type 1 for hypothesized mergers of signs in complementary distribution. The hypothesized targets are in bold and with a blue background.

The tests of type 1 were performed by projecting the vector from the 13 shapes attested in CM1 and comparing the result with a set of acceptable targets, consisting of signs restricted to the *Tablet* subset in our dataset (010, 029, 040, 047, 049, 051, 054, 056, 060, 062, 064b, 074, 076, 078, 079, 080, 089, 090, 466 095, 100). The results of this evaluation are shown in Table 6.4.

The application of the paleographic vector resulted in 8/13 positive matches using the type 1 tests. In order to estimate the probability of this result being due to random chance, we applied the Poisson binomial distribution using the algorithm presented in Hong (2013). The choice of this distribution and not the binomial distribution is motivated by the fact that two of our tests involve two possible targets (088→089/090 and 099→064$_{\text{b}}$/100), so the probability of a positive result varies for different tests. The resulting probability for having 8 successful matches or more by chance is $1.0 * 10^{-7}$.

Using the Poisson distribution, is also possible to calculate the mean of the resulting distribution, by summing the probability of each successful event:

$$\sum_{i=0}^{n} p_i = \frac{15}{20} = \frac{3}{4} \tag{6.8}$$

These values, then, show that the number of matches found by the paleographic vector cannot be attributed to chance, due to the low probability of obtaining these results using a random baseline. In fact, a random baseline would yield a single correct match only 3/4 of the times, on average. It is also interesting to note that in the case of the hypothesized merger 088→089/090, both of the proposed mergers are found as the closest matches resulting from the application of the paleographic vector. It can be argued, then, that the model supports both mergers in this test. Another interesting result involves whether 064$_{\text{a}}$ and 064$_{\text{b}}$ should be considered as a single grapheme. The application of the paleographic vector indicates that 064$_{\text{a}}$ is closest to 062, while 064$_{\text{b}}$ is found only in 6$^{\text{th}}$ position when applying the vector. This result appears to support the fact that 064$_{\text{a}}$ and 064$_{\text{b}}$ should be considered as distinct graphemes, and that 064$_{\text{a}}$ is probably an allograph of 062, while 064$_{\text{b}}$ is probably an allograph of 099.

With respect to the tests of type 2, three hypotheses were tested, involving possible matches between 015,085 and 101 (CM1) as possible allographs of 021,096 and 101 (CM1,CM2,CM3) respectively (see Figure 6.17). In order to fairly assess these types of

| 015 ⟨⟩ (CM1) | 078 ٨ (CM1/2/3) |
|---|---|
| 085 ⅋ (CM1) | 078 ⅁ (CM1/2/3) |
| 101 Ⴘ (CM1) | 102 ⊕ (CM1/2/3) |

Figure 6.17: Pairs of signs for the type 2 tests involving a shape attested only in CM1 and a shape attested on all subcorpora.

tests, we include all signs attested on clay tablets as possible targets for the test, in contrast to the more restrictive criterion used in the type 1 tests for signs in complementary distribution.

The results of these tests, shown in Table 6.5, are largely positive, since 2/3 of the hypothesized mergers are supported by the application of the paleographic vector. In this case, we considered some targets as impossible given a CM1 sign, since they appear on the same document as the CM1 sign while having a distinct shape, suggesting that they cannot be allographs of the CM1 sign.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 015⟨⟩ | **021 ٨** | 029ʌ | 012 ໒ | ~~023٨~~ | 028 ໗ | ~~024ດ~~ | ~~033ⴽ~~ | 010ⵎ | 030ⴀ | ~~091Ⴎ~~ |
| Cosine Distance | **0.13** | 0.17 | 0.31 | 0.39 | 0.46 | 0.53 | 0.56 | 0.59 | 0.60 | 0.64 |
| 085⅋ | 097ⴲ | 095ⴴ | **096⅁** | ~~082ⴸ~~ | 075ⵀ | 008ⵟ | 070ⴹ | 076ⴺ | 051ⴼ | 069ⴹ |
| Cosine Distance | 0.25 | 0.32 | **0.32** | 0.46 | 0.49 | 0.51 | 0.53 | 0.54 | 0.55 | 0.57 |
| 101Ⴘ | ~~104ⴷ~~ | ~~044ⵜ~~ | **102⊕** | ~~110ⴘ~~ | ~~004ⴻ~~ | ~~107ⴱ~~ | 056Ⅲ | 051ⴼ | 068ⴲ | ~~082ⴸ~~ |
| Cosine Distance | 0.15 | 0.22 | **0.35** | 0.36 | 0.41 | 0.43 | 0.48 | 0.50 | 0.51 | 0.52 |

Table 6.5: Results of the tests of type 1, involving signs attested only in CM1 and signs attested in all subcorpora. Hypothesized targets are in bold and with a blue background. Some signs are stricken through to denote impossible matches, where the CM1 sign and the target are attested on the same document.

Once again, by applying the Poisson binomial distribution and considering impossible mergers when calculating the probabilities, we obtain a p-value of $1.7*10^{-3}$ of obtaining 2 or more positive matches. Moreover, the mean value of the distribution is 0.07, meaning

that a single positive result would be obtained by random chance only 7% of the times. These results show that, despite a lower number of type 2 tests, the application of the paleographic vector results in a number of matches that cannot be due to random chance.

The combination of Sign2Vec$_d$ with the paleographic vector applied to both type 1 and type 2 tests confirmed 68.75% of the alleged allographs, which is a figure that is close to the 69% top one accuracy obtained on consensual graphemes during the validation step. These results, in addition to their statistical improbability when compared with a random baseline, strongly suggest that the traditional classification of the Cypro Minoan signs can be integrated with some allographs, as well as the separations of 064$_a$ from 064$_b$.

## 6.9    Conclusions

The combination of a completely unsupervised model and an ad-hoc technique that leverages the observed separation between *Tablet* and *Other* sign was successful, since the technique is able to confirm most of the hypotheses of allography for Cypro-Minoan. This success results from a model which has no knowledge of Cypro-Minoan beyond the shape and sequences of signs that are present in the corpora, while the paleographic vector uses only consensually agreed upon graphemes, that are not viewed as problematic in the literature. For this reason, it can be argued that the results are not biased by any hypothesis on the structure of Cypro-Minoan, nor on the alleged three variants CM1, CM2 and CM3.

The application of the paleographic vector, as well as the observations on the separation of *Tablet* and *Other* signs in the scatter plot also has some interesting implications for the proposed tripartite division of Cypro-Minoan. In particular, it can be noted that the division of signs in the scatter plot does not exactly follow the CM1/2/3 division, since the main differentiating factor appears to be whether each sign was inscribed on a clay tablet or not. Furthermore, the application of the paleographic vector to alleged allographs confirms approximately 69% of the hypotheses. These two factors are strong evidence for the absence of a tripartite division in the form envisioned by Olivier, since the number of signs that are unique to only one of the variants was reduced, with the possibility that more allographs are missing even from this revised version of the syllabary.

This means that the number of graphemes that are unique to each variant was at the very least overestimated, which would be consistent with the idea that Cypro-Minoan is a single writing system, with a high variation in the shape of signs that is mostly explained by the different type of support they are inscribed on.

What could not be verified using Sign2Vec$_d$ and the paleographic vector, then, is whether more than one language is transcribed in Cypro-Minoan. In order to determine whether there is any evidence for this, however, a prerequisite is the finalization of the sign inventory, combined with an analysis of the sequences present in Cypro-Minoan. This analysis, however, might still be partial, due to the very limited number of long inscriptions in Cypro-Minoan, which might not allow any definitive answer on the presence of more than one language in the corpus. Our approach, then, can be used as a starting point for any further analysis on Cypro-Minoan and, while it is not able to prove any linguistic property of the language inscribed in Cypro-Minoan, the contextual nature of Sign2Vec$_d$ does incorporate some of the patterns emering from the sequences of syllables in the language(s) in the model. For this reason, if the sequences of signs in the three subcorpora differed significantly, we would expect the three groups to be separate in our scatterplots, which is not the case. Despite this result, our model cannot reach conclusive results regarding the possibility that CM was used to record one or more languages.

# Chapter 7

# Conclusions

This thesis presented the results of the application of computational methods to two very different kinds of open problems in the study of undeciphered scripts. In particular, the first contribution concerns the decipherment of the fraction signs in Linear A using a combination of constraint programming, ad-hoc metrics and paleographic considerations. Thanks to this approach, we were able to produce a proposal for decipherment that is grounded in the mathematical properties of the fraction system for most signs, while a few tentative decipherments of other signs also use some paleographic considerations. This result shows that the application of more rigorous approaches such as constraint programming can prove very useful when dealing with systems numerical notation, where the assignment of numerical values to signs is bound by very strict rules that can be expressed as constraints. While this step guarantees that any results respect these constraints, it is through the application of ad-hoc metrics that measure the quality of each possible solution that we were able to reach our conclusions. To the best of our knowledge, this is the first attempt at decipherment using such an approach, and its success could possibly be replicated when attempting a decipherment of systems of numerical notation with well defined rules.

The second part of the thesis, then, is focused on the development and application of an unsupervised deep learning model to the signs of Cypro-Minoan, in order to investigate situation of alleged allography and to test the hypothesis that Cypro-Minoan is constituted by three separate scripts. These tests on possible allographs proved fairly

successful, since we were able to reconstruct relations between alleged allographs by learning a Paleographic vector that encodes the *Other→Tablet* direction in the vector space obtained from the model. This procedure resulted in an accuracy value of approximately 69% across all experiments, mirroring some preliminary results obtained during a validation step. The usage of a completely unsupervised method for the study of the inventory of signs in an undeciphered writing system is, to the best of our knowledge, the first of its kind. Its success proves that even a completely unsupervised model can be applied in a scenario where only a scarce amount of data is available. From this experience, it is also possible to derive some knowledge on how to develop such systems in the future. The first step when building a system for undeciphered scripts is the usage of a deciphered writing system as a proxy, in order to allow the evaluation of the performance of the system. Ideally, such a script should be as close as possible to the undeciphered writing system that the model is targeted at, in order to reduce the procedures need it to adapt it to the latter. While the two Cypriot writing systems used during the development of the two Sign2Vec models are closely related, a crucial aspect of the development process was the consideration of the peculiar characteristics of each script, which was achieved in this case through invaluable discussions with paleographers with extensive knowledge of these two scripts and Cypro-Minoan in particular. In our experiments, the scarce number of non damaged sign trigrams in Cypro-Minoan lead to the development of a distinct model for it called Sign2Vec$_d$. The careful consideration of the specificities of each script can then be used to determine what is the main scientific question that needs to be answered using which automatic method, since it is presently unfeasible to build a system that is able on its own to build a perfect categorization of an undeciphered script in the graphemes that compose it. From a computational point of view, then, an important aspect of our pipeline is the data augmentation that is used in all the models, since it is very beneficial when dealing with a very limited amount of data. The usage of visual aids that can be provided to experts in order to evaluate the vectors produced by the neural network can also prove an important step, as it can lead to insight about the behaviour of the system, especially in a multi-disciplinary team.

In this thesis we showed that through a rigorous application of unsupervised deep learning models, it is possible to derive crucial insight in open questions in the field,

without subscribing to any hypothesis on the nature of the scripts a priori. Furthermore, this thesis proves that it is possible and desirable to combine techniques from the fields of Computer Vision and Natural Language Processing to produce methods that incorporate both the statistical patterns that are present in any system of writing, as well as the graphical representation of each sign. Through this combination it might be possible to further advance the state-of-the-art in this field, by adapting these techniques to other writing systems while considering their peculiarities. In fact, after the development of Sign2Vec$_c$ and Sign2Vec$_d$ models and the application of the latter to Cypro-Minoan, one of the future prospects for our methodology is their application in a comparative study between the writing systems of the Aegean and Cyprus. In particular, in Valério (2016) a comparative analysis of the signs of Linear A, Cypro-Minoan and the Cypriot Greek Syllabary is performed, in order to derive some possible phonetic values for these signs. While this experiment has not yet been performed, it might be possible to adapt one of our models and apply it to these three writing systems in order to assess whether it is able to reconstruct the correspondence between similarly shaped signs. Since the chronology of the three systems puts the usage Cypro-Minoan between the early Linear A and the late Cypriot Greek Syllabary, it stands to reason that, if corresponding graphemes could be found in the three scripts and the phonetic values for these corresponding graphemes in Linear A and the Cypriot Greek Syllabary were similar or the same, this result could be used to derive phonetic values for Cypro-Minoan as well.

Thought our experiments on Cypro-Minoan, a revised version of the syllabary was constructed, which includes the mergers between the allographs that were reconstructed using the paleographic vector. This new version of the syllabary contains less signs than the one proposed by Olivier and widely accepted in the literature. For this reason, a possible prospect in the application of computational methods to Cypro-Minoan could feasibly start from the symbolic representation of signs in sequences, where each sign is replaced with the code assigned to the grapheme it represents. In this context, it might be possible to compare the statistical patterns emerging from Cypro-Minoan with those emerging in multiple plausible languages, in an effort to attempt to reconstruct which language(s) the Cypriot script transcribes.

In conclusion, the results of this thesis, as well as multiple contributions from other

authors described in section 3.3, pave the way for new techniques for the study of ancient writing systems, as well as undeciphered scripts. In particular, while our contributions regard the decipherment of the Linear A fraction signs and an investigation of the Cypro-Minoan syllabary, other scholars have successfully applied computational methods to multiple open problems, including scribal hands attribution, topic modeling for undeciphered scripts and the reconstruction of damaged inscriptions. With multiple different methods and tasks, the application of computational methods to paleography still shows great potential and the advancements in the field of natural language processing and computer vision could lead to new results in the study of ancient scripts.

# Bibliography

Assael, Y., Sommerschield, T., and Prag, J. (2019). Restoring ancient text using deep learning: a case study on Greek epigraphy. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6367–6374. Association for Computational Linguistics.

Assael, Y., Sommerschield, T., Shillingford, B., Bordbar, M., Pavlopoulos, J., Chatzipanagiotou, M., Androutsopoulos, I., Prag, J., and de Freitas, N. (2022). Restoring and attributing ancient texts using deep neural networks. Nature, 603(7900):280–283.

Bennett, E. L. (1950). Fractional quantities Minoan bookkeeping. American Journal of Archaeology, 54(3):204–222.

Bennett, E. L. (1980). Linear A fractional retractation. Kadmos, 19(1):12–23.

Bennett, E. L. (1999). Minos and Minyas: Writing Aegean measures. In Deger-jalkotzy, S., Hiller, S., and Panagl, O., editors, Floreant Studia Mycenaea I. Austrian Academy of Sciences Press.

Berg-Kirkpatrick, T. and Klein, D. (2011). Simple effective decipherment via combinatorial optimization. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 313–321.

Born, L., Kelley, K., Kambhatla, N., Chen, C., and Sarkar, A. (2019). Sign clustering and topic extraction in Proto-Elamite. In Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature, pages 122–132.

Born, L., Kelley, K., Monroe, M. W., and Sarkar, A. (2021). Compositionality of complex graphemes in the undeciphered Proto-Elamite script using image and text embedding models. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4136–4146.

Bridson, R. (2007). Fast poisson disk sampling in arbitrary dimensions. SIGGRAPH sketches, 10(1):1.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In European Conference on Computer Vision.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In Proceedings of Advances in Neural Information Processing Systems (NeurIPS).

Casabonne, O., , and Egetmeyer, M. (2002). À propos du sceau de Diweiphilos. In Notes ciliciennes, volume 10 of Anatolia Antiqua, pages 177–181. Institut français des études anatoliennes.

Cash, R. and Cash, E. (2012). La tablette HT 123: une comptabilité en linéaire A. Kadmos, 50(1):33–62.

Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. Journal of Machine Learning Research, 11(3).

Chikamatsu, N., Yokoyama, S., Nozaki, H., Long, E., and Fukuda, S. (2000). A Japanese logographic character frequency list for cognitive science research. Behavior Research Methods, Instruments, & Computers, 32(3):482–500.

Corazza, M. (2022). Unsupervised deep learning for ancient Aegean scripts: from deciphered to undeciphered. Lingue e linguaggio, Rivista semestrale, 2/2022:311–331.

Corazza, M., Ferrara, S., Montecchi, B., Tamburini, F., and Valério, M. (2021). The mathematical values of fraction signs in the Linear A script: A computational, statistical and typological approach. Journal of Archaeological Science, 125:105214.

Corazza, M., Tamburini, F., Valério, M., and Ferrara, S. (2022a). Contextual unsupervised clustering of signs for ancient writing systems. In Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages, pages 84–93.

Corazza, M., Tamburini, F., Valério, M., and Ferrara, S. (2022b). Unsupervised deep learning supports reclassification of Bronze Age Cypriot writing system. PloS one, 17(7):e0269544.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303–314.

Davis, B., Maran, J., and Wirghová, S. (2014). A new Cypro-Minoan inscription from Tiryns: TIRY Avas 002. Kadmos, 53(1-2):91–109.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Duhoux, Y. (2012). The most ancient Cypriot text written in Greek: The Opheltas' spit. Kadmos, 1(51):71–91.

Egetmeyer, M. (2010). Le dialecte grec ancien de Chypre. Tome I: Grammaire; Tome II: Répertoire des inscriptions en syllabaire chypro-grec. De Gruyter.

Egetmeyer, M. (2016). Appendix V: A bronze bowl from Palaepaphos-Skales with a new Cypro-Minoan inscription from the Cypro-Geometric period. In Palaepaphos-Skales. Tombs of the Late Cypriot IIIB and Cypro- Geometric Periods (Excavations of 2008 and 2011), pages 131–136. The Cyprus Institute.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, page 226–231. AAAI Press.

Evans, A. (1909). Scripta Minoa: the written documents of Minoan Crete, with special reference to the archives of Knossos, volume 1. Clarendon Press.

Evans, A. J. (1899). Knossos. summary report of the excavations in 1900: I. the palace. The Annual of the British School at Athens, 6:3–70.

Facchetti, G. M. (1994). Linear A metrograms. Kadmos, 33(2):142–148.

Faucounau, J. (1977). Études chypro-minoennes. Syria, pages 209–249.

Ferrara, S. (2012). Cypro-Minoan Inscriptions: Volume 1: Analysis, volume 2. Oxford University Press.

Ferrara, S. (2013a). Cypro-Minoan Inscriptions: Volume 2: The Corpus, volume 2. Oxford University Press.

Ferrara, S. (2013b). Writing in Cypro-Minoan: one script, too many? In Syllabic Writing on Cyprus and its Context, pages 49–76. Cambridge University Press.

Fetaya, E., Lifshitz, Y., Aaron, E., and Gordin, S. (2020). Restoration of fragmentary Babylonian texts using recurrent neural networks. Proceedings of the National Academy of Sciences, 117(37):22743–22751.

Gecode Team (2005). Gecode: Generic constraint development environment. Available from `http://www.gecode.org`.

Godart, L., Poursat, J.-C., and Olivier, J.-P. (1996). Corpus hieroglyphicarum inscriptionum Cretae. École française d'Athènes.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Hirschfeld, N. (1999). Potmarks of the Late Bronze Age Eastern Mediterranean. Phd thesis, University of Texas at Austin.

Hirschfeld, N. (2012). Appendix iv: Potmarks. In Tombs of the Late Bronze Age in the Limassol Area, Cyprus (17th - 13th centuries BC), pages 289–299. Municipality of Limassol.

Hong, Y. (2013). On computing the distribution function for the Poisson binomial distribution. Computational Statistics & Data Analysis, 59:41–51.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2):251–257.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning, pages 448–456. PMLR.

Karageorghis, J. (1976). Une cruche chypriote inscrite du début du 5e siècle av. notre ère. Studi Ciprioti e rapporti di scavo, 2:59–68.

Karageorghis, V. and Karageorghis, J. V. (1956). Some inscribed Iron-Age vases from Cyprus. American Journal of Archaeology, 60(4):351–359.

Karnava, A. (2016). La scrittura "geroglifica" cretese. In Del Freo, M. and Perna, M., editors, Manuale di epigrafia micenea, pages 63–86. libreriauniversitaria.it.

Karnava, A. (2019). Old inscriptions, new readings: A god for the Rantidi sanctuary in South-West Cyprus. Cahiers du Centre d'Études Chypriotes, 49:19–36.

Knight, K., Nair, A., Rathod, N., and Yamada, K. (2006). Unsupervised analysis for decipherment problems. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 499–506.

Knight, K. and Yamada, K. (1999). A computational approach to deciphering unknown scripts. In Unsupervised Learning in Natural Language Processing.

Kober, A. E. (1945). Evidence of inflection in the" Chariot" tablets from Knossos. American Journal of Archaeology, 49(2):143–151.

Kober, A. E. (1948). The Minoan scripts: fact and theory. American Journal of Archaeology, 52(1):82–103.

Koch, G., Zemel, R., Salakhutdinov, R., et al. (2015). Siamese neural networks for one-shot image recognition. In ICML deep learning workshop, volume 2, page 0. Lille.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. AIChE journal, 37(2):233–243.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP).

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551.

Lewand, R. (2000). Cryptological mathematics, volume 16. Cambridge University Press.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. Advances in neural information processing systems, 30.

Luo, J., Cao, Y., and Barzilay, R. (2019). Neural decipherment via minimum-cost flow: From Ugaritic to Linear B. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3146–3155, Florence, Italy. Association for Computational Linguistics.

Luo, J., Hartmann, F., Santus, E., Barzilay, R., and Cao, Y. (2021). Deciphering undersegmented ancient scripts using phonetic prior. Transactions of the Association for Computational Linguistics, 9:69–81.

Masson, É. (1974). Cyprominoica: répertoires; documents de Ras Shamra; essais d'interprétation. P. B. Åstroms.

Masson, É. and Olivier, M. (1983). Appendix 4: Les objets inscrits de Palaepaphos-Skales. In Karageorghis, V. and Clerc, G., editors, Palaepaphos-Skales: An Iron Age Cemetery in Cyprus, Ausgrabungen in Alt-Paphos auf Cypern, pages 411–415. Universitätsverl.

Masson, O. (1983). Les inscriptions chypriotes syllabiques: recueil critique et commenté. (Étude chypriotes 1). Réimpression augmentée.

Masson, O. and Mitford, T. B. (1986). Les inscriptions syllabiques de Kouklia-Paphos.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Mitford, T., Masson, O., and Institut, D. A. (1983). The Syllabic Inscriptions of Rantidi-Paphos. Ausgrabungen in Alt-Paphos auf Cypern. Universitätsverlag Konstanz.

Mitford, T. B. (1958). Three inscriptions of Marium. Bulletin of the Institute of Classical Studies, 5:58–60.

Mitford, T. B. (1971). The inscriptions of Kourion, volume 83. American Philosophical Society.

Mitford, T. B. (1981). The Nymphaeum of Kafizin: the inscribed pottery, volume 2. De Gruyter.

Mitford, T. B. et al. (1961). Unpublished syllabic inscriptions of the Cyprus Museum. Minos, 7:15–48.

Montecchi, B. (2009). Le frazioni, gli errori di calcolo e le unità di misura nella documentazione in lineare A. Annali (Istituto Italiano di Numismatica), 55:29–52.

Montecchi, B. (2013). An updating note on Minoan fractions, measures, and weights. Annali (Istituto Italiano di Numismatica), 59:9–26.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). MiniZinc: Towards a standard CP modelling language. In International Conference on Principles and Practice of Constraint Programming, pages 529–543. Springer.

Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In European conference on computer vision, pages 69–84. Springer.

Olivier, J.-P. (2007). Édition holistique des textes chypro-minoens. Serra.

Olivier, J.-P. and Godart, L. (1975). Recueil des inscriptions en Linéaire A. École française d'Athènes.

Ostertag, C. and Beurton-Aimar, M. (2020). Matching ostraca fragments using a siamese neural network. Pattern Recognition Letters, 131:336–340.

Packard, D. W. (1971). Computer techniques in the study of the Minoan Linear Script A. KADMOS, 10.

Palaima, T. G. (1989). Cypro-Minoan scripts: Problems of historical context. In Problems in Decipherment, pages 121–187. Peeters.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau,

D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Perna, M. (1990). Segni di frazione semplici e composti. In Proceedings of the 6th bi-national critology conference.

Perna, M. (2015). Corpus of Cypriote syllabic inscriptions of the 1st millennium BC. Kyprios Character. History, Archaeology & Numismatics of Ancient Cyprus: kyprioscharacter. eie. gr/en/t/AS.

Pirrone, A., Aimar, M. B., and Journet, N. (2019). Papy-S-Net: A Siamese network to match papyrus fragments. In Proceedings of the 5th International Workshop on Historical Document Imaging and Processing, pages 78–83.

Pope, M. (1960). The cretulae and the Linear A accounting system. Annual of the British School at Athens, 55:200–210.

Popović, M., Dhali, M. A., and Schomaker, L. (2021). Artificial intelligence based writer identification generates new evidence for the unknown scribes of the Dead Sea Scrolls exemplified by the Great Isaiah scroll (1QIsaa). PloS one, 16(4):e0249769.

Rahmah, N. and Sitanggang, I. S. (2016). Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in Sumatra. In IOP conference series: earth and environmental science, volume 31, page 012012. IOP Publishing.

Ravanelli, R., Lastilla, L., and Ferrara, S. (2022). A high-resolution photogrammetric workflow based on focus stacking for the 3D modeling of small Aegean inscriptions. Journal of Cultural Heritage, 54:130–145.

Reimers, N. and Gurevych, I. (2017). Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Salgarella, E. and Castellan, S. (2020). SigLA: The Signs of Linear A. A Palæographical Database. In Haralambous, Y., editor, Proceedings of Grapholinguistics in the 21st Century, 2020, volume 5 of Grapholinguistics and Its Applications, pages 945–962, Brest. Fluxus Editions.

Schoep, I. (2002). The administration of neopalatial Crete: a critical assessment of the Linear A tablets and their role in the administrative process. Minos: Revista de filología egea, 17:1–230.

Schrijver, P. (2014). Fractions and food rations in linear A. Kadmos, 53(1-2):1–44.

Shaus, A., Gerber, Y., Faigenbaum-Golovin, S., Sober, B., Piasetzky, E., and Finkelstein, I. (2020). Forensic document examination and algorithmic handwriting analysis of Judahite biblical period inscriptions reveal significant literacy level. Plos one, 15(9):e0237962.

Skelton, C. (2008). Methods of using phylogenetic systematics to reconstruct the history of the Linear B script. Archaeometry, 50(1):158–176.

Smith, G. (1872). On the reading of the Cypriote inscriptions. Transactions of the Society of Biblical Archaeology, 1:129–144.

Snyder, B., Barzilay, R., and Knight, K. (2010). A statistical model for lost language decipherment. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1048–1057.

Srivatsan, N., Vega, J., Skelton, C., and Berg-Kirkpatrick, T. (2021). Neural representation learning for scribal hands of Linear B. In ICDAR 2021 Workshop on Computational Paleography, pages 325–338.

Steele, P. M. (2013). A linguistic history of ancient Cyprus: the non-Greek languages, and their relations with Greek, c. 1600–300 BC. Cambridge University Press.

Steele, P. M. and Meißner, T. (2017). From Linear B to Linear A: The problem of the backward projection of sound values, pages 93–110. Oxbow Books.

Stoltenberg, H. L. (1955). Die termilische Sprache Lykiens: Sprachbau und Wortschatz, Stele von Xanthos, Verwandtheit mit dem Etruskischen, jungminoische Lautschrift und alminoische Bruchzahlzeichen, volume 2. Gottschalksche Verlagsbuchhandlung.

Valério, M. (2013). Problems of Cypro-Minoan paleography: The case of sign shapes 08, 13 and 78. Kadmos, 52:111–134.

Valério, M. (2014). Seven uncollected Cypro-Minoan inscriptions. Kadmos, 53(1-2):111–127.

Valério, M. (2016). Investigating the Signs and Sounds of Cypro-Minoan. Phd thesis, Universitat de Barcelona.

Valério, M. and Davis, B. (2017). Cypro-Minoan in marking systems of the Eastern and Central Mediterranean: New methods of investigating old questions. Non-scribal Communication Media in the Bronze Age Aegean and Surrounding Areas, page 131.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. (2020). Scan: Learning to classify images without labels. In European conference on computer vision, pages 268–285. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Wang, W. Y. and Yang, D. (2015). That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Was, D. A. (1971). Numerical fractions in the Minoan Linear script a: the evaluation of the fraction signs. Kadmos, 10(1):35–51.

Wei, J., Bosma, M. P., Zhao, V., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2022). Finetuned language models are zero-shot learners. In Proceedings of the International Conference on Learning Representations.

Younger, J. (2000–2022). Linear A texts & inscriptions in phonetic transcription.

Younger, J. G. (2003). Calculating vessel volumes. Metron: Measuring the Aegean Bronze Age, ed. by Karen Polinger Foster, pages 491–492.

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., Li, H., and Liu, T. (2020). Incorporating BERT into neural machine translation. In International Conference on Learning Representations.

# Appendix A

# Validation Tables for the Paleographic vector

| Other sign | First 10 Tablet signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 001 �首 | **001 ꓮ** | 006 ꙮ | 005 ✦ | 008 ꝥ | 009 � | 040 Ꙩ | 075 ◻ | 004 ꓤ | 076 ꙭ | 092 ꙟ |
| Cosine Distance | **0.07** | 0.14 | 0.39 | 0.42 | 0.43 | 0.53 | 0.55 | 0.59 | 0.61 | 0.62 |
| 004 ꓤ | 005 ✦ | **004 ꓤ** | 040 Ꙩ | 006 ꙮ | 001 ꓮ | 044 ꝧ | 008 ꝥ | 029 ʌ | 028 ꙩ | 075 ◻ |
| Cosine Distance | 0.07 | **0.13** | 0.24 | 0.34 | 0.47 | 0.49 | 0.59 | 0.60 | 0.60 | 0.61 |
| 005 ✦ | **005 ✦** | 006 ꙮ | 040 Ꙩ | 004 ꓤ | 001 ꓮ | 009 ꓪ | 044 ꝧ | 102 ꙫ | 059 ꙇ | 107 Ꙫ |
| Cosine Distance | **0.05** | 0.24 | 0.28 | 0.34 | 0.41 | 0.49 | 0.50 | 0.54 | 0.55 | 0.55 |
| 006 ꙮ | **006 ꙮ** | 005 ✦ | 001 ꓮ | 107 Ꙫ | 009 ꓪ | 008 ꝥ | 040 Ꙩ | 104 Ꙭ | 102 ꙫ | 110 ꙭ |
| Cosine Distance | **0.14** | 0.26 | 0.33 | 0.41 | 0.45 | 0.48 | 0.49 | 0.50 | 0.58 | 0.59 |
| 008 ꝥ | **008 ꝥ** | 001 ꓮ | 006 ꙮ | 097 ꙩ | 096 Ꙩ | 095 ꙧ | 005 ✦ | 029 ʌ | 078 ꞃ | 107 Ꙫ |
| Cosine Distance | **0.09** | 0.28 | 0.34 | 0.40 | 0.48 | 0.48 | 0.50 | 0.58 | 0.58 | 0.58 |
| 009 ꓪ | **009 ꓪ** | 010 ꙅ | 006 ꙮ | 027 ꙙ | 005 ✦ | 001 ꓮ | 059 ꙇ | 040 Ꙩ | 074 ꙗ | 060 Ꙙ |
| Cosine Distance | **0.09** | 0.23 | 0.33 | 0.37 | 0.38 | 0.40 | 0.48 | 0.51 | 0.57 | 0.61 |
| 011 ꜄ | **011 ꜀** | 012 ꜃ | 005 ✦ | 001 ꓮ | 008 ꝥ | 040 Ꙩ | 009 ꓪ | 006 ꙮ | 028 ꙩ | 004 ꓤ |
| Cosine Distance | **0.33** | 0.35 | 0.42 | 0.44 | 0.47 | 0.48 | 0.49 | 0.49 | 0.50 | 0.54 |

Table A.1: Validation of CM *Other* → *Tablet* paleographic vector for DeepClusterv2. The correct targets are marked in bold and with a blue background.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 012 | **012** | 028 | 029 | 021 | 033 | 082 | 011 | 008 | 052 | 010 |
| Cosine Distance | **0.08** | 0.16 | 0.41 | 0.42 | 0.43 | 0.44 | 0.51 | 0.53 | 0.58 | 0.59 |
| 017 | 049 | 080 | **017** | 052 | 021 | 023 | 029 | 024 | 056 | 051 |
| Cosine Distance | 0.23 | 0.24 | **0.26** | 0.31 | 0.37 | 0.38 | 0.40 | 0.42 | 0.42 | 0.43 |
| 021 | **021** | 029 | 028 | 012 | 023 | 024 | 095 | 069 | 072 | 033 |
| Cosine Distance | **0.15** | 0.21 | 0.32 | 0.33 | 0.41 | 0.42 | 0.50 | 0.51 | 0.54 | 0.56 |
| 023 | 024 | **023** | 021 | 095 | 072 | 069 | 029 | 033 | 038 | 028 |
| Cosine Distance | 0.17 | **0.24** | 0.48 | 0.50 | 0.54 | 0.55 | 0.57 | 0.61 | 0.64 | 0.65 |
| 024 | **024** | 023 | 021 | 033 | 095 | 029 | 028 | 072 | 069 | 012 |
| Cosine Distance | **0.08** | 0.24 | 0.42 | 0.47 | 0.49 | 0.51 | 0.51 | 0.56 | 0.57 | 0.65 |
| 025 | **025** | 074 | 110 | 102 | 068 | 072 | 069 | 061 | 064 | 024 |
| Cosine Distance | **0.09** | 0.36 | 0.39 | 0.45 | 0.53 | 0.63 | 0.66 | 0.68 | 0.72 | 0.73 |
| 027 | 074 | **027** | 010 | 025 | 072 | 009 | 069 | 089 | 060 | 068 |
| Cosine Distance | 0.10 | **0.22** | 0.30 | 0.43 | 0.49 | 0.52 | 0.54 | 0.58 | 0.60 | 0.61 |
| 028 | **028** | 012 | 033 | 010 | 029 | 082 | 021 | 027 | 036 | 008 |
| Cosine Distance | **0.12** | 0.15 | 0.29 | 0.44 | 0.51 | 0.51 | 0.53 | 0.54 | 0.59 | 0.66 |
| 033 | 024 | 023 | 030 | **033** | 021 | 029 | 095 | 072 | 069 | 038 |
| Cosine Distance | 0.22 | 0.29 | 0.49 | **0.50** | 0.52 | 0.56 | 0.60 | 0.65 | 0.65 | 0.66 |
| 036 | **036** | 035 | 038 | 054 | 037 | 051 | 033 | 052 | 056 | 107 |
| Cosine Distance | **0.09** | 0.15 | 0.28 | 0.32 | 0.37 | 0.40 | 0.49 | 0.50 | 0.51 | 0.63 |
| 037 | **037** | 062 | 061 | 059 | 064 | 055 | 044 | 056 | 035 | 060 |
| Cosine Distance | **0.07** | 0.12 | 0.12 | 0.13 | 0.15 | 0.33 | 0.38 | 0.45 | 0.54 | 0.55 |
| 038 | **038** | 035 | 036 | 051 | 054 | 056 | 037 | 052 | 024 | 082 |
| Cosine Distance | **0.19** | 0.22 | 0.31 | 0.39 | 0.42 | 0.44 | 0.54 | 0.55 | 0.60 | 0.63 |
| 044 | **044** | 062 | 064 | 061 | 059 | 055 | 037 | 060 | 056 | 104 |
| Cosine Distance | **0.05** | 0.18 | 0.18 | 0.26 | 0.27 | 0.35 | 0.40 | 0.41 | 0.45 | 0.47 |
| 061 | 064 | 062 | 055 | 059 | **061** | 060 | 090 | 089 | 044 | 087 |
| Cosine Distance | 0.17 | 0.20 | 0.23 | 0.27 | **0.27** | 0.27 | 0.33 | 0.35 | 0.37 | 0.37 |

Table A.1: Validation of CM *Other* → *Tablet* paleographic vector for DeepClusterv2. The correct targets are marked in bold and with a blue background.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 069 | 070 | 071 | 076 | 068 | 073 | **069** | 075 | 072 | 097 | 095 |
| Cosine Distance | 0.25 | 0.27 | 0.27 | 0.29 | 0.32 | **0.36** | 0.39 | 0.40 | 0.45 | 0.47 |
| 070 | 073 | 071 | 076 | **070** | 075 | 092 | 097 | 068 | 100 | 069 |
| Cosine Distance | 0.28 | 0.35 | 0.36 | **0.40** | 0.44 | 0.48 | 0.49 | 0.52 | 0.53 | 0.56 |
| 075 | **075** | 087 | 089 | 090 | 070 | 092 | 068 | 076 | 069 | 078 |
| Cosine Distance | **0.06** | 0.34 | 0.38 | 0.39 | 0.40 | 0.40 | 0.41 | 0.46 | 0.47 | 0.48 |
| 082 | **082** | 012 | 028 | 052 | 049 | 017 | 051 | 021 | 004 | 029 |
| Cosine Distance | **0.19** | 0.22 | 0.27 | 0.37 | 0.43 | 0.45 | 0.46 | 0.46 | 0.49 | 0.53 |
| 087 | **087** | 089 | 090 | 091 | 092 | 060 | 019 | 075 | 027 | 010 |
| Cosine Distance | **0.16** | 0.19 | 0.23 | 0.23 | 0.29 | 0.30 | 0.40 | 0.40 | 0.50 | 0.50 |
| 096 | **096** | 095 | 097 | 104 | 068 | 107 | 076 | 082 | 110 | 102 |
| Cosine Distance | **0.08** | 0.16 | 0.22 | 0.41 | 0.49 | 0.49 | 0.52 | 0.56 | 0.56 | 0.56 |
| 097 | **097** | 078 | 096 | 079 | 008 | 047 | 095 | 029 | 104 | 004 |
| Cosine Distance | **0.19** | 0.37 | 0.44 | 0.45 | 0.48 | 0.50 | 0.51 | 0.57 | 0.61 | 0.61 |
| 102 | 104 | 107 | 110 | **102** | 047 | 079 | 044 | 097 | 078 | 025 |
| Cosine Distance | 0.27 | 0.39 | 0.46 | **0.57** | 0.60 | 0.60 | 0.60 | 0.64 | 0.67 | 0.69 |
| 104 | **104** | 052 | 080 | 049 | 004 | 017 | 054 | 044 | 051 | 107 |
| Cosine Distance | **0.45** | 0.55 | 0.58 | 0.59 | 0.60 | 0.61 | 0.61 | 0.63 | 0.65 | 0.66 |
| 107 | 054 | 104 | 052 | 051 | **107** | 056 | 049 | 044 | 036 | 080 |
| Cosine Distance | 0.30 | 0.33 | 0.43 | 0.45 | **0.47** | 0.53 | 0.56 | 0.58 | 0.59 | 0.60 |
| 110 | 104 | 107 | **110** | 044 | 080 | 054 | 056 | 102 | 079 | 047 |
| Cosine Distance | 0.30 | 0.45 | **0.52** | 0.55 | 0.64 | 0.64 | 0.64 | 0.64 | 0.66 | 0.66 |

Table A.1: Validation of CM *Other* → *Tablet* paleographic vector for DeepClusterv2. The correct targets are marked in bold and with a blue background.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 001 ⌶ | **001 ⌶** | 009 ⌁ | 006 ⚵ | 010 ⌳ | 027 ⚲ | 074 ⊖ | 068 ⊟ | 092 ⌇ | 005 ✦ | 011 ⌡ |
| Cosine Distance | **0.09** | 0.11 | 0.28 | 0.36 | 0.38 | 0.38 | 0.46 | 0.48 | 0.52 | 0.55 |
| 004 ⊢ | 005 ✦ | **004 ⊢** | 040 ✢ | 044 ⋓ | 068 ⊟ | 006 ⚵ | 009 ⌁ | 102 ⋔ | 059 ⚟ | 075 ⊡ |
| Cosine Distance | 0.13 | **0.16** | 0.29 | 0.37 | 0.37 | 0.44 | 0.46 | 0.47 | 0.50 | 0.50 |
| 005 ✦ | **005 ✦** | 040 ✢ | 009 ⌁ | 006 ⚵ | 004 ⊢ | 068 ⊟ | 001 ⌶ | 044 ⋓ | 102 ⋔ | 110 ⋓ |
| Cosine Distance | **0.10** | 0.25 | 0.31 | 0.35 | 0.40 | 0.43 | 0.44 | 0.53 | 0.55 | 0.55 |
| 006 ⚵ | **006 ⚵** | 009 ⌁ | 005 ✦ | 001 ⌶ | 068 ⊟ | 110 ⋓ | 010 ⌳ | 025 ⋒ | 102 ⋔ | 040 ✢ |
| Cosine Distance | **0.20** | 0.30 | 0.34 | 0.42 | 0.42 | 0.47 | 0.48 | 0.49 | 0.50 | 0.50 |
| 008 ⍑ | **008 ⍑** | 001 ⌶ | 009 ⌁ | 006 ⚵ | 074 ⊖ | 069 ⊖ | 027 ⚲ | 005 ✦ | 072 ⊟ | 033 ⋌ |
| Cosine Distance | **0.23** | 0.29 | 0.31 | 0.47 | 0.50 | 0.52 | 0.53 | 0.56 | 0.57 | 0.57 |
| 009 ⌁ | **009 ⌁** | 001 ⌶ | 010 ⌳ | 027 ⚲ | 059 ⚟ | 074 ⊖ | 005 ✦ | 060 ⌣ | 006 ⚵ | 012 ⌀ |
| Cosine Distance | **0.05** | 0.25 | 0.28 | 0.31 | 0.37 | 0.38 | 0.40 | 0.41 | 0.46 | 0.49 |
| 011 ⌡ | 009 ⌁ | **011 ⌡** | 012 ⌀ | 005 ✦ | 040 ✢ | 028 ⋔ | 010 ⌳ | 001 ⌶ | 027 ⚲ | 033 ⋌ |
| Cosine Distance | 0.27 | **0.29** | 0.36 | 0.36 | 0.44 | 0.44 | 0.46 | 0.50 | 0.54 | 0.55 |
| 012 ⌀ | **012 ⌀** | 028 ⋔ | 029 ⋏ | 021 ⋀ | 010 ⌳ | 027 ⚲ | 011 ⌡ | 033 ⋌ | 009 ⌁ | 091 ⋁ |
| Cosine Distance | **0.10** | 0.26 | 0.30 | 0.30 | 0.30 | 0.39 | 0.42 | 0.44 | 0.51 | 0.57 |
| 017 ⋋ | 080 ⋀⋀ | 049 ⋎⋏ | **017 ⋋⋏** | 052 ⋎⌡ | 056 ⋀⋀⋀ | 021 ⋀ | 051 ⋎⋌ | 054 ⋎⋏ | 023 ⋀⋀ | 047 ⋔ |
| Cosine Distance | 0.21 | 0.26 | **0.31** | 0.43 | 0.46 | 0.47 | 0.49 | 0.49 | 0.50 | 0.52 |
| 021 ⋀ | **021 ⋀** | 029 ⋏ | 012 ⌀ | 023 ⋀⋀ | 075 ⊡ | 028 ⋔ | 074 ⊖ | 024 ⋀⋌ | 010 ⌳ | 078 ⋔ |
| Cosine Distance | **0.11** | 0.13 | 0.35 | 0.38 | 0.40 | 0.47 | 0.50 | 0.52 | 0.57 | 0.58 |
| 023 ⋀⋀ | 024 ⋀⋌ | **023 ⋀⋀** | 021 ⋀ | 028 ⋔ | 072 ⊟ | 095 ⋔ | 074 ⊖ | 029 ⋏ | 033 ⋌ | 069 ⊖ |
| Cosine Distance | 0.07 | **0.20** | 0.48 | 0.53 | 0.53 | 0.55 | 0.55 | 0.55 | 0.56 | 0.58 |
| 024 ⋀⋌ | **024 ⋀⋌** | 023 ⋀⋀ | 028 ⋔ | 033 ⋌ | 021 ⋀ | 029 ⋏ | 095 ⋔ | 074 ⊖ | 072 ⊟ | 035 ⋎⌡⌡ |
| Cosine Distance | **0.04** | 0.24 | 0.44 | 0.45 | 0.47 | 0.52 | 0.52 | 0.55 | 0.55 | 0.59 |
| 025 ⋒ | **025 ⋒** | 074 ⊖ | 072 ⊟ | 069 ⊖ | 076 ⊕ | 095 ⋔ | 070 ⊟ | 027 ⚲ | 068 ⊟ | 023 ⋀⋀ |
| Cosine Distance | **0.22** | 0.29 | 0.44 | 0.46 | 0.49 | 0.51 | 0.52 | 0.54 | 0.58 | 0.60 |
| 027 ⋀⋌ | **027 ⚲** | 074 ⊖ | 010 ⌳ | 025 ⋒ | 072 ⊟ | 019 ⋎⋌ | 069 ⊖ | 012 ⌀ | 009 ⌁ | 076 ⊕ |
| Cosine Distance | **0.19** | 0.19 | 0.26 | 0.27 | 0.49 | 0.56 | 0.57 | 0.60 | 0.60 | 0.62 |

Table A.2: Validation of CM *Other* > *Tablet* paleographic vector for Sign2Vec$_d$

The correct targets are marked in bold and with a blue background.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 028 ⌇ | **028 ⌇** | 033 ⚐ | 012 ⌐ | 024 ⚐ | 027 ⚑ | 010 ⚏ | 036 ⚐ | 021 ⋀ | 029 ⋀ | 082 ⚐ |
| Cosine Distance | **0.09** | 0.29 | 0.33 | 0.37 | 0.43 | 0.44 | 0.54 | 0.57 | 0.59 | 0.59 |
| 033 ⚐ | 024 ⚐ | 023 ⋀ | **033 ⚐** | 021 ⋀ | 029 ⋀ | 030 ⚐ | 025 ⚐ | 074 ⚐ | 028 ⌇ | 027 ⚑ |
| Cosine Distance | 0.19 | 0.21 | **0.49** | 0.50 | 0.54 | 0.54 | 0.55 | 0.56 | 0.65 | 0.66 |
| 036 ⚐ | **036 ⚐** | 054 ⚐ | 035 ⚐ | 033 ⚐ | 051 ⚐ | 049 ⚐ | 037 ⚐ | 024 ⚐ | 028 ⌇ | 010 ⚏ |
| Cosine Distance | **0.11** | 0.15 | 0.15 | 0.43 | 0.45 | 0.47 | 0.47 | 0.49 | 0.49 | 0.54 |
| 037 ⚐ | **037 ⚐** | 062 ⚐ | 055 ⚐ | 061 ⚐ | 059 ⚐ | 107 ⚐ | 019 ⚐ | 091 ⚐ | 087 ⚐ | 010 ⚏ |
| Cosine Distance | **0.07** | 0.17 | 0.23 | 0.24 | 0.31 | 0.32 | 0.33 | 0.39 | 0.40 | 0.40 |
| 038 ⚐ | 054 ⚐ | 035 ⚐ | 049 ⚐ | 036 ⚐ | **038 ⚐** | 056 ⚐ | 051 ⚐ | 052 ⚐ | 055 ⚐ | 080 ⚐ |
| Cosine Distance | 0.26 | 0.30 | 0.31 | 0.34 | **0.38** | 0.43 | 0.45 | 0.48 | 0.48 | 0.50 |
| 044 ⚐ | **044 ⚐** | 104 ⚐ | 062 ⚐ | 107 ⚐ | 037 ⚐ | 059 ⚐ | 087 ⚐ | 061 ⚐ | 110 ⚐ | 092 ⚐ |
| Cosine Distance | **0.05** | 0.15 | 0.18 | 0.19 | 0.29 | 0.31 | 0.33 | 0.35 | 0.40 | 0.41 |
| 061 ⚐ | 062 ⚐ | 087 ⚐ | 092 ⚐ | 044 ⚐ | **061 ⚐** | 037 ⚐ | 055 ⚐ | 059 ⚐ | 091 ⚐ | 090 ⚐ |
| Cosine Distance | 0.17 | 0.21 | 0.24 | 0.26 | **0.29** | 0.30 | 0.30 | 0.31 | 0.32 | 0.34 |
| 069 ⚐ | 070 ⚐ | 076 ⚐ | **069 ⚐** | 072 ⚐ | 073 ⚐ | 075 ⚐ | 071 ⚐ | 096 ⚐ | 097 ⚐ | 095 ⚐ |
| Cosine Distance | 0.15 | 0.16 | **0.20** | 0.23 | 0.24 | 0.31 | 0.31 | 0.37 | 0.37 | 0.37 |
| 070 ⚐ | **070 ⚐** | 076 ⚐ | 073 ⚐ | 069 ⚐ | 075 ⚐ | 072 ⚐ | 071 ⚐ | 097 ⚐ | 095 ⚐ | 060 ⚐ |
| Cosine Distance | **0.18** | 0.25 | 0.26 | 0.28 | 0.31 | 0.33 | 0.40 | 0.42 | 0.50 | 0.52 |
| 075 ⚐ | **075 ⚐** | 076 ⚐ | 060 ⚐ | 070 ⚐ | 069 ⚐ | 061 ⚐ | 059 ⚐ | 089 ⚐ | 072 ⚐ | 064 ⚐ |
| Cosine Distance | **0.04** | 0.22 | 0.25 | 0.28 | 0.29 | 0.35 | 0.35 | 0.36 | 0.37 | 0.42 |
| 082 ⚐ | **082 ⚐** | 052 ⚐ | 049 ⚐ | 004 ⚐ | 028 ⌇ | 087 ⚐ | 021 ⋀ | 075 ⚐ | 029 ⋀ | 017 ⚐ |
| Cosine Distance | **0.20** | 0.32 | 0.40 | 0.41 | 0.44 | 0.49 | 0.50 | 0.51 | 0.51 | 0.52 |
| 087 ⚐ | **087 ⚐** | 092 ⚐ | 091 ⚐ | 090 ⚐ | 019 ⚐ | 089 ⚐ | 059 ⚐ | 075 ⚐ | 060 ⚐ | 021 ⋀ |
| Cosine Distance | **0.13** | 0.14 | 0.14 | 0.21 | 0.29 | 0.37 | 0.37 | 0.38 | 0.38 | 0.47 |
| 096 ⚐ | **096 ⚐** | 095 ⚐ | 076 ⚐ | 097 ⚐ | 104 ⚐ | 102 ⚐ | 070 ⚐ | 051 ⚐ | 068 ⚐ | 110 ⚐ |
| Cosine Distance | **0.10** | 0.25 | 0.43 | 0.45 | 0.48 | 0.49 | 0.53 | 0.54 | 0.55 | 0.55 |
| 097 ⚐ | **097 ⚐** | 095 ⚐ | 096 ⚐ | 013 ⚐ | 075 ⚐ | 004 ⚐ | 070 ⚐ | 076 ⚐ | 069 ⚐ | 017 ⚐ |
| Cosine Distance | **0.17** | 0.36 | 0.37 | 0.51 | 0.51 | 0.53 | 0.55 | 0.57 | 0.61 | 0.65 |

Table A.2: Validation of CM *Other* → *Tablet* paleographic vector for Sign2Vec$_d$

The correct targets are marked in bold and with a blue background.

| *Other* sign | First 10 *Tablet* signs ranked by distance | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 102 𐤂 | **102** | 110 | 104 | 044 | 107 | 047 | 068 | 079 | 004 | 038 |
| Cosine Distance | **0.21** | 0.21 | 0.23 | 0.44 | 0.46 | 0.51 | 0.53 | 0.53 | 0.53 | 0.56 |
| 104 | **104** | 044 | 110 | 102 | 004 | 052 | 107 | 056 | 047 | 079 |
| Cosine Distance | **0.20** | 0.36 | 0.36 | 0.36 | 0.47 | 0.47 | 0.49 | 0.51 | 0.55 | 0.57 |
| 107 | 104 | **107** | 044 | 056 | 110 | 052 | 051 | 102 | 038 | 037 |
| Cosine Distance | 0.14 | **0.27** | 0.34 | 0.44 | 0.44 | 0.44 | 0.45 | 0.49 | 0.51 | 0.54 |
| 110 | 104 | 102 | **110** | 044 | 107 | 056 | 047 | 004 | 079 | 038 |
| Cosine Distance | 0.16 | 0.27 | **0.27** | 0.36 | 0.41 | 0.44 | 0.50 | 0.51 | 0.53 | 0.53 |

Table A.2: Validation of CM *Other* → *Tablet* paleographic vector for Sign2Vec$_d$

The correct targets are marked in bold and with a blue background.