



**Manchester  
Metropolitan  
University**

---

Paxton-Fear, Katie (2018) A Computational Decipherment of Linear B. In: Computer Applications and Quantitative Methods in Archaeology Conference (CAA 2018), 19 March 2018 - 23 March 2018, Tübingen, Germany.

---

**Downloaded from:** <https://e-space.mmu.ac.uk/627538/>

**Version:** Presentation

Please cite the published version

<https://e-space.mmu.ac.uk>



# Lessons of the Past, Tools of the Future

## A Computational Decipherment of Linear B

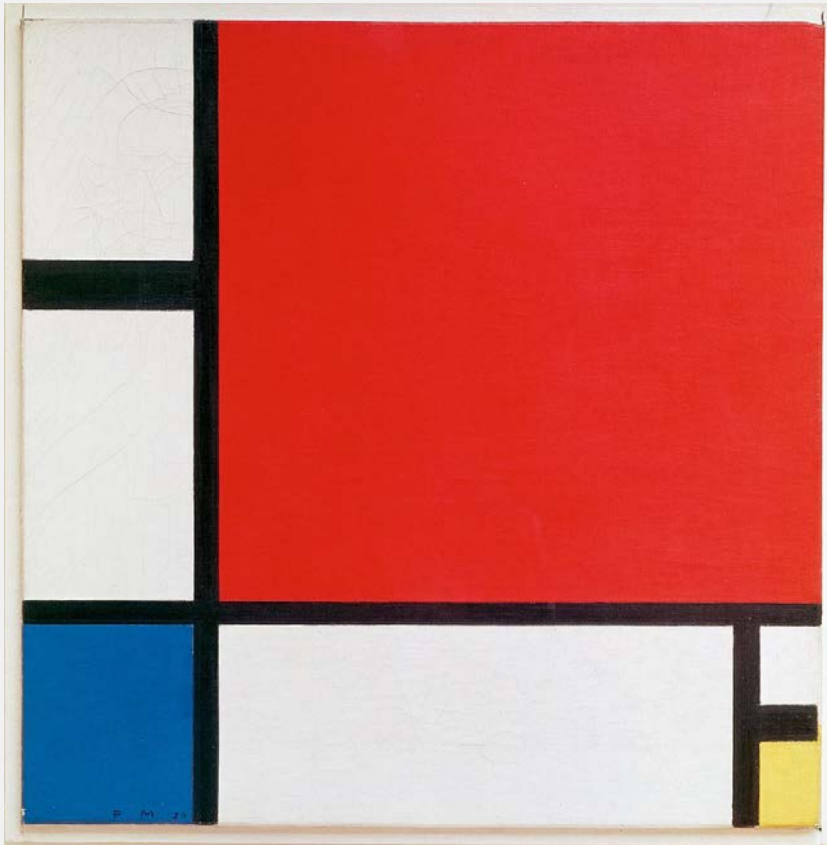
Katie Paxton-Fear

PhD student in Defence and Security

Cranfield University

[K.Paxton-Fear@cranfield.ac.uk](mailto:K.Paxton-Fear@cranfield.ac.uk)

# Creativity is Limitation



- Limitation: Follow the steps of the original decipherment
  - “Standing on the Shoulders of Giants”
- A different approach to a interdisciplinary project

# Background

- Linear B was found on Crete and at select places on the mainland
- It is a syllabic language
- The language was used administratively
- Related languages
  - Linear A, Cypro-Minoan, Cretan Hieroglyphs, Classical Cypriot



# A Recipe for Decipherment



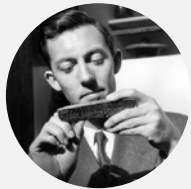
1) Correctly classify and transcribe tablets

Completed by Emmett L. Bennett Jr.



2) Find evidence of inflection

Completed by Alice Kober



3) Create a grid of characters

Completed by Michael Ventris



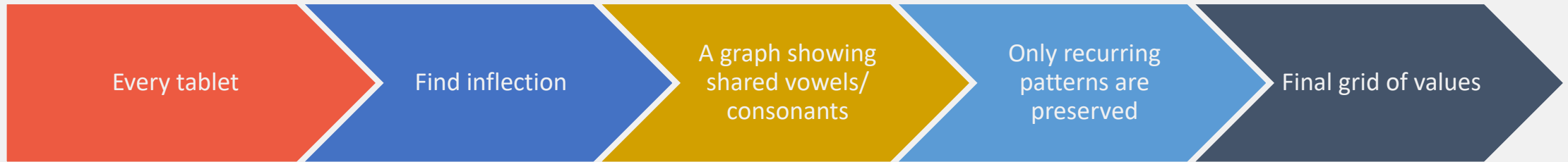
4) Begin assigning likely values to the grid

Completed by John Chadwick & Ventris

	A	E	I	O	U
VOWEL	𐀀	𐀁	𐀂	𐀃	𐀄
D	𐀅	𐀆	𐀇	𐀈	𐀉
J	𐀊	𐀋		𐀌	𐀍
K	𐀎	𐀏	𐀐	𐀑	𐀒
M	𐀓	𐀔	𐀕	𐀖	𐀗
N	𐀘	𐀙	𐀚	𐀛	𐀜
P	𐀝	𐀞	𐀟	𐀠	𐀡
Q	𐀢	𐀣	𐀤	𐀥	
R	𐀦	𐀧	𐀨	𐀩	𐀪
S	𐀫	𐀬	𐀭	𐀮	𐀯
T	𐀰	𐀱	𐀲	𐀳	𐀴
W	𐀵	𐀶	𐀷	𐀸	
Z	𐀹	𐀺		𐀻	



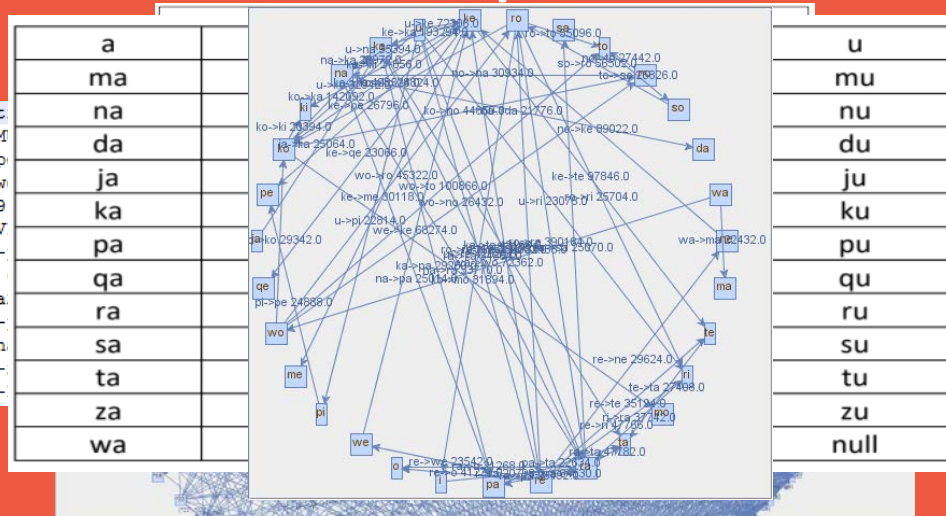
# System flow



## Output

```

identifier;location;series;inscript.
KN Ag 87;KN;Ag;]-wa , ;]-wa VIR 1 M
KN Ag 88 + 7033;KN;Ag;pe-re-ko , ;p
KN Ag 89;KN;Ag;to-ro-wo , ;]to-ro-w
KN Ag 90;KN;Ag;e-ri-*19 , ;e-ri-*19
KN Ag 91;KN;Ag;ke-re-u , ;ke-re-u V
KN Ag 1654;KN;Ag;qe-ri-jo , ;qe-ri-
KN Ai 338;KN;Ai;]-ja , ;.A ] ko-wa
KN Ai 632;KN;Ai;]-ta-ra2 , ;]-ta-ra
KN Ai 752 + 753;KN;Ai;re-ja , ;]re-
KN Ai 762;KN;Ai;e-ne-ra , ;]ra-ma-n
KN Ai 824;KN;Ai;do-e-ra , ;a-pi-qo-
KN Ak 611;KN;Ak;to-te-ja , (de)-di-
    
```



firmed with Chadwick & Ventris 1973  
 ] ko-wo , me-wi-jo (1)[ vacat [

# Finding Inflection: Original Work

- Kober originally found evidence that Linear B was inflected
- Kober's algorithm
  - Select **words which are followed by ideograms and numerals**
  - Find the same **word in different contexts**
  - Find **predictable patterns** where the word endings change

	Type A		Type B		
Case I	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢
Case II	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢	𐀮 𐀶 𐀭 𐀢
Case III	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢

	Type C	Type D	Type E
Case I	𐀮 𐀶 𐀢 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢
Case II	𐀮 𐀶 𐀢 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢
Case III	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢	𐀮 𐀶 𐀢

# Finding Inflection: Computational Approach

- A visual representation
- Loop through each word
  - Loop through each word
    - If the word is exactly the same – ignore
    - Else
      - Loop through the characters in word 1
        - Does this character match the character in word 2
          - Increase the similarity
        - Else – stop, these words are dissimilar

walk [w,a,l,k]  
talking [t,a,l,k,i,n,g]  
walking → [w,a,l,k,i,n,g]  
wanting [w,a,n,t,i,n,g]  
walked [w,a,l,k,e,d]

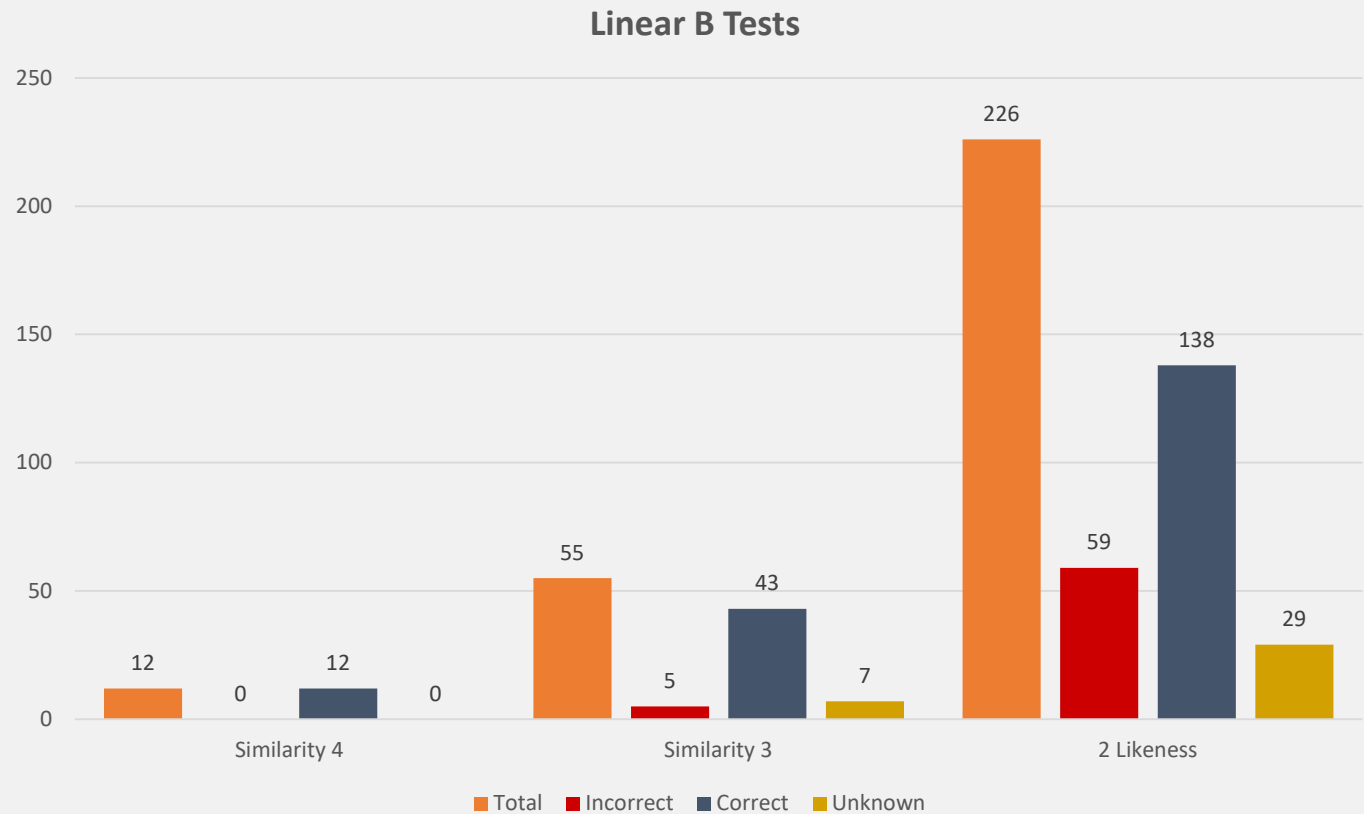
Loop	Word 1	Word 2	Similarity
1	walk	walk	0
2	walk	talking	0
3	walk	walking	4
4	walk	wanting	2

Loop	walk	wanting	Similarity
1	w	w	1
2	a	a	2
3	l	n	2



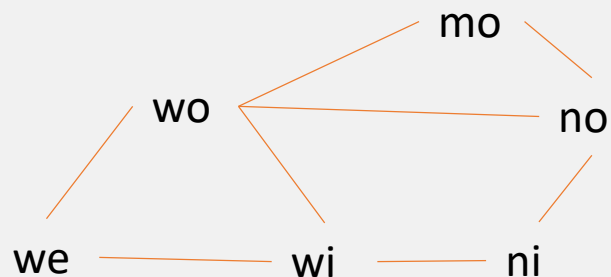
# Finding Inflection: The Results

```
po-ti-ni-ja Confirmed? true
-po-ti-ni-ja-we-jo
-po-ti-ni-ja-we
-po-ti-ni-ja-wi-jo
u-ru-pi-ja-jo Confirmed? false
-u-ru-pi-ja-jo
a-ko-so-ta Confirmed? false
-a-ko-so-ta
-a-ko-so-ta-o
po-ro-u-te Confirmed? false
-po-ro-u-te-u
-po-ro-u-te-we
```



# Creating the Connections: Original Work

- Kober showed how characters are connected
  - Computerise this process
- Predictable patterns, evidence of inflection
- Then this is plot on a graph



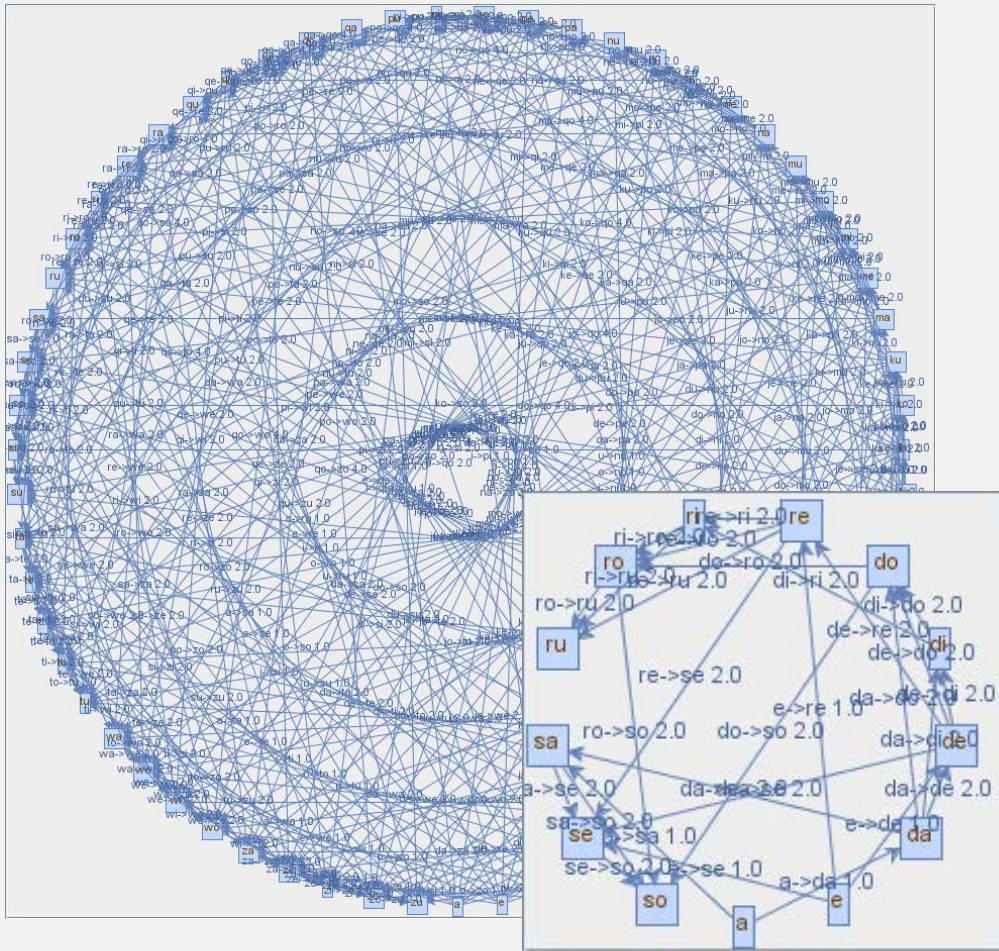
Ser-vu-s(a) -> Servus  
 Ser-vu-m(a) -> Servum  
 Ser-vi -> Servi

Same word -  
 different case

First characters are the same, next character is the same, the next character likely shares a consonant  
 character likely shares a vowel

	Type A				Type B											
Case I	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷
Case II	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷
Case III	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷	𐌸	𐌺	𐌶	𐌷

# Creating the Connections: Results



- Graph
  - Node -> A Linear B character
  - Edge -> A shared vowel or consonant
  - Weight -> How often it appears
- Seed the graph with likely values
  - da, ma, mi, ni, so, do, su, du
- Plot onto a table

# Final Grid

	a	e	i	o	u
M	ma	me	mi	mo	mu
N	na	ne	null	no	nu
D	da	de	di	do	du
J	ja	je	ni	jo	ju
K	ka	ke	ki	ko	ku
P	pa	pe	pi	po	pu
Q	qa	qe	qi	qo	qu
R	ra	re	ri	ro	ru
S	sa	se	si	so	su
T	ta	te	ti	to	tu
Z	za	ze	zi	zo	zu
W	wa	we	wi	wo	null

# Conclusion

- It is possible to replicate the decipherment of Linear B computationally
  - Different approach than typical Machine Learning decipherments
- Working with limitations can encourage creative solutions
- Interdisciplinary projects are great sources of personal growth

# Thank you for listening

## Any Questions?



[@InsiderPhD](https://twitter.com/InsiderPhD)



[K.Paxton-Fear@cranfield.ac.uk](mailto:K.Paxton-Fear@cranfield.ac.uk)



[www.somewebpage.com](http://www.somewebpage.com)



<https://github.com/greenpencil>

My Linear B datasets are available and free for use

<https://github.com/InsiderPhD/Linear-B-Dataset>

My inflection algorithm is available and free for use

<https://github.com/greenpencil/Java-Inflection-Algorithm>