



SAPIENZA  
UNIVERSITÀ DI ROMA

# An AI Framework for Linear B Translation into Ancient Greek and English

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica  
Engineering in Computer Science

**Alessio Maiola**

ID number 1933744

Advisor

Prof. Aristidis Anagnostopoulos

Academic Year 2024/2025

---

**An AI Framework for Linear B Translation into Ancient Greek and English**  
Master Thesis. Sapienza University of Rome

© 2025 Alessio Maiola. All rights reserved

This thesis has been typeset by  $\text{\LaTeX}$  and the Sapthesis class.

Author's email: [maiola.1933744@studenti.uniroma1.it](mailto:maiola.1933744@studenti.uniroma1.it)

*Dedicated to  
Luigi Ricci*

## Acknowledgments

## Abstract

# Contents

<b>1</b>	<b>The Aegean Linear Scripts</b>	<b>1</b>
1.1	Historical Context . . . . .	1
1.2	The main sites . . . . .	2
1.3	Linguistic features . . . . .	4
1.4	The decipherment of Linear B . . . . .	7
1.4.1	The knowledge before the decipherment . . . . .	7
1.4.2	The decipherment by Ventris and Chadwick . . . . .	9
<b>2</b>	<b>The cognate matching task</b>	<b>13</b>
2.1	Identifying Cognates . . . . .	13
2.2	Datasets . . . . .	15
2.2.1	Prompt engineering . . . . .	16
2.2.2	Luo’s Dataset . . . . .	16
2.2.3	Tselentis’ Dataset . . . . .	17
2.2.4	Brute-Force Cognate Extraction . . . . .	19
2.3	Cognate Matching Model . . . . .	24
2.3.1	Cognate Matching Principles . . . . .	25
2.3.2	The Generative Framework . . . . .	25
2.3.3	NeuroDecipher Model . . . . .	26
2.3.4	Output Refinement . . . . .	29
2.3.5	The loss functions . . . . .	30
2.3.6	Training Procedure . . . . .	32
2.3.7	Results . . . . .	34
<b>3</b>	<b>Auxiliary Classifiers</b>	<b>38</b>
	<b>Bibliography</b>	<b>39</b>

## Chapter 1

# The Aegean Linear Scripts

Linear A and Linear B were writing systems used during the Bronze Age, primarily on the island of Crete, with some discoveries also made on the Greek mainland.

### 1.1 Historical Context

Around 2000 BCE, the already established Minoan civilization on the island of Crete began constructing large, complex architectural buildings commonly referred to as "palaces." These edifices served not only as administrative and economic centers, but also played important religious and ceremonial roles within Minoan society.

The founders of these palatial complexes were undoubtedly powerful landowners. Minoan society was highly organized and capable of mobilizing substantial manpower for major construction projects, such as leveling the hilltops at Knossos and Phaistos and erecting monumental palaces. [1]

Hence, this highly structured society began to feel the need for a form of administrative writing to record transactions, compile inventories, and manage other aspects of economic and bureaucratic activity.

The first form of writing developed by this society was a logographic script known as Minoan Hieroglyphics, or Cretan Hieroglyphics, attested between 2100 and 1700 BCE. The earliest and most archaic script was composed entirely of logographic symbols, which superficially resembled Egyptian hieroglyphs. It was later abandoned in favor of a linear script known as Linear A, employed between 1800 and 1450 BCE. The two systems initially coexisted for over a century, but in the following years, Linear A gradually replaced the former and became the sole writing system in use. [13]

Notably, the latest attestations of Cretan Hieroglyphs date to around 1700 BCE, when a catastrophe struck the island of Crete. All the palaces in the island's three main centers, Knossos, Phaistos, and Malia were destroyed. However, this did not lead to a cultural shift, as the palaces were promptly rebuilt, marking the passage from the Proto-palatial to the Neo-Palatial phase in Minoan history. [2]

This second phase of palace construction is the one that has survived to the present day, particularly at sites such as Knossos, Phaistos, Malia, and Zakros.

In 1450 BCE, a major catastrophe struck, probably caused by the eruption of the Thera volcano. It triggered devastating earthquakes and a tidal wave that

swept the north coast of Crete. As a result, the main centers of Minoan civilization, Phaistos, Aghia Triadha, Malia, the mansions of Tyliossos and Ammisos, as well as the eastern cities of Gournia and Zakros, were reduced to ruins. Knossos also suffered significant damage, often accompanied by widespread fires. [3]

In 1400 BCE, Crete began losing its central cultural role, and the focus shifted to mainland Greece, particularly the Peloponnese. The palace of Knossos was destroyed, while major fortified citadels (fortresses) were built in places like Mycenae and Tiryns. [4]

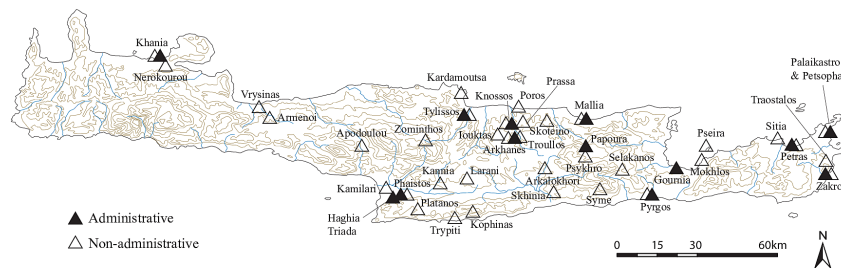
During this period, a new linear writing system emerged. Although visually similar to Linear A, it encoded a different language: an archaic form of Ancient Greek. Its name is Linear B, and it was used from 1400 to around 1100 BCE on Crete and the Greek mainland. The Mycenaean civilization, which flourished during this period, is characterized by its extensive use of Linear B for administrative purposes, particularly in palace economies. However, the destruction in Crete should not be interpreted as a Mycenaean military takeover, but rather as a transformative phase of socio-political and cultural adaptation. [14]

**Table 1.1.** Chronological framework of LA and LB [13]

Chronology	Crete			Mainland		
High Dating	Pottery Phase	Cultural Phase	Scripts	Pottery Phase	Cultural Phase	Scripts
1900–1800	MM II	Proto-Palatial	CH; LA	MH III	–	–
1800–1700	MM III		CH; LA	MH III		–
1700–1600	LM IA	Neo-Palatial	LA	LH I	Early Mycenaean	LA
1600–1450	LM IB		LA	LH IIA		?
1450–1400	LM II	Final-Palatial	LA?	LH IIB		?
1400–1375	LM IIIA1		LB	LH IIIA1	Late Mycenaean	LB
1375–1300	LM IIIA2	Post-Palatial	LB	LH IIIA2		LB
1300–1200	LM IIIB		LB	LH IIIB		LB
1200–1050	LM IIIC		–	LH IIIC		LB

## 1.2 The main sites

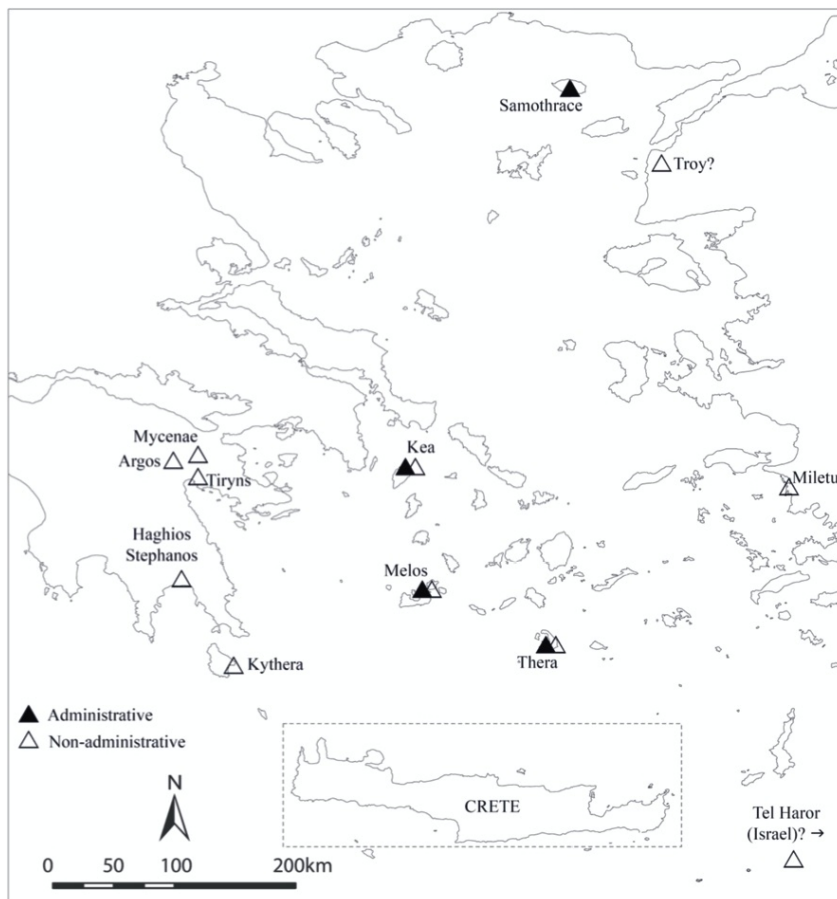
The main sites where Linear A documents have been found are located on the island of Crete. These include Knossos, Phaistos, Aghia Triada, Zakros, Khania, Tyliossos, and Malia.



**Figure 1.1.** Sites of Linear A fragments in Crete.<sup>1</sup>



Linear A was more widespread, covering completely Crete and the Aegean Islands and reaching the Greek mainland. The main attestations of Linear A on the Greek mainland are very limited and generally considered sporadic and isolated. At Mycenae, a few Linear A inscriptions have been found, likely as a result of commercial or cultural exchanges with Crete. Similarly, some fragmentary finds have been uncovered at Tiryns, probably also related to trade or contacts with Minoan Crete.



**Figure 1.2.** Sites of Linear A fragments in the Greek mainland.<sup>2</sup>

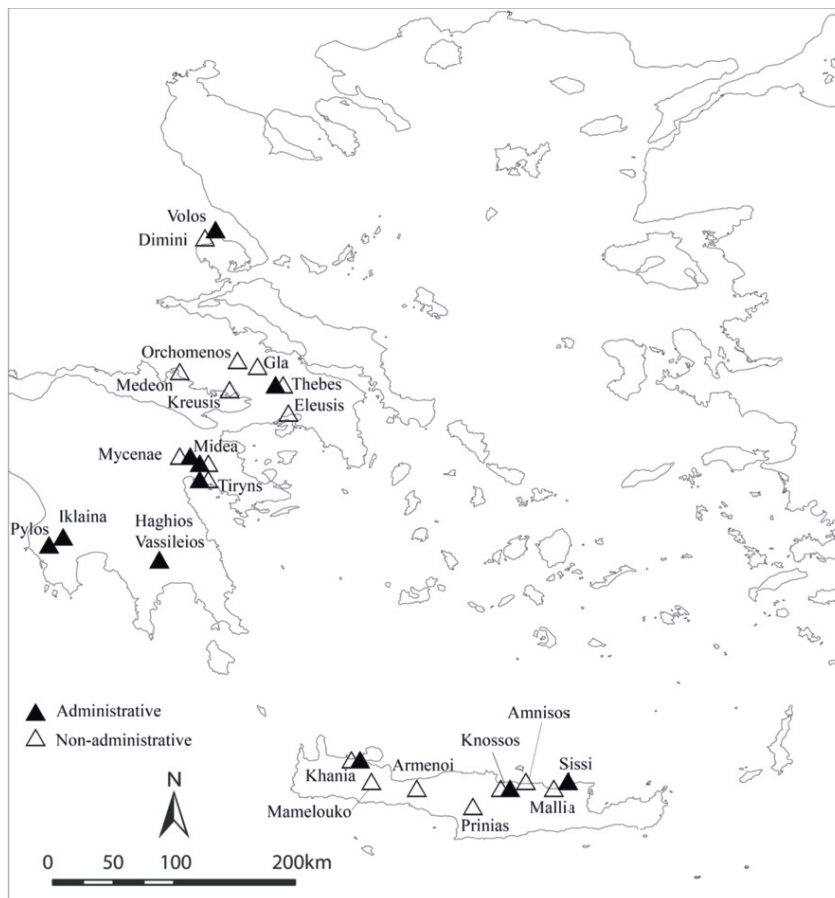
In contrast, Linear B is extensively attested on the Greek mainland, particularly in the Peloponnese, reflecting its administrative function during the Mycenaean period. Major sites where Linear B documents have been found include Mycenae, Tiryns, Pylos, Thebes, and Athens. Additionally, significant findings of Linear B tablets have been made in Crete, especially at Knossos and Khania.

The corpora of the two writing systems are relatively small, with Linear A consisting of approximately 1,400 documents, while Linear B comprises around 6,000 documents. Another notable difference is that Linear A was more widely used for non-administrative purposes, particularly in religious contexts, whereas the number of non-administrative Linear B documents is considerably more limited.

<sup>1</sup>Figure 1.1 prepared by Yannis Galanakis and Ester Salgarella.

<sup>2</sup>Figure 1.2 prepared by Yannis Galanakis and Ester Salgarella.

[13]



**Figure 1.3.** Sites of Linear B fragments in Crete and the Greek mainland.<sup>3</sup>

### 1.3 Linguistic features

The two writing systems are characterized by similar structural features, reflecting the connection between Linear A and Linear B and the derivation of the latter from the former.

The primary similarity between the two scripts lies in their syllabic structure, which constitutes a defining feature of both writing systems. Both Linear A and Linear B are syllabic scripts, meaning that each symbol represents a syllable rather than an individual letter or a full word. In addition to syllabic signs, both systems incorporate a set of logograms: symbols representing entire words or concepts.

This logographic component is particularly prominent in Linear A, where a significant number of signs are used to denote specific objects, actions, or concepts, often associated with administrative and religious contexts. By contrast, Linear B employs a more restricted set of logograms, reflecting its primary function in

<sup>3</sup>Figure 1.3 prepared by Yannis Galanakis and Ester Salgarella.

administrative record-keeping. Notably, the logograms used in Linear A were generally not inherited by Linear B, with a single exception: the logogram for "wool" (MA+RU), which is attested in both scripts. However, the principles governing the formation of logograms remained unchanged, as in both scripts they are formed by juxtaposing or combining two or more signs, either horizontally or vertically.

	Conf. a	Conf. b	Conf. c	Conf. d
Analytic juxtaposition				
Synthetic juxtaposition				

(a) Logogram construction criteria.



A 559 (MA+RU)

(b) LA logogram for wool.

**Figure 1.4.** Logograms in Linear A and Linear B.

100 VIR	85-108 SUS	115 P	125 CYPerus	132	154	160
102 MULier	108 <sup>f</sup> SUS <sup>f</sup>	116 N	125+KU CYP+KU	133 A+RE+PA	155 <sup>vas</sup>	161
104 CERVus	108 <sup>m</sup> SUS <sup>m</sup>	117 M	125+O CYP+O	135 ME+RI	155 <sup>vas</sup> +DI	162 TUNica
105 EQUus	108+KA SUS+KA	118 L	125+PA CYP+PA	140 AES	155 <sup>vas</sup> +NI	162+KI TUN+KI
105 <sup>f</sup> EQU <sup>f</sup>	108+SI SUS+SI	120 GRANum	125+QA CYP+QA	141 AURum	156 TU+RO <sub>2</sub>	162+QE TUN+QE
105 <sup>m</sup> EQU <sup>m</sup>	23-109 BOS	120+Q GRA+Q	127 KA+PO	142	157	162+RI TUN+RI
21-106 OVIS	109 <sup>f</sup> BOS <sup>f</sup>	120+PE GRA+PE	128 KA+NA+KO	144 CROCus	158	163 ARMa
106 <sup>f</sup> OVIS <sup>f</sup>	109 <sup>m</sup> BOS <sup>m</sup>	121 HORDeum	129 FAR	145 LANA	159 TELA	164
106 <sup>m</sup> OVIS <sup>m</sup>	109+SI BOS+SI	122 OLIVa	130 OLEum	146	159+KU TELA+KU	165
106+TA OVIS+TA	110 Z	122+A OLIV+A	130+A OLE+A	146+PE	159+PA TELA+PA	166
22-107 CAPer	111 V	122+TI OLIV+TI	130+PA OLE+PA	150	159+PO TELA+PO	166+WE
107 <sup>f</sup> CAP <sup>f</sup>	112 T	123 AROMa	130+SI OLE+SI	151 CORNu	159+PU TELA+PU	167
107 <sup>m</sup> CAP <sup>m</sup>	113 S	123+KO AROM+KO	130+WE OLE+WE	152	159+TE TELA+TE	167+PE
107+E CAP+E	114 Q	123+125 AROM+CYP	131 VINum	153	159+ZO TELA+ZO	168

**Figure 1.5.** Linear B logograms (symbols 100-168).

Figure 1.5 illustrates how logograms in Linear B can also incorporate syllabograms. In these cases, the syllabogram is referred to as an adjunct and typically serves to qualify or specify the meaning of the logogram. Moreover, the use of adjuncts is significantly more frequent in the Knossos corpus than on the Mainland, suggesting a possible continuity with Linear A, where isolated signs with semato-

graphic value appear more commonly. [15]

Furthermore, a substantial portion of the Linear A syllabary is shared with Linear B, with approximately 72% of Linear A signs being identical to those used in Linear B. This overlap also illustrates continuity in symbol creation and in the assignment of phonetic values between the two systems.

Syllabic signs					Special/unknown signs		
a	e	i	o	u	a <sub>2</sub> (ha)	a <sub>3</sub> (ai)	au
da	de	di	do	du	dwe	dwo	nwa
ja	je		jo		pu <sub>2</sub> (phu)	pte	ra <sub>2</sub> (rya)
ka	ke	ki	ko	ku	ra <sub>3</sub> (rai)	ro <sub>2</sub> (ryo)	ta <sub>2</sub> (tya)
ma	me	mi	mo	mu	tve	two	
					<b>Unknown / Doubtful values</b>  18  19  20  34  47  49  pa <sub>3</sub> ?  63  swi?  ju?  zu?  swa?  83  86  89		
na	ne	ni	no	nu			
pa	pe	pi	po	pu			
qa	qe	qi	qo				
ra	re	ri	ro	ru			
sa	se	si	so	su			
ta	te	ti	to	tu			
wa	we	wi	wo				
za	ze		zo				

**Figure 1.6.** All Linear B syllabograms with the associated phonetic values.

As observed in Figure 1.6, signs referring to the same vowel exhibit recurring patterns, a characteristic feature of syllabic scripts also evident in Linear A. One

of the most debated assumptions regarding the relationship between Linear A and Linear B is the principle of homomorphy and homophony. This principle posits that signs which are visually similar (homomorphy) in both scripts also share the same phonetic value (homophony), representing the same syllable. [13]

This observation has led to the widely accepted conclusion that Linear A encodes a language fundamentally different from Linear B, with the latter used to represent an archaic form of Ancient Greek. Consequently, although scholars are able to phonetically transcribe Linear A inscriptions, the language remains undeciphered and its meaning unknown.

## 1.4 The decipherment of Linear B

Ever since the discovery of the first Linear B tablets in 1900 by Sir Arthur Evans at Knossos, the script has been a subject of intense scholarly interest. Evans himself introduced the classification of Aegean scripts that is still used today. He also made the earliest attempts to decipher Linear B, though without success.

The breakthrough in understanding Linear B came after World War II, following major discoveries at the site of Pylos in 1939, which uncovered a large number of tablets and inscriptions. A key figure in the decipherment of Linear B was Michael Ventris, a British architect and amateur linguist. Ventris, in collaboration with philologist John Chadwick, succeeded in deciphering the script in 1952, demonstrating that it encoded an early form of Ancient Greek.

### 1.4.1 The knowledge before the decipherment

Before the decipherment of Linear B, scholars attempted to understand the script by studying it in isolation and by comparing it with known languages. One of the earliest hypotheses focused on the similarity between the Linear B syllabary and the Cypriot syllabary, a syllabic script used from about the eleventh to the fourth centuries BCE. The latter, which was also used to write an archaic form of Greek, shared a significant number of signs with Linear B.

However, this resemblance proved misleading in several respects. Although many signs appeared visually similar, as can be observed in Figure 1.7, their phonetic values often differed between the two systems. Moreover, the scripts treated grammatical suffixes differently. In Linear B, grammatical suffixes were frequently omitted, whereas this was not the case in the Cypriot syllabary, where the syllabogram "se" was regularly employed to indicate word endings.

Because "se" appeared regularly in the Cypriot syllabary but was rare in Linear B, early researchers concluded that Linear B could not represent the Greek language. In particular, Arthur Evans was convinced that the Minoan civilization was entirely distinct from the Mycenaean Greek world. This assumption, reinforced by Evans's authority and influence, contributed to delaying the recognition of the script's true linguistic nature. [7]

An example of this is the word "anthropos" (Ancient Greek: ἄνθρωπος), which appears in Linear B as "a-to-ro-qo" (𐀀𐀃𐀗𐀓), while in Cypriot it is spelled with the "se" suffix as follows: "a-to-ro-po-se" (𐀀𐀃𐀗𐀓𐀕𐀖, written from right to left).

Linear B	Cypriot	Value in Cypriot
𐀀	𐀀	ta
𐀁	𐀁	lo
𐀂	𐀂	to
𐀃	𐀃	se
𐀄	𐀄	pa
𐀅	𐀅	na
𐀆	𐀆	ti

Figure 1.7. Similarity between Linear B and Cypriot syllabary.

Evans had already established that most documents were administrative records. However, any attempt to decipher the script was hindered by unreliable underlying assumptions, while many aspects of the language remained unknown.

The most significant contribution came from Dr. Alice Kober, an American linguist. She was the first to approach the script methodically in order to uncover the nature of the underlying language. She sought to answer fundamental questions, such as whether the language was inflected or whether specific forms were used to indicate gender and number.

Through her careful analysis, she was able to identify grammatical patterns, including distinctions between masculine and feminine nouns, as well as examples of inflection. These results were achieved through a systematic study of repeated sign groups and their contexts, which she meticulously documented in her notebooks. Her work challenged Evans’s assumptions about the non-Greek nature of the language and paved the way for Ventris’ later decipherment. The outcome of her research was a set of "triplets", groups of three related words differing only in their endings, which provided critical evidence of an inflected language structure and were later referred to as "Kober’s triplets". Some examples of these triplets are shown in Figure 1.8. [8]

Case Types:	A	B	C	D	E
I:	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂
II:	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂
III:	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂	𐀀𐀁𐀂

Figure 1.8. Kobler’s triplets examples.

### 1.4.2 The decipherment by Ventris and Chadwick

The initial phase of the decipherment of Linear B involved the identification of numerical signs and easily recognizable logograms. This preliminary work was largely accomplished by Sir Arthur Evans, who successfully identified the most frequently occurring logograms and reconstructed the basic features of the numerical system. Notably, the Linear B script lacked a symbol for zero, but included fractional signs and was based on a decimal structure.

Building on these foundations, scholars began to assign tentative phonetic values to individual syllabograms through contextual analysis. By examining recurring patterns and the placement of signs within administrative texts, it became possible to propose the phonetic values of certain symbols. For instance, words such as "total" and "sons," which regularly appeared in similar tabular contexts, offered valuable clues for these early phonetic assignments.

However, the construction of a comprehensive and reliable grid of syllabograms was not feasible until the publication of the Pylos tablets in 1951. This newly available corpus significantly expanded the body of evidence, enabling more systematic linguistic analysis.

Michael Ventris began his work on the decipherment in 1950, initially by circulating a questionnaire among scholars to gather views on the possible nature of the language encoded in Linear B and its potential relationship to Linear A or the Cypriot syllabary.

Realizing that scholarly opinion was divided, Ventris adopted a combinatorial and structural approach. He explored positional patterns of signs within words, aiming to deduce their possible function or phonetic value. By analyzing the frequency and distribution of certain symbols, particularly those likely to represent vowels, he was able to identify a subset of syllabograms corresponding to pure vowels.

A key breakthrough came from the identification of inflectional endings, many of which were made possible by the foundational work of Alice Kober. Kober had demonstrated, through rigorous tabulation of sign groups, that the language encoded by Linear B exhibited an inflectional structure. Ventris incorporated these findings into his research and began constructing a syllabic grid, updating it continually in light of new phonological rules and morphological patterns.

For example, he correctly identified the syllabogram  $\text{𐀱}$  as representing the syllable *si*, based on its frequent use in noun declensions and verb conjugations. Its function mirrored that of the Ancient Greek suffix  $\sigma\iota$ , which appears in both oblique noun forms and verbal endings.

Additionally, place names proved particularly helpful, especially when accompanied by adjectival derivatives. These offered valuable comparisons with known Greek toponyms and morphological structures.

As Ventris refined his hypotheses and incorporated increasingly sophisticated linguistic deductions, especially regarding inflectional patterns and suffixes, he was able to assign phonetic values to a majority of the signs. Through this methodical approach, he successfully constructed a nearly complete syllabary grid. Most of his assignments proved to be correct, with only minor exceptions that were later adjusted through collaborative efforts with John Chadwick and subsequent scholarly review. [9]

**LINEAR B SYLLABIC GRID**

THIRD STATE : REVIEW OF PYLOS EVIDENCE

FIGURE 11  
WORK NOTE 17  
20 FEB 1952

SMALL SIGNS INDICATE UNCERTAIN POSITION, CIRCLED SIGNS HAVE NO OBVIOUS EQUIVALENT IN LINEAR SCRIPT A.

POSSIBLE VALUES		VOWELS					VOWEL UNCERTAIN
		-i ? -e ?	-o ? -a ?	-u ? -i ?	-e ? -i ?		
CONSONANTS		v 1	v 2	v 3	v 4	v 5	
PURE VOWEL ?	—	𐀀				𐀁	
j-?	c 1			𐀂		𐀃	
g-? v-? θ-? c-?	c 2	𐀄	𐀅	𐀆	𐀇	𐀈	
z-? p-?	c 3	𐀉		𐀊		𐀋	𐀌
ḡ-?	c 4	𐀍	𐀎	𐀏		𐀐	
t-?	c 5		𐀑			𐀒	
t-?	c 6	𐀓	𐀔	𐀕			𐀖
θ-? r-?	c 7	𐀗	𐀘	𐀙		𐀚	
n-?	c 8	𐀛	𐀜	𐀝		𐀞	
t-?	c 9	𐀟	𐀠	𐀡		𐀢	
h/x-? θ-?	c 10		𐀣	𐀤		𐀥	𐀦
r-? l-?	c 11	𐀧		𐀨		𐀩	𐀪
t-?	c 12	𐀫	𐀬	𐀭		𐀮	𐀯
v-? f-?	c 13	𐀰		𐀱		𐀲	
c-?	c 14			𐀳			
m-?	c 15		𐀴	𐀵		𐀶	𐀷
OTHER CONSONANTS		𐀸		𐀹			

Figure 1.9. Ventris' syllabary grid.

Together, Ventris and Chadwick continued to investigate how the script encoded the phonology of the Mycenaean Greek language, recognizing that Linear



B imposed structural constraints on how sounds were represented. The syllabary, like many early writing systems, lacked a one-to-one correspondence with spoken Greek. To better understand these adaptations, they analyzed the principles governing the script's orthography, namely how Greek words were transcribed within the limitations of the Linear B system.

The following rules summarize the most significant phonological and orthographic conventions that Ventris and Chadwick identified in the course of their research:

1. The script distinguishes five vowels: *a*, *e*, *i*, *o*, *u*. Vowel length, however, is not indicated.
2. The second component of diphthongs ending in *-u*, such as *au*, *eu*, *ou*, is represented explicitly.
3. In diphthongs ending in *-i* (e.g., *ai*, *ei*, *oi*, *ui*), the second component is generally omitted, except:
  - when it occurs before another vowel, in which case it is represented as *y*;
  - in the initial syllable *ai*, where the full diphthong is retained.
4. Glides occurring between a front vowel and a following vowel are indicated:
  - the glide after *i* is written as *j*;
  - the glide after *u* is written as *w*.

These sounds are typically omitted in Greek alphabetic spelling.

5. The script represents twelve consonants:
  - **j**: used only to represent diphthongal *i* or as a glide (see point 3);
  - **w**: corresponding to the archaic Greek digamma (*Ϝ*), pronounced as in English *w*;
  - **d**, **m**, **n**, **s**: approximately as in later Greek and English;
  - **r**: corresponding to Greek *l* and *r*;
  - **z**: corresponding to Greek *z*; its exact phonetic value in Mycenaean Greek remains uncertain;
  - **p**, **t**, **k**: representing both plain and aspirated stops, as the script does not distinguish aspiration;
  - **q**: representing a series of labio-velar stops (*kw*, *gw*, *khw*), which later disappeared from Greek but were preserved in Latin (e.g., *quis*, *unguem*).
6. The script does not represent aspiration; thus, aspirated consonants (*ph*, *th*, *kh*) are not distinguished from their unaspirated counterparts.
7. The consonants *l*, *m*, *n*, *r*, *s* are omitted when they occur:
  - at the end of a word;

- before another consonant.

For example, *po-me* = *poimen* (ποιμήν) "shepherd"; *ka-ko* = *khalkos* (χαλκός) "bronze"; *pa-te* = *pater* (πατήρ) "father".

- Initial *s*- is generally omitted before another consonant.
- In consonant clusters involving a stop + *w*, both consonants are represented, with an intervening vowel that is either taken from the following syllable or supplied as a default. However, *r* preceding *w* is typically omitted.
- Stop consonants (*d*, *k*, *p*, *q*, *t*) occurring before another consonant are typically written with the vowel of the following syllable (less often the preceding syllable). For example:

- *ku-ru-so* = *khrusos* (χρυσός) "gold";
- *A-mi-ni-so* = *Amnisos* (Ἀμνισός).

Special orthographic solutions are used to represent final consonant clusters, as in:

- *wa-na-ka* = *wanax* (φάναξ) "king".

These conventions reveal the extent to which the Linear B script adapted to the phonology of Mycenaean Greek, despite being constrained by a syllabic system originally designed for a different language. [10]

The decipherment of Linear B was finally confirmed one year later, when in 1953 a new tablet found in Pylos could be translated only by using Ventris' grid. This independent verification of the decipherment on material not previously available to Ventris provided undeniable evidence of his success. [11]

The work of Ventris and Chadwick, by combining structural analysis with comparative linguistics, thus not only unlocked the meaning of Linear B but also illuminated the phonological landscape of the earliest attested form of the Greek language.

## Chapter 2

# The cognate matching task

As explained in Section 1.4, the decipherment of Linear B required a collective effort that involved several scholars and extended over multiple decades.

The main difficulties in any decipherment process concerning ancient languages arise from two factors: the absence of parallel texts that could guide the interpretation, and the extreme scarcity of surviving documents. Even for domain experts, decipherment therefore demands encyclopedic linguistic and historical knowledge, combined with an enormous amount of manual work that is often prohibitive in terms of time and resources. Moreover, the challenges encountered during the decipherment of one language are rarely reusable for others, since much of the required work and insights are closely tied to the specific features of that language and cannot be easily generalized [12]. For example, the rules mentioned at the end of Section 1.4.2 are highly specific to the context of Linear B and its unique characteristics, and do not even apply to closely related scripts, such as Cypriot or possibly Linear A (for which no certainty exists, as the language remains undeciphered).

For these reasons, the introduction of computational methods and machine learning techniques has the potential to provide crucial assistance. Nevertheless, the central obstacle remains the limited amount of training data. This scarcity of examples makes it essential to design models that are able to learn effectively from small datasets and to generalize from minimal evidence.

In this respect, it is important to note that not all machine learning architectures are equally suitable for such low-resource conditions. Transformers, for example, are a class of neural networks that have achieved remarkable results in modern natural language processing tasks, such as machine translation and text generation. However, their success is strongly dependent on access to very large training corpora. In the absence of such data, as is the case for ancient scripts, the application of transformers becomes impractical. This explains why alternative architectures, which are better suited to scenarios where training data is scarce, are currently preferred in the computational study of ancient languages.

## 2.1 Identifying Cognates

Cognates are words in different languages that share a common etymological origin. Identifying cognates is a crucial task in historical linguistics, as it allows researchers

to trace the evolution of languages and to understand their relationships. In the case of Linear B, the identification of cognates provided valuable correspondences between the syllabic script and the phonetic representations of Ancient Greek, which proved instrumental in the decipherment process. Cognate matching was indeed a key aspect of Ventris' work, as he relied on clearly identifiable cognates to assign phonetic values to Linear B signs. However, the task of identifying cognates is not always straightforward, since it requires a deep understanding of the historical and linguistic context of the languages involved. In addition, several phonological transformations had taken place over time, including the differentiation of aspirated consonants and the loss of certain sounds once present in Linear B, such as the digamma (Ϝ) and the labio-velar consonant *q*.

Below are some examples of cognates identified between Linear B and Ancient Greek:

- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (a-e-ti-to) → ἀέθιστος, ἐθίζω  
This example shows how a Linear B sequence corresponds directly to a recognizable Greek adjective and verb.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (a-di-nwa-ta) → Ἀδινῶτας  
An instance where the Linear B syllabogram "nwa" reflects the presence of the digamma.
- $\text{𐀀} \text{𐀃} \text{𐀆}$  (e-ma-ha) → Ἑρμᾶς, Ἑρμαῖ, Ἑρμαίας  
Illustrates a clear lexical link to Greek terms related to Hermes.
- $\text{𐀀} \text{𐀃} \text{𐀆}$  (ko-no-so) → Κνωσός  
A straightforward toponym, the name of the main Minoan palace site.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (wo-no-qe-wa) → Φονοκέφας, Φονοκέφαν  
Preserves traces of the digamma and labio-velar consonants.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (wo-ro-ki-jo-ne-jo) → φοργιονεῖον, Ὀργιονεῖος, Ὀργίωνες  
A more complex example, relating to anthroponyms and associations.
- $\text{𐀀} \text{𐀃} \text{𐀆}$  (qi-si-pe-e) → ξίφεις, ξίφη, ξίφεα  
Reflects the Linear B representation of the Greek word for "sword."
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (re-u-ko-to-ro) → Λεῦκτρον, Λεῦκτροι  
A toponym, attested both in Linear B and later Greek sources.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (qo-u-qo-ta) → Βουβώτας  
Demonstrates the phonetic evolution of labio-velar sounds.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (qe-qi-no-me-na) → γεγινόμενα, γιγνόμενα  
A verbal form that shows continuity in Greek morphology.
- $\text{𐀀} \text{𐀃} \text{𐀆} \text{𐀇}$  (po-ro-wi-to-jo) → πλωριστοῖο  
Example of a genitive form with preservation of the digamma.
- $\text{𐀀} \text{𐀃} \text{𐀆}$  (po-ti-pi) → Πόρτις, Πόρτιφι  
Reflects a proper name with different Greek attestations.

- $\text{𐀀} \text{𐀃} \text{𐀆}$  (a-ri-qa) → Ἀρισβας  
Example of an anthroponym preserving older consonantal values.
- $\text{𐀓} \text{𐀬}$  (ko-no) → Σκοῖνος  
Shows the addition of a consonant in Ancient Greek.

## 2.2 Datasets

In this section, the process of gathering and preparing the datasets used in this study is described. It is worth noting that I made several choices in order to create a clean and consistent dataset. The main choices are the following:

- All diacritics, accents, breathings, and iota subscripts were removed from Ancient Greek forms, resulting in a simplified text. For example ἀέθιστος is represented as αεθιστος.
- All instances of uppercase letters were converted to lowercase in order to normalize the data. For instance Κνωσός is represented as κνωσος. Note also that Linear B does not distinguish uppercase and lowercase.
- All instances of punctuation were removed.
- Linear B words were represented using their Latinized transcription. Therefore, the dataset contains the word  $\text{𐀓} \text{𐀬}$  as ko-wo instead of the original Linear B characters.
- The instances of digamma were inserted in a suitable position also in the Greek form, despite their disappearance from Classical Greek. For example, the word κόπος is represented as κοπος in the dataset. In some cases, the version without digamma is also included, as in κυρος.
- The only additional symbol employed, besides the standard Greek alphabet, is "h", used to represent the aspiration conveyed by the syllabogram  $\text{𐀀}$  (ha). For instance,  $\text{𐀀} \text{𐀮} \text{𐀓} \text{𐀀}$  (a-pi-ha-ro) is rendered as αμφηαλος.

The sources of all the data included in the final version of the dataset are the following:

- **Luo's dataset** [12]: a collection of cognates between Linear B and Ancient Greek, compiled by Jiaming Luo. It contains 919 Linear B words together with their proposed Ancient Greek correspondences.
- **Chris Tselentis' \*Linear B Lexicon\*** [16]: a lexicon comprising 1338 Linear B entries with Ancient Greek equivalents. A substantial portion of these entries overlap with Luo's dataset.
- **Ventris and Chadwick's Vocabulary** [17]: a digitized compilation based on the original notes and lexical work of Michael Ventris and John Chadwick,

later supplemented with commentary by other scholars. The resource, available as a CSV file at <https://linear-b.kinezika.com/lexicon.html>, comprises 2747 unique words. It is organized as a vocabulary, offering definitions and interpretative remarks on the terms, and thus represents an extended digital derivative of their foundational work.

### 2.2.1 Prompt engineering

Some tasks were automated using prompt engineering techniques with Gemini 2.0 Flash and Gemini 2.5 Flash, which proved effective for text processing and data manipulation. Whenever these techniques are employed, I explicitly indicate their use and summarize the prompt instructions that were critical to the task. In general, the prompts followed these principles:

- **Clarity and specificity:** Clear, unambiguous instructions to reduce variance and align outputs with task requirements [6].
- **Iterative refinement:** Prompts were refined based on model outputs to improve quality across iterations.
- **Contextualization:** Task-relevant context (e.g., field definitions, examples) was included to guide disambiguation.
- **Structured reasoning:** Prompts encouraged stepwise reasoning for complex tasks (e.g., breaking a problem into sub-steps), leading to more coherent outputs [18].
- **Structured formatting:** Outputs were requested in explicit schemas (XML/JSON, bullet/numbered lists) to ensure machine-readable, post-processable results.
- **Salience cues (including UPPERCASE for emphasis):** Key requirements were emphasized to prioritize the most important aspects and reduce omission errors [5].

When necessary, I adjusted the model’s generation settings to favor determinism: the `temperature` was set low (e.g., 0.1–0.3) to reduce randomness and increase repeatability, and `top-k` was fixed at 1 (greedy selection).

### 2.2.2 Luo’s Dataset

Luo’s dataset is the one on which the creator of the NeuroDecipher model (introduced later) tested its performance. The model achieves excellent results on this dataset, reaching an accuracy of 84.7% of cognates correctly matched [12].

However, the dataset is not without limitations. The main issue I identified upon closer inspection is that some proposed Greek cognates are not attested in extant Ancient Greek sources, but rather tentative transliterations of Linear B forms or artificially modified versions of Greek correspondences. While this may artificially improve the cognate matching task, it does not reflect a realistic linguistic scenario

and makes the dataset unsuitable for use in an automatic translation pipeline. A few illustrative examples are given below:

- $\text{𐀓𐀗}$  (qo-o) is correctly associated with βους, as also confirmed by Tselentis' lexicon [16]. However, the dataset additionally lists κφος, which does not correspond to any attested Ancient Greek form.
- $\text{𐀓𐀗}$  (to-no) is linked to θοπος, likewise confirmed by Tselentis' lexicon [16], but the dataset also includes θορνος, an unattested form that results from an unjustified inversion of letters. This error extends to  $\text{𐀓𐀗𐀓𐀗𐀓𐀗}$  (to-ro-no-wo-ko), where the listed cognate is θορνοφοργος. I corrected this instead to θορνοφοργος and θορνοφεργος, both plausible formations obtained by combining θρόνος ("throne, chair") with the productive suffix derived from ἔργον ("work, deed"). The resulting compound restores the original sense of the term as "chair-maker" or "craftsman of thrones."
- In several cases, the dataset conflates distinct gendered forms by grouping them under a single entry. For example,  $\text{𐀓𐀗}$  (ne-wa), corresponding to νεα or νεα, feminine, and  $\text{𐀓𐀗}$  (ne-wo), corresponding to νεφος or νεος, masculine, were all grouped under  $\text{𐀓𐀗}$ , despite both variants being independently attested in the Linear B corpus.

The adjustments described above were applied to Luo's dataset in order to enhance its reliability and linguistic accuracy. This revised version was then used both to measure the performance of the NeuroDecipher model on a more realistic dataset and as the foundation for constructing the final comprehensive dataset, which also integrates additional material from Tselentis' Linear B Lexicon and from the digitized vocabulary of Ventris and Chadwick. The resulting revised dataset comprises 976 Linear B entries paired with their respective Ancient Greek correspondences.

### 2.2.3 Tselentis' Dataset

Tselentis' dataset represents a valuable resource for the study of Linear B, as it comprises a comprehensive lexicon of Linear B terms and their corresponding Ancient Greek forms. It serves as a crucial reference point for validating and enriching the cognate pairs identified in Luo's dataset.

The main drawback of Tselentis' lexicon is that it was only available as a PDF document, which made it necessary to manually transcribe the entries into a more usable format. After using an online OCR tool to extract its content into a CSV file, followed by targeted cleaning, a structured file containing all the fields in the lexicon was obtained.

Nevertheless, the data still required further processing to extract the Greek and Linear B forms in accordance with the normalization criteria outlined at the start of this section. Several parsing mistakes, along with inconsistencies in accents, diacritics, and formatting, had to be corrected to ensure accuracy and consistency. To streamline this process, Prompt Engineering techniques were employed with Gemini 2.0 Flash, guided by explicit processing directives.

These techniques enabled the automated correction of recurrent errors and inconsistencies, significantly accelerating data preparation. The processed dataset was then reviewed manually to ensure its quality and reliability before integration into the final comprehensive dataset.

For transparency and repeatability, I detail here the precise directives provided to Gemini for dataset processing.

### **PROCESSING RULES FOR LINEAR B TO GREEK COGNATES:**

0. **CRITICAL: DO NOT MAKE ANY MODIFICATION TO GREEK COGNATES OR LINEAR B SEQUENCES IF THE MODIFICATION IS NOT MENTIONED IN THE FOLLOWING RULES! DO NOT CHANGE THE INPUT IN ANY POSSIBLE WAY AND ONLY APPLY THE GIVEN MODIFICATION RULES! NO FANTASY JUST BLINDLY OBEY!**
1. **SPLITTING MULTIPLE WORDS:** When Linear B field contains multiple words separated by "/", create separate JSON objects for each word, matching with the corresponding Greek cognate in the same position.
2. **HANDLING PARENTHESES:** For Linear B words with parenthetical elements like "po-ni-ke-(j)a", create two separate entries (one with and one without the parenthetical element, like "po-ni-ke-ja" and "po-ni-ke-a").
  - 2.1. **HANDLING PARENTHESES FOR GREEK COGNATE:** if a word is presented with optional greek characters with parenthetical elements like "αιξμά(ν)ς", include both variants with and without the letter in parentheses.
  - 2.2. **HANDLING PARENTHESES FOR GREEK COGNATE:** if a word is presented within parentheses, include it regardless, like "Αιθαλεύσι(Αιθαλεύς)".
3. **MULTIPLE TRANSLATIONS:** If a Linear B word has multiple possible Greek cognates, include all of them as an array within the same JSON object.
4. **REMOVING DIACRITICS:** Remove all accents, breathing marks, and other diacritics from Greek cognates.
5. **HANDLING "ha" SIGN:** When "ha" appears in Linear B, ensure the corresponding Greek cognate includes "h".
6. **DIGAMMA CONVERSION:** Convert every instance of digamma "F" to lowercase "f" in Greek cognates.
7. **CRITICAL: ALLOWED CHARACTERS:** Use ONLY these characters in Greek cognates: ϖαβγδεζηθικλμνξοπρςστυφχψω. DO NOT USE ANY OTHER CHARACTER FOR ANY REASON!
8. **DISALLOWED CHARACTERS:** Drop cognates containing disallowed characters, but preserve valid cognates found within parentheses or other markers.



9. IN SOME VERY RARE AND PARTICULAR CASES some cognates may be considered as DUBIOUS, IF AND ONLY IF THEY CONTAIN A LIKELY WRONG TRANSLITERATION AND A CORRECT MATCH IS ALREADY PRESENT. Put them in the "dubious" field, another optional array field in the JSON object.
10. if white spaces are present between syllables separated by - in the linear b sequence, remove them.
11. DO NOT USE PARENTHESES IN THE LINEAR B SEQUENCES OR IN THE GREEK COGNATE.

Applying these directives reduced OCR noise and errors, preserving valid cognate pairs while preventing format drift that would hinder downstream parsing. These directives, together with a number of examples and some input and output definitions, allowed me to automate most of the manual work that the data needed in order to be ready to use.

#### 2.2.4 Brute-Force Cognate Extraction

To enlarge the dataset, I implemented and applied a brute-force, syllabogram-aware matcher over a large Greek lexicon (composed by the Iliad and Odyssey). Greek forms were first normalized (diacritics removed, lowercased) and then latinized via a direct character map aligned to Linear B conventions (e.g.,  $\xi \rightarrow ks$ ); special cases included the digamma ( $\varphi$  with later re-insertion of  $f/h$  during reconstruction) and the labio-velar  $q$  (permitted to align with  $\{q, p, k\}$ ).

**Matching logic.** Each Linear B form is tokenized into syllabograms and compared against every latinized Greek word by scanning both sequences left-to-right and greedily aligning syllabograms to characters. The matcher handles three syllabogram classes, with tailored rules:

- **V (length 1):** align the same vowel; mismatches advance the Greek pointer (counted as a skip).
- **CV (length 2):** align the initial consonant and then the vowel; special handling covers clusters such as  $k+s$  that straddle the next syllable (e.g.,  $ks$ ).
- **Specific triads (length 3):** a small set of syllabograms (e.g.,  $phu$ ) is matched as a fixed mini-pattern.

**State tracked during scanning.** The algorithm maintains (i) the total number of skipped Greek characters and the maximum consecutive skip streak (to avoid over-skipping), (ii) a flag for illegal syllable mappings, and (iii) a small relaxation allowing a single liquid glide (e.g., an isolated “r”).

**Acceptance gates.** After the pass, a candidate pair is accepted only if all of the following hold:

- (i) near-complete coverage of the Linear B syllables, i.e. the pointer must reach the last or penultimate syllable;
- (ii) the Greek pointer must lie within three characters of the word's end;
- (iii) the total number of skips must be fewer than four and no more than two consecutive characters may be skipped;
- (iv) an initial-character compatibility heuristic holds (to avoid wild misalignments);
- (v) no illegal syllable mapping was flagged;
- (vi) for short Linear B words, stricter thresholds are applied (fewer allowed skips and a smaller tail).

**Outputs and design choice.** For each accepted pair, the algorithm records a coverage score,

$$\text{coverage} = \frac{\text{matched syllables}}{\text{total syllables}},$$

and reconstructs the Greek surface form by re-inserting any f/h positions suppressed during normalization. The overarching design targets recall over precision: collect as many plausible pairs as possible, even at the cost of some spurious matches, to be filtered later.

[illegible]

```

28         max_skip_streak = skip_streak
29         j_chr += 1
30
31     elif len(lb_syllable) == 2:
32         cons = lb_syllable[0]
33
34         if cons == "k":
35             has_room = (j_chr + 2) < len(gr_chars)
36             tail = gr_chars[j_chr + 1 : j_chr + 3]
37             # checks double guttural
38             has_dg = has_room and ("".join(tail) ==
↪ lb_syllable)
39
40             if gr_char == "k" and not has_dg:
41                 skip_streak = 0
42
43             has_next_char = (j_chr + 1) < len(gr_chars)
44             # checks ks
45             next_is_s = has_next_char and (gr_chars[j_chr
↪ + 1] == "s")
46             has_next_syl = i_syl < (len(lb_syllables) -
↪ 1)
47
48             if next_is_s and has_next_syl: # matched ks
49                 next_syl = lb_syllables[i_syl + 1]
50                 if next_syl[0] == "s":
51                     i_syl += 2
52                     j_chr += 2
53                     vowel_ok = (
54                         j_chr < len(gr_chars)
55                         and next_syl[1] == gr_chars[j_chr
↪ ]
56                     )
57                     if vowel_ok:
58                         j_chr += 1
59                 else:
60                     i_syl += 1
61                     j_chr += 1
62                     vowel_ok = (
63                         j_chr < len(gr_chars)
64                         and lb_syllable[1] == gr_chars[
↪ j_chr]
65                     )
66                     if vowel_ok:
67                         j_chr += 1
68                 else:
69                     i_syl += 1
70                     j_chr += 1
71                     vowel_ok = (
72                         j_chr < len(gr_chars)
73                         and lb_syllable[1] == gr_chars[j_chr]
74                     )
75                     if vowel_ok:
76                         j_chr += 1
77                 else:
78                     j_chr += 1
79                     skip_count += 1

```

```

80         skip_streak += 1
81         if skip_streak > max_skip_streak:
82             max_skip_streak = skip_streak
83
84         if cons == "q": # labio-velar mapped to {q,p,k}
85             is_labio_map = gr_char in ("q", "p", "k")
86             if is_labio_map:
87                 skip_streak = 0
88                 i_syl += 1
89                 j_chr += 1
90                 vowel_ok = (
91                     j_chr < len(gr_chars)
92                     and lb_syllable[1] == gr_chars[j_chr]
93                 )
94                 if vowel_ok:
95                     j_chr += 1
96             else:
97                 j_chr += 1
98                 skip_count += 1
99                 skip_streak += 1
100                 if skip_streak > max_skip_streak:
101                     max_skip_streak = skip_streak
102             ...
103         if len(lb_syllable) == 3:
104             if lb_syllable == "phu":
105                 has_u = ((j_chr + 1) < len(gr_chars)) and (
106                 ↪ gr_chars[j_chr + 1] == "u")
107                 if gr_char == "p" and has_u:
108                     skip_streak = 0
109                     i_syl += 1
110                     j_chr += 2
111                 else:
112                     j_chr += 1
113                     skip_count += 1
114                     skip_streak += 1
115                     if skip_streak > max_skip_streak:
116                         max_skip_streak = skip_streak
117                 ...
118         # allow single liquid glide
119         if max_skip_streak == 2 and len(skipped_syllables) == 1:
120             if skipped_syllables[0][0] == "r":
121                 max_skip_streak = 0
122
123         # --- Acceptance constraints ---
124         start_ok = (
125             lb_word[0] == gr_word[0]
126             or (lb_word[0] == "p" and gr_chars[0] == "p")
127             ...
128         )
129         lb_aligned = i_syl >= (len(lb_syllables) - 1)
130         gr_near_end = j_chr >= (len(gr_word) - 3)
131         skips_ok = skip_count < 4
132         streak_ok = max_skip_streak <= 2
133         mapping_ok = not invalid_syllable
134

```

```

135     conds1 = (lb_aligned, gr_near_end, skips_ok, start_ok,
136             ↪ streak_ok, mapping_ok)
137
138     short_lb = len(lb_syllables) <= 3
139     near_end_tight = j_chr >= (len(gr_word) - 2)
140     short_ok = (skip_count <= 2) and near_end_tight and (
141             ↪ max_skip_streak < 2)
142     # conditions for shorter LB words
143     gate2 = (short_lb and short_ok) or (not short_lb)
144
145     if gate1 and gate2:
146         # Reconstruct normalized Greek, re-inserting 'f'/'h'
147         greek_norm_chars = list(greek_words[gr_word])
148         dropped = len(greek_words[gr_word]) < len(gr_chars)
149         if dropped:
150             fh = ("f", "h")
151             positions = [p for p, c in enumerate(gr_chars) if c
152                     ↪ in fh]
153             for off, pos in enumerate(positions):
154                 greek_norm_chars.insert(pos + off, gr_chars[pos])
155
156         coverage = i_syl / len(lb_syllables)
157         match_pair = ("".join(greek_norm_chars), coverage)
158         lin_b_words[lb_word]["cognates"].append(match_pair)
159
160     return lin_b_words

```

**Listing 2.1.** Brute-Force matching algorithm

The brute-force search has complexity  $O(N_{\text{LB}} \cdot N_{\text{GR}} \cdot L)$ , where  $N_{\text{LB}}$  is the number of Linear B forms,  $N_{\text{GR}}$  is the number of Greek forms, and  $L$  is the maximum length of words. For the translation dataset,  $N_{\text{LB}} = 4809$  and  $N_{\text{GR}} = 18646$ , with  $L \leq 19$ , as the longest word is  $\Upsilon \Xi \Delta \Lambda + \ddagger \S \oplus + \Upsilon$  (a-ni-ja-e-e-ro-pa-jo-qe-ro-sa).

**Outputs refinement.** Clearly, this brute-force approach is not perfect and, while it attempts to filter out very different words as much as possible, it inevitably returns more pairs than true cognates. This limitation is acceptable because a second filtering stage prunes and re-ranks candidates by means of prompt engineering. The filter is driven by a structured XML prompt and produces a strict JSON response. The main instructions given to Gemini 2.5 Flash can be summarized as follows:

- **Invocation of Luo’s principles:** the model must evaluate each proposal against the four cognate matching principles, which will be detailed in the next section, ensuring that they are applied consistently and rigorously. These principles are distributional similarity of matching characters, monotonic character mapping within cognates, structural sparsity of cognate mapping, and significant cognate overlap within related languages.
- **Character policy:** enforce the restricted output alphabet for Ancient Greek forms introduced in Section 2.2, disallowing accents, breathings, or subscript iota.

- **Phonological mapping tables:** require adherence to explicit Linear B  $\rightarrow$  Ancient Greek correspondence tables for consonants and vowels, supported by contextual notes (e.g. treatment of labiovelars, liquids, or clusters).
- **Independence from hints:** all provided "proposed cognates" are treated as non-binding; the model must search beyond them and consider the broader Ancient Greek corpus.
- **Plausibility assessment:** evaluate candidates against a rubric of evaluation criteria, aligned with the four principles, and reject items that fail any check.
- **Reasoning scaffold:** follow an explicit five-step chain-of-thought structure to ensure consistency in the decision path: syllabogram analysis  $\rightarrow$  monotonicity enforcement  $\rightarrow$  sparsity audit  $\rightarrow$  pattern comparison  $\rightarrow$  composite ranking.
- **Calibration of likelihoods:** apply a calibrated numerical scale with illustrative examples ranging from well-attested to speculative cases, so that likelihood scores are interpretable and comparable across words.
- **Automatic downweighting and hard caps:** apply four penalties, for example,  $-0.3$  for three or more non-trivial transformations or any reordering;  $-0.2$  for rarity, conflict with scholarship, or semantic stretch, and enforce global caps: likelihood  $< 0.7$  whenever the Linear B sequence contains unknown syllabograms such as \*19, and  $< 0.85$  for novel, unattested proposals even if other checks are passed.
- **Worked examples:** provide a wide set of input-output examples illustrating common patterns (such as  $q+s$ , digamma insertion, or suffixal transformations) to calibrate the model's application of rules and character policy.
- **Quality-control gate:** prefer fewer, higher-quality outputs over speculative additions and abstain when uncertain.
- **Instance metadata:** input word block: Linear B form, completeness level, optional definition from Chadwick and Ventris' vocabulary [17], any pre-proposed cognates with their brute-force matching scores, and entity type, to give context without constraining the final choice.

The complete XML prompt and the exact JSON schema are presented in the source code repository. After this automated filtering stage, the dataset was manually reviewed to ensure the quality and reliability of the final cognate pairs. Additionally, some cognates pairs were added from Ventris and Chadwick's vocabulary [17] when they were missing from the previous datasets.

## 2.3 Cognate Matching Model

In this section, the architecture and training procedure of the cognate matching model created by Jiaming Luo [12] are described. The model is based on a sequence-to-sequence (seq2seq) architecture with attention mechanisms, which has

been shown to be effective for various natural language processing tasks, including machine translation and text generation. The model is trained to take a Linear B word as input and generate its corresponding Ancient Greek cognate as output. As explained in Luo’s paper, the main challenge of the decipherment task is the lack of a strong supervision signal that guides standard machine translation algorithms. To address this challenge, he proposed an architecture that learns patterns of language transformation. Moreover, the model is designed to be language-agnostic, meaning that it can be applied to other decipherment tasks beyond Linear B and Ancient Greek. In order to achieve this, the model relies on a set of principles that capture the general characteristics of cognate relationships between languages.

### 2.3.1 Cognate Matching Principles

The four principles that guide the model’s architecture and training are the following:

1. **Distributional similarity of matching characters:** matching characters are expected to appear in similar places in corresponding cognates, therefore their local context should match as well.
2. **Monotonic character mapping within cognates:** cognates are expected to exhibit largely monotonic alignments, as character reorderings are rare.
3. **Structural sparsity of cognate mapping:** cognate matchings are expected to be sparse and near one-to-one between segments derived from the same proto-origin.
4. **Significant cognate overlap within related languages:** it is assumed that the derived vocabulary will provide sufficient coverage for recovering lost cognates.

### 2.3.2 The Generative Framework

The model architecture relies on a latent variable  $\mathcal{F} = \{f_{ij}\}$ , representing the word-level alignment between the words in the lost language  $\mathcal{X} = \{x_i\}$  and the known language  $\mathcal{Y} = \{y_j\}$ . The following joint probability is derived:

$$\mathbf{P}(\mathcal{X}, \mathcal{Y}) = \sum_{\mathcal{F} \in \mathbb{F}} \mathbf{P}(\mathcal{F}) \mathbf{P}(\mathcal{X} \mid \mathcal{F}) \mathbf{P}(\mathcal{Y} \mid \mathcal{F}, \mathcal{X}) \propto \sum_{\mathcal{F} \in \mathbb{F}} \mathbf{P}(\mathcal{Y} \mid \mathcal{X}, \mathcal{F}) = \sum_{\mathcal{F} \in \mathbb{F}} \prod_{y_j \in \mathcal{Y}} \mathbf{P}(y_j \mid \mathcal{X}, \mathcal{F}),$$

where a uniform prior is assumed for  $\mathbf{P}(\mathcal{F})$  and  $\mathbf{P}(\mathcal{X} \mid \mathcal{F})$ , and independence and identical distribution (i.i.d.) are assumed across  $y_j \in \mathcal{Y}$ . Here,  $\mathbb{F}$  denotes the set of all valid values of the latent variable  $\mathcal{F}$ .

The conditional probability  $\mathbf{P}(y_j \mid \mathcal{X}, \mathcal{F})$  is further decomposed as:

$$\mathbf{P}(y_j \mid \mathcal{X}, \mathcal{F}) = \sum_{x_i \in \mathcal{X}} f_{ij} \cdot \mathbf{P}_\theta(y_j \mid x_i),$$

where  $\mathbf{P}_\theta(y_j \mid x_i)$  is a neural seq2seq model with parameters  $\theta$  that generates  $y_j$  given  $x_i$ . In this framework, the latent variable  $\mathcal{F}$  captures the alignment between

the two languages, incorporating the global constraints described by Properties 3 and 4, while the neural model learns to generate cognates based on character-level constraints defined in Properties 1 and 2. However, directly optimizing the joint probability is infeasible, as it requires summing over all possible values of  $\mathcal{F}$ . To overcome this issue, Luo adopted an Expectation-Maximization (EM) iterative training algorithm: the neural model is trained in the M-step to maximize the likelihood of  $\prod_{y_j \in \mathcal{Y}} \mathbf{P}(y_j \mid \mathcal{X}, \mathcal{F})$  given fixed  $\mathcal{F}$ , while in the E-step  $\mathcal{F}$  is estimated via a minimum-cost flow problem defined over the trained neural network.

### 2.3.3 NeuroDecipher Model

The neural model  $\mathbf{P}_\theta(y_j \mid x_i)$ , named NeuroDecipher, is based on a standard seq2seq architecture. The three main components of the model are the encoder, the decoder, and the attention mechanism.

**Encoder.** The encoder consists of a stack of two embedding layers and a bidirectional LSTM. This design enforces the requirement that character embeddings of the two languages reside in the same space, the Universal Character Embedding Space. This is achieved by using a universal embedding matrix  $U \in \mathbb{R}^{\mathcal{U} \times E}$ , where  $\mathcal{U}$  is the dimensionality of the universal embedding space and  $E$  is the character embedding size used by the model. The second embedding layer is a lost language character weight matrix  $W_x \in \mathbb{R}^{V_b \times \mathcal{U}}$ , where  $V_b$  is the vocabulary size of the lost language. Therefore, the final embedding matrix for the lost language is  $E_x = W_x U$ . The bidirectional LSTM has hidden size  $H$  and produces a sequence of hidden and cell states  $(h_l, c_l)$  for  $l = 1, \dots, N$ , where  $N$  is the number of layers of the LSTM.

Thus, the encoder input is a batch of Linear B sequences with shape  $\mathbb{R}^{B \times L \times V_b}$ , where  $B$  is the batch size, and  $L$  is the (maximum) sequence length. The encoder output has shape  $\mathbb{R}^{B \times L \times 2H}$ , where  $H$  is the hidden size of the LSTM and  $2H$  accounts for the concatenation of forward and backward hidden states. Moreover, the hidden and cell states from all layers, each of shape  $\mathbb{R}^{2N \times B \times H}$  (for  $N$  LSTM layers in each direction), are averaged along the first dimension and used to initialize the hidden and cell states of each layer of the decoder.

One of my experiments involved initializing the lost language LSTM’s cell state with a projection of the FastText embeddings of the words in the batch concatenated. The motivation was that injecting additional contextual information might help the model capture relationships between Linear B and Ancient Greek more effectively. Empirically, this initialization stabilized training (more reliable convergence) but reduced final performance (see Section 2.3.7). A likely cause is representational mismatch: FastText captures word-level semantics, but the task needs character-level mappings; this biases the model toward irrelevant signals, which confuses learning and hurts generalization.

**Decoder and Global Attention.** Decoding proceeds by iterative next-character prediction from a fixed start token. The decoder is a multi-layer Custom LSTM. For the first LSTM layer at step  $t$ , the input is the concatenation of the context vector  $\tilde{h}_{t-1}$  and the embedding of the previously generated character  $e_{t-1}$ ; for higher layers, the input is the hidden state of the preceding layer. The last LSTM layer



produces the decoder state  $c_t \in \mathbb{R}^{B \times H}$ , which feeds a Global Attention module that integrates encoder and decoder's information.

Let the encoder outputs be  $o_s(1), \dots, o_s(L) \in \mathbb{R}^{B \times 2H}$ . A learnable matrix  $W_a \in \mathbb{R}^{2H \times H}$  linearly projects encoder states into the decoder space, and attention scores are computed by a scalar product with  $c_t$ :

$$s_t(k) = (W_a o_s(k)) \cdot c_t, \quad \alpha_t = \text{softmax}(s_t(1:L)) \in \mathbb{R}^{B \times L}.$$

The attention weights summarize encoder information in two complementary ways: a content summary in the hidden-state space,

$$c_s = \sum_{k=1}^L \alpha_t(k) o_s(k) \in \mathbb{R}^{B \times 2H},$$

and a content summary directly in the embedding space,

$$r = \sum_{k=1}^L \alpha_t(k) e_x(k) \in \mathbb{R}^{B \times E},$$

where  $e_x(k)$  is the lost-language input embedding of the  $k$ -th encoder character.

The context vector  $\tilde{h}_t$  fuses what the decoder is currently trying to produce ( $c_t$ ) with what the encoder deems most relevant ( $c_s$ ), and a residual anchor  $r$  in the input embedding space. Concretely, we concatenate  $c_s$  and  $c_t$ , map back to the model's embedding space with  $W_o \in \mathbb{R}^{3H \times E}$ , and add the residual:

$$\tilde{h}_t = W_o [c_s; c_t] + \rho r \in \mathbb{R}^{B \times E}.$$

The residual term  $r$  serves two purposes: (a) it stabilizes learning by providing a direct pathway from input embeddings to the decoder (improving gradient flow and helping early decoding steps), and (b) it regularizes the attention fusion by anchoring  $\tilde{h}_t$  to the same representation space as the inputs, which reduces drift and improves alignment consistency. The scalar  $\rho$  controls the contribution of this residual connection.

The vector  $\tilde{h}_t$  is then projected to the Ancient Greek vocabulary via

$$\text{logits}_t = \tilde{h}_t E_y^\top \in \mathbb{R}^{B \times V_g}, \quad E_y = W_y U \in \mathbb{R}^{V_g \times E},$$

followed by a softmax:  $p_t = \text{softmax}(\text{logits}_t)$ . The predicted character is  $\hat{y}_t = \arg \max p_t$ . For the next step, the "previous-token" embedding is taken as the expected embedding under  $p_t$ ,

$$e_t = p_t E_y \in \mathbb{R}^{B \times E},$$

and the process repeats for a number of steps corresponding to the length of the longest Ancient Greek word in the corpus. This design makes the context vector  $\tilde{h}_t$  the central carrier of both encoder-side evidence and decoder-side character-level information, while the residual pathway ensures robust, stable decoding grounded in the input embedding space. The overall model architecture is illustrated in Figure 2.1.

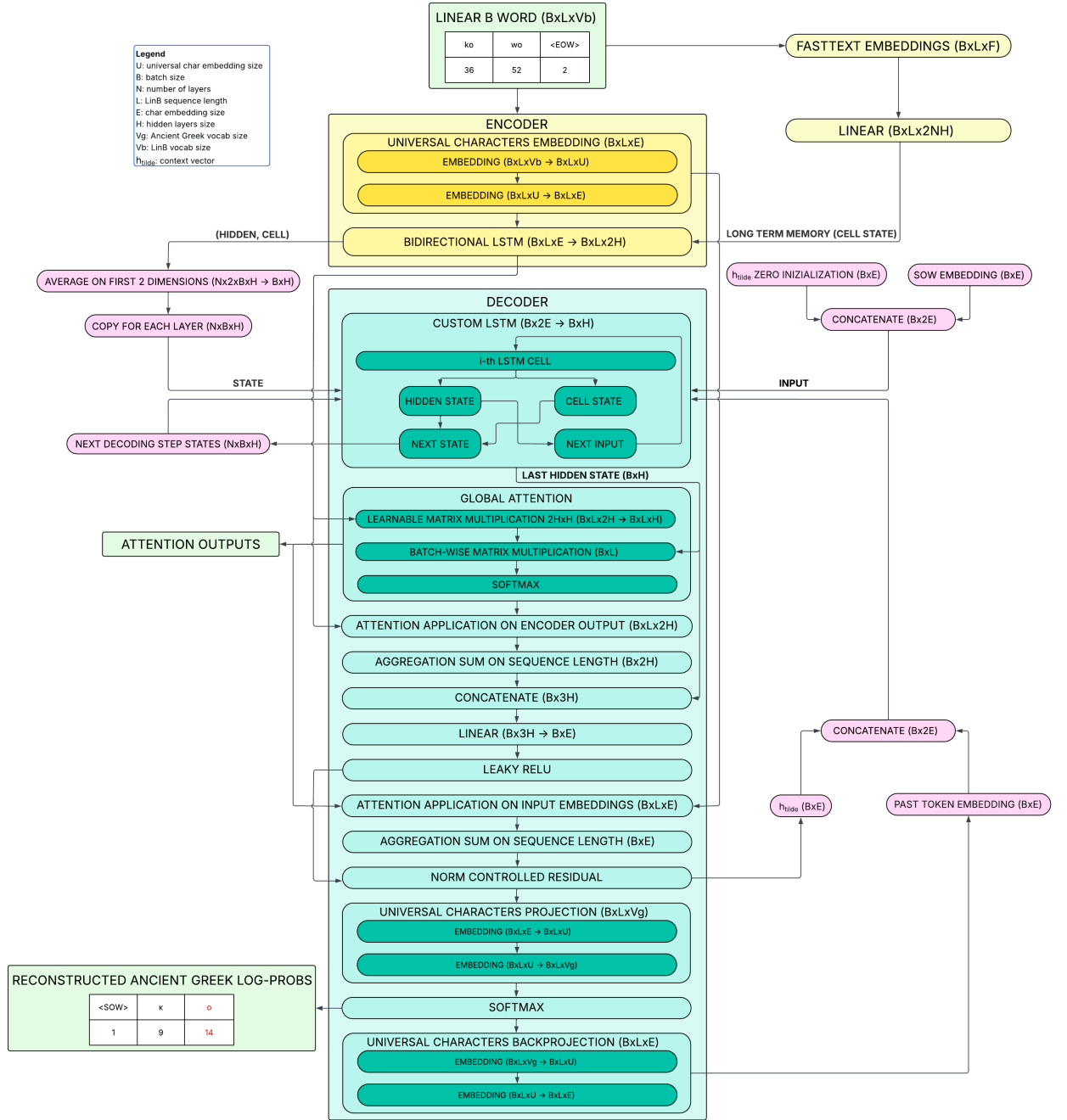


Figure 2.1. Graphical representation of the model architecture.

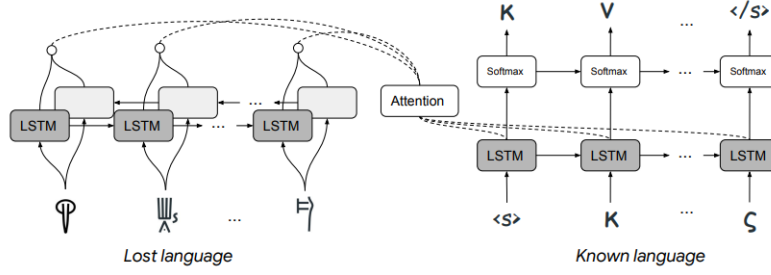


Figure 2.2. Overview of the model architecture.<sup>1</sup>

### 2.3.4 Output Refinement

The model supports three output modes. The first is a corpus-constrained maximum-likelihood selection (MLE) based on the decoder’s character distributions. Given the decoder outputs  $\{p_t\}$  for  $t = 1, \dots, T_{\max}$ , where  $p_t \in \mathbb{R}^{V_g}$  is a probability vector over the Greek vocabulary at step  $t$ , each candidate word  $y_j = (y_{j,1}, \dots, y_{j,T_w})$  in the lexicon is scored by the product of the probabilities of its characters:

$$\mathbf{P}_{\theta}(y_j \mid x_i) = \prod_{t=1}^{T_w} p_t[y_{j,t}],$$

Equivalently, if  $E(y_j) \in \{0, 1\}^{T_w \times V_g}$  is the one-hot matrix of  $y_j$  (row  $t$  is one-hot for  $y_{j,t}$ ), indicating with  $E_t(y_j)$  the  $t$ -th row, then

$$\log \mathbf{P}_{\theta}(y_j \mid x_i) = \sum_{t=1}^{T_w} E_t(y_j) \log p_t,$$

which is the form used in practice for numerical stability.

As explained in Section 2.3.2, the other two output modes are defined via a minimum-cost flow over a bipartite graph. Edges from source to lost-language nodes have unit capacity; edges from known-language nodes to sink have capacity  $C$  (a hyperparameter); the edges between languages have capacity 1. The two modes differ only in how the edge costs  $d_{ij}$  (from  $x_i$  to  $y_j$ ) are computed.

In the first flow mode, set  $d_{ij} = -\log \mathbf{P}_{\theta}(y_j \mid x_i)$ , i.e., reuse the corpus-constrained MLE score.

In the second flow mode, the character probabilities  $\{p_t\}$  are used to sample multiple candidate Ancient Greek words, which are then compared to each lexicon word  $y_j$  via normalized edit distance. For each lost language item  $x_i$  we estimate, for every lexicon word  $y_j$ , an expected normalized edit distance under the decoder’s output distributions:

1. **Sample  $S$  strings from the decoder.** Using the per-step character distributions  $\{p_t\}_{t=1}^{T_{\max}}$ , optionally sharpened by a temperature  $\alpha > 1$ , independently draw  $S$  strings and truncate at EOS:  $s_{i,k} = (s_{i,k,1}, \dots, s_{i,k,T_{s_{i,k}}})$ ,  $k = 1, \dots, S$ .

<sup>1</sup>Figure 2.2 prepared by Jiaming Luo.

2. **Score each sample by its (log-)likelihood.** With an EOS mask  $m_{i,k,t} \in \{0, 1\}$ , define

$$\log \mathbf{P}(s_{i,k} | x_i) = \sum_{t=1}^{T_{\max}} m_{i,k,t} \log p_t[s_{i,k,t}].$$

3. **Compute normalized edit distances to each candidate.** For every lexicon word  $y_j$  and each sample  $s_{i,k}$ ,

$$\delta(y_j, s_{i,k}) = \frac{\text{Lev}(y_j, s_{i,k})}{\min(|y_j|, |s_{i,k}|) + 1}.$$

Also the "original" candidate with zero distance is included:  $\delta(y_j, s_{i,0}) \equiv 0$  and score  $\log \mathbf{P}(s_{i,0} | x_i) \equiv \log \mathbf{P}_\theta(y_j | x_i)$ .

4. **Weight and average.** Convert the  $(1 + S)$  log-scores  $\{\log \mathbf{P}(s_{i,k} | x_i)\}_{k=0}^S$  to non-negative weights  $\{w_{i,k}\}_{k=0}^S$  proportional to their likelihoods and normalized to sum to 1. Set the edge cost to the expected normalized edit distance:

$$d_{ij} = \sum_{k=0}^S w_{i,k} \delta(y_j, s_{i,k}).$$

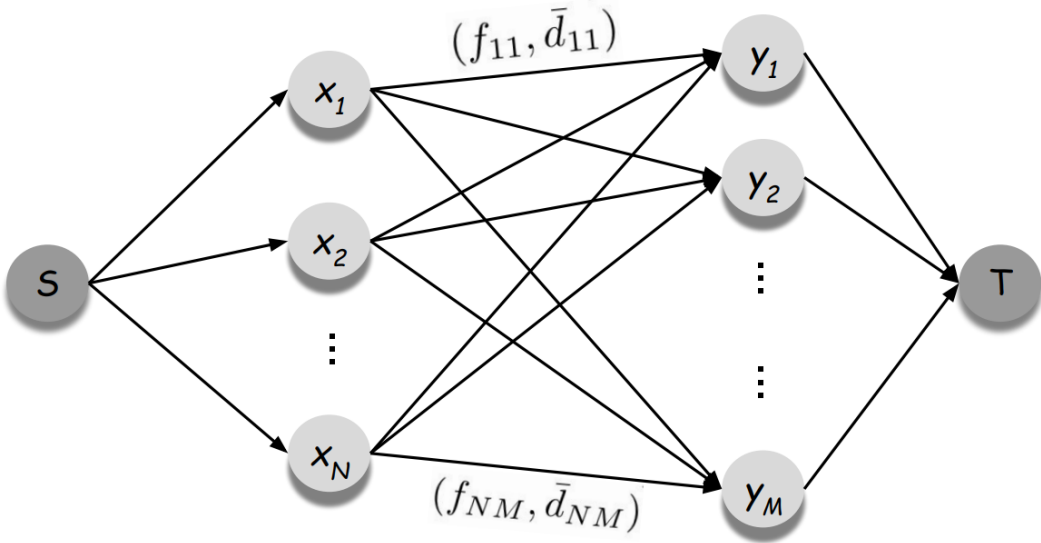


Figure 2.3. Overview of the min-cost flow architecture.<sup>2</sup>

### 2.3.5 The loss functions

The model is trained by minimizing the objective function introduced in Section 2.3.2. However, the flow solver returns sparse values, as the flow values for the

<sup>2</sup>Figure 2.3 prepared by Jiaming Luo.

edges are mostly zeros. This would likely discard most valid cognate pairs during training.

To address this issue, Luo proposed to use an exponential decay for the flow values in the loss function computation. These values are updated in each E-step and used in the M-step to compute the loss. Concretely, the flow values at iteration  $\tau$  are updated as follows:

$$f_{ij}^{(\tau)} = \gamma f_{ij}^{(\tau-1)} + (1 - \gamma) \tilde{f}_{ij}^{(\tau)} \quad \forall i, j,$$

where  $\tilde{f}_{ij}^{(\tau)}$  are the flow values returned by the flow solver at iteration  $\tau$ , and  $\gamma \in [0, 1]$  is a hyperparameter that controls the decay rate.

The loss function is then defined as:

$$\begin{aligned} \mathcal{L}_{\text{NLL}} &= - \sum_{y_j \in \mathcal{Y}} f_j^{(\tau)} \log \left( \sum_{x_i \in \mathcal{X}} \exp \left( \log f_{ij}^{(\tau)} + \log \mathbf{P}_\theta(y_j | x_i) \right) \right) = \\ &= - \sum_{y_j \in \mathcal{Y}} f_j^{(\tau)} \log \left( \sum_{x_i \in \mathcal{X}} f_{ij}^{(\tau)} \cdot \mathbf{P}_\theta(y_j | x_i) \right) = - \log \left( \prod_{y_j \in \mathcal{Y}} \left( \sum_{x_i \in \mathcal{X}} f_{ij}^{(\tau)} \cdot \mathbf{P}_\theta(y_j | x_i) \right)^{f_j^{(\tau)}} \right), \end{aligned}$$

where  $f_j^{(\tau)} = \sum_{x_i \in \mathcal{X}} f_{ij}^{(\tau)}$  is the aggregated flow into node  $y_j$  and acts as a weight for the loss of that word.

An additional loss component is introduced to encourage monotonic alignments. This component, called monotonic alignment regularization, penalizes unaligned positions of predicted characters. The loss acts directly on the attention weights  $\alpha_t$  produced by the Global Attention module.

Concretely, the attention weights are interpreted, for each word of the lost language  $x_i$ , as the alignment probability over the input sequence at each decoding step  $t$ . Therefore,  $\alpha_t^{(i)}(k) = \mathbf{P}(a_t^{(i)} = k | x_i)$ , where  $a_t^{(i)}$  is the alignment position at step  $t$  for word  $x_i$ . This probability is used to derive the expected alignment position for each character of the output sequence:

$$pos_t^{(i)} = \sum_{k=1}^L k \cdot \alpha_t^{(i)}(k) = \sum_{k=1}^L k \cdot \mathbf{P}(a_t^{(i)} = k | x_i).$$

The monotonic alignment regularization loss is then defined as:

$$\mathcal{L}_{\text{MAR}} = \sum_{x_i \in \mathcal{X}} \sum_{t=1}^{T_{\max}} (pos_t^{(i)} - pos_{t-2}^{(i)} - 1)^2,$$

to accommodate the fact that Linear B is a syllabic language and usually one Linear B sign corresponds to two Greek letters. An example of alignment between a Linear B word and its Ancient Greek cognate is presented in Figure 2.4.

		𐀀	𐀁	𐀂 (Linear B)
(Greek)	κ	✓		
	ν		✗	
	ω		✓	
	σ			✓
	ο			✓
	ς			✗

**Figure 2.4.** Example of alignment between a Linear B word and its Ancient Greek cognate. ✓ and ✗ indicate correct and incorrect alignment positions, respectively. The misalignment between 𐀁 (no) and ν corresponds to a deletion error, while the misalignment between 𐀂 (so) and ς corresponds to an insertion error.<sup>3</sup>

The final loss function is a weighted sum of the two components:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda \mathcal{L}_{\text{MAR}},$$

where  $\lambda$  is a hyperparameter that controls the trade-off between the two loss components.

### 2.3.6 Training Procedure

As anticipated in Section 2.3.2, training proceeds in an EM-style loop. We initialize the alignment flow matrix with a simple prior and then alternate between:

- **M-step:** fit the seq2seq parameters  $\theta^{(\tau)}$  by maximizing the corpus likelihood under the fixed flows  $f^{(\tau-1)}$ .
- **E-step:** recompute edge costs from  $\theta^{(\tau)}$ , solve a minimum-cost flow to obtain raw flows  $\tilde{f}^{(\tau)}$ , and update the stable flows via exponential smoothing:

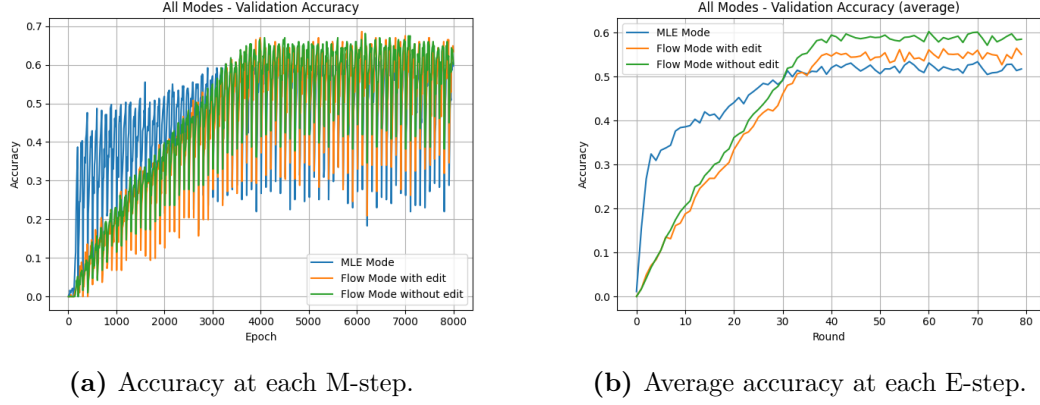
$$f_{ij}^{(\tau)} = \gamma f_{ij}^{(\tau-1)} + (1 - \gamma) \tilde{f}_{ij}^{(\tau)}.$$

To reduce overfitting to the previous iterate and to avoid poor local minima, the neural parameters are re-initialized at the start of each new M-step. This procedure is summarized in the pseudocode of Algorithm 1.

This behavior yields distinctive loss/accuracy trajectories for each split (Figure 2.5): curves change sharply at E-M boundaries because flows are recomputed and model parameters are re-initialized. A practical mitigation is to aggregate within each E-step (e.g., average across its M-steps) or, more conservatively, to report only the final M-step metrics.

<sup>3</sup>Figure 2.4 prepared by Jiaming Luo.

The full training procedure and loss computations are graphically illustrated in Figure 2.6.



**Figure 2.5.** Validation accuracy during training. (a) Raw per M-step accuracy. (b) Accuracy averaged within each E-step.

---

**Algorithm 1:** Iterative training with EM and minimum-cost flow <sup>4</sup>

---

**Input :**  $\mathcal{X}, \mathcal{Y}$  (vocabularies);  $T$  (number of iterations);  $N$  (number of cognate pairs to identify)

**Output:**  $f_{ij}^{(T)}$  (final soft alignments/flows)

- 1  $f_{ij}^{(0)} \leftarrow \frac{N}{|\mathcal{X}| |\mathcal{Y}|}$  // Initialize uniform flow
- 2 **for**  $\tau \leftarrow 1$  **to**  $T$  **do**
- 3    $\theta^{(\tau)} \leftarrow \text{MLE-TRAIN}(f_{ij}^{(\tau-1)})$
- 4    $d_{ij}^{(\tau)} \leftarrow \text{EDIT-DIST}(x_i, y_j; \theta^{(\tau)})$
- 5    $\tilde{f}_{ij}^{(\tau)} \leftarrow \text{MIN-COST-FLOW}(d_{ij}^{(\tau)})$
- 6    $f_{ij}^{(\tau)} \leftarrow \gamma f_{ij}^{(\tau-1)} + (1 - \gamma) \tilde{f}_{ij}^{(\tau)}$
- 7    $\text{RESET}(\theta^{(\tau)})$
- 8 **return**  $f_{ij}^{(T)}$
- 9 **Function**  $\text{MLE-TRAIN}(f_{ij}^{(\tau)})$
- 10    $\theta^{(\tau)} \leftarrow \arg \max_{\theta} \prod_{y_j \in \mathcal{Y}} \mathbf{P}_{\theta}(y_j | \mathcal{X}, \mathcal{F})$
- 11   **return**  $\theta^{(\tau)}$

---

<sup>4</sup>Algorithm 1 taken by Jiaming Luo.





and 10% for test. For both sets of experiments, I compared the model across all output modes, with and without initializing the LSTM cell states with FastText embeddings. The results are presented in the tables below.

LSTM Initialization	MLE	Flow without edit	Flow with edit
zero	0.830	0.831	0.781
custom	0.822	0.822	0.820

**Table 2.1.** Comparison of model performance on Luo’s original dataset.

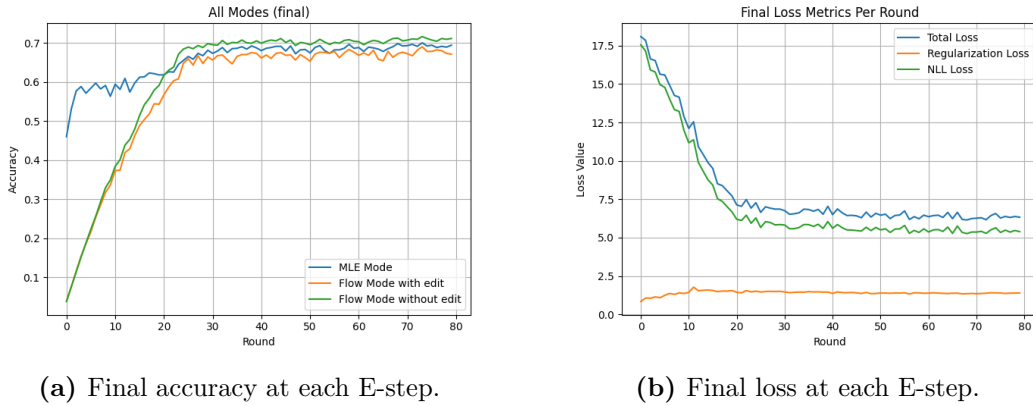
LSTM Initialization	MLE	Flow without edit	Flow with edit
zero	0.827	0.828	0.781
custom	0.841	0.841	0.840

**Table 2.2.** Comparison of model performance on Luo’s corrected dataset.

LSTM Initialization	MLE	Flow without edit	Flow with edit
zero	0.694	0.711	0.671
custom	0.662	0.662	0.672

**Table 2.3.** Comparison of model performance on our dataset.

It can be observed that, despite the stabilization effect of the custom LSTM initialization, when the model converges with a zero initialization it achieves better performance. Particularly, the model attains higher accuracy on our more challenging and extensive dataset, reaching 71.1% with the flow without edit mode. The final accuracy and loss at each E-step during training on our dataset with zero initialization of the cell states are presented in Figure 2.7.



**Figure 2.7.** Metrics for the zero-initialized model during training on our dataset.

(a) Accuracy. (b) Loss.

The same trends are observed in the experiments with train/validation/test splits. However, due to the reduced size of the training set and the removal of some cognate pairs from it, the overall performance is lower. While the zero-initialized

model remains strong, the custom-initialized variant degrades substantially. The results are presented in the tables below.

LSTM Init	Split	MLE	Flow w/o edit	Flow w/ edit
zero	Train	0.789	0.790	0.735
	Validation	0.663	0.674	0.696
	Test	0.761	0.761	0.728
custom	Train	0.411	0.423	0.435
	Validation	0.391	0.424	0.435
	Test	0.359	0.359	0.435

**Table 2.4.** Performance (Train/Validation/Test) on Luo’s original dataset.

LSTM Init	Split	MLE	Flow w/o edit	Flow w/ edit
zero	Train	0.782	0.785	0.732
	Validation	0.717	0.717	0.663
	Test	0.739	0.739	0.696
custom	Train	0.457	0.469	0.486
	Validation	0.446	0.467	0.511
	Test	0.370	0.380	0.511

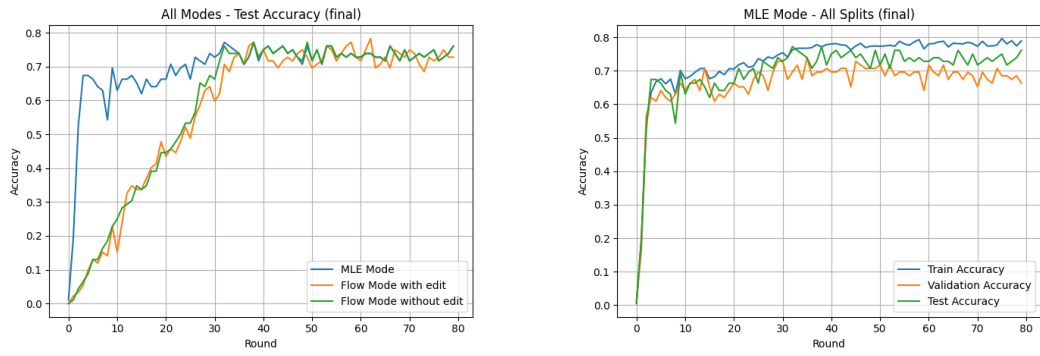
**Table 2.5.** Performance (Train/Validation/Test) on Luo’s corrected dataset.

LSTM Init	Split	MLE	Flow w/o edit	Flow w/ edit
zero	Train	0.645	0.658	0.639
	Validation	0.597	0.602	0.634
	Test	0.625	0.646	0.646
custom	Train	0.238	0.275	0.367
	Validation	0.262	0.262	0.377
	Test	0.224	0.234	0.370

**Table 2.6.** Performance (Train/Validation/Test) on our dataset.

This significant drop in the custom-initialized runs is consistent with a loss accuracy divergence caused by a representation mismatch. FastText injects word-level semantic priors into a character-mapping task. As a result, the model becomes confidently wrong on many instances: the negative log-likelihood can keep improving (higher confidence), while accuracy on held-out data declines (worse argmax decisions). In short, the initialization biases the decoder toward signals that are orthogonal to the target grapheme-level correspondences, leading to miscalibration and poorer generalization, due to representational mismatch.

The final accuracy of all splits, using MLE mode, and the final accuracy of the test split, using all output modes, of the zero-initialized model during training on our dataset are presented in Figure 2.8.



(a) Final accuracy at each E-step for the test set in all modes.

(b) Final accuracy at each E-step in MLE mode for all splits.

**Figure 2.8.** Metrics for the zero-initialized model during training on our dataset.

(a) Accuracy for the test set in all modes. (b) Accuracy in MLE mode for all splits.

## Chapter 3

# Auxiliary Classifiers

PALLE

# Bibliography

- [1] S. Alexiou. “Minoan Civilization.” In: trans. by C. Ridley. Heraklion, Crete: Spyros Alexiou Sons, 1969. Chap. 2, p. 23.
- [2] S. Alexiou. “Minoan Civilization.” In: trans. by C. Ridley. Heraklion, Crete: Spyros Alexiou Sons, 1969. Chap. 3, p. 34.
- [3] S. Alexiou. “Minoan Civilization.” In: trans. by C. Ridley. Heraklion, Crete: Spyros Alexiou Sons, 1969. Chap. 4, pp. 50–51.
- [4] S. Alexiou. “Minoan Civilization.” In: trans. by C. Ridley. Heraklion, Crete: Spyros Alexiou Sons, 1969. Chap. 5, p. 59.
- [5] Vivi Andersson et al. “UPPERCASE IS ALL YOU NEED.” In: *Proceedings of SIGBOVIK 2025*. Presented at SIGBOVIK 2025, Carnegie Mellon University, April 4, 2025. Association for Computational Heresy. Pittsburgh, PA, USA, Apr. 2025. URL: <https://sigbovik.org/2025/proceedings.pdf>.
- [6] Maximilian Beurer-Kellner, Jürgen Cito, and Zhendong Su. “LMQL: Prompting Is Programming.” In: *arXiv preprint arXiv:2212.06094* (2023). arXiv: 2212.06094 [cs.CL]. URL: <https://arxiv.org/abs/2212.06094> (visited on 08/31/2025).
- [7] J. Chadwick. “The Decipherment of Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 1958. Chap. 2, pp. 22–25.
- [8] J. Chadwick. “The Decipherment of Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 1958. Chap. 3, pp. 26–28, 33–35.
- [9] J. Chadwick. “The Decipherment of Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 1958. Chap. 4, pp. 40–66.
- [10] J. Chadwick. “The Decipherment of Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 1958. Chap. 5, pp. 75–76.
- [11] J. Chadwick. “The Decipherment of Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 1958. Chap. 6, pp. 81–84.
- [12] Jiaming Luo, Yuan Cao, and Regina Barzilay. “Neural Decipherment via Minimum-Cost Flow: From Ugaritic to Linear B.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3146–3155. DOI: 10.18653/v1/P19-1303. URL: <https://aclanthology.org/P19-1303/>.

- [13] E. Salgarella. “Aegean Linear Script(s): Rethinking the Relationship between Linear A and Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 2020. Chap. 1, pp. 1–7, 31–36, 39–40.
- [14] E. Salgarella. “Aegean Linear Script(s): Rethinking the Relationship between Linear A and Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 2020. Chap. 5, p. 378.
- [15] E. Salgarella. “Aegean Linear Script(s): Rethinking the Relationship between Linear A and Linear B.” In: Cambridge, United Kingdom: Cambridge University Press, 2020. Chap. 3, pp. 209–215.
- [16] Chris Tselentis. *Linear B Lexicon*. Athens, Greece, Apr. 9, 2011. URL: <https://archive.org/details/LinearBLexicon>.
- [17] Michael Ventris and John Chadwick. *Documents in Mycenaean Greek: Three Hundred Selected Tablets from Knossos, Pylos and Mycenae, with Commentary and Vocabulary*. 2nd ed. Cambridge: Cambridge University Press, 1973.
- [18] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.” In: *arXiv preprint arXiv:2201.11903* (2022). arXiv: 2201.11903 [cs.CL]. URL: <https://arxiv.org/abs/2201.11903> (visited on 08/31/2025).