Використання прихованих марківських моделей для декодування тексту

Ivan Zhytkevych

December 14, 2022

0 Prerequisites

У цій доповіді буде розглянуто навчання прихованої марківської моделі для дослідження структури українського алфавіту та використання такої ж моделі для декодування зашифрованого тексту.

Код доступний на GitHub.

0.1 Resources

Для навчання та декодування потрібно мати достатні масиви даних. Тому було взято декілька екземплярів:

1. Марко Вовчок. Дев'ять братів.
2. Марко Вовчок. Інститутка.
3. Марко Вовчок. Кармелюк.
4. Марко Вовчок. Три долі.
5. Пантелеймон Куліш. Чорна рада.
6. Пантелеймон Куліш. Огненний змій

0.2 Tools

Оскільки алгоритм Баума-Велша, який застосовувався для отримання результатів у даній роботі, може вимагати велику кількість ітерацій було вирішено використовувати для таких обчислень більш швидку мову, ніж Python. Отже було обрано мову Nim, яка є компільованою та має статичну типізацію, що вже надає більше швидкості у таких обчисленнях. У той же час її синтаксис є простим:

```
proc computePLog*(C: seq[float64], T: int): float64 =
  var lp: float64 = 0
  for i in 0 ..< T:
    lp += math.log(C[i], math.E)
  return -lp

```

Listing 1: Nim example

Для дослідження отриманих даних буде все ж використовуватися Python через вже відомі засоби мови Python (Numpy, Pandas, matplotlib, ...), що є знайомі автору.

1 Структура українського алфавіту

Two different procedures of stochastic matrix initializations were programmed to pass the resulted data as input into Baum-Welch procedure.

Name	Procedure
normal	normal distribution $\Rightarrow$ normalized on $[0, 1]$
evenly	full matrix of $\frac{1}{\#\text{columns}}$ $\Rightarrow$ applied a little shift $\Rightarrow$ normalization

1.1 Прихована марківська модель з двома станами

Марко Вовчок. Дев'ять братів.

Parameter	Value
N	2
Epsilon (precision)	2e-3
Min iterations	20
$P^{(0)}$ distribution	normal
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	evenly

Two clusterization methods were used:

1. **maximum** which picks the state with the highest probability
2. **k-means** clustering

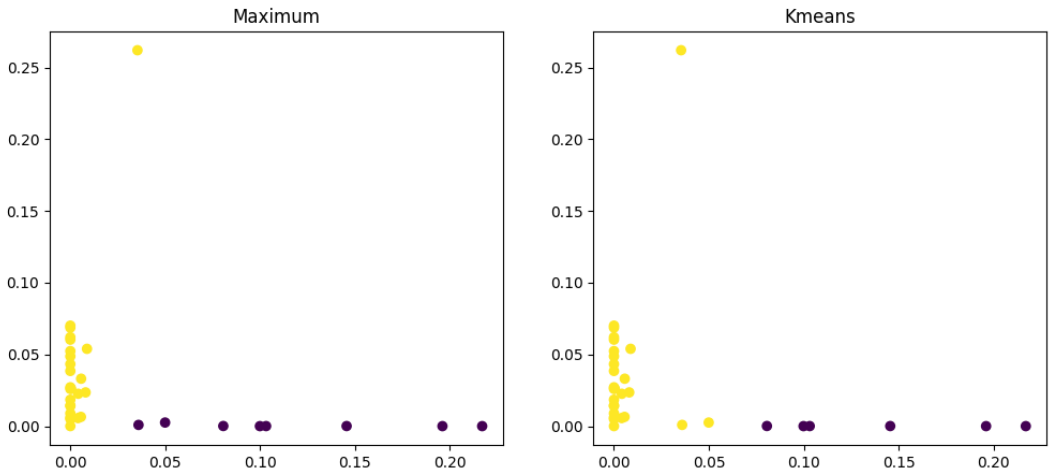
[Clustering by maximum]

1. 'и' 'а' 'о' 'і' 'у' 'е' 'я' 'ь'
2. ' ' 'ж' 'л' 'с' 'б' 'д' 'в' 'к' 'є' 'н' 'п' 'т' 'м' 'щ' 'г' 'р' 'ї' 'ч' 'з' 'ц' 'х' 'ю' 'ш' 'й' 'ф'

[Clustering by kmeans]

1. 'и' 'а' 'о' 'і' 'у' 'е'
2. ' ' 'ж' 'л' 'с' 'б' 'д' 'в' 'к' 'є' 'н' 'п' 'т' 'м' 'щ' 'я' 'г' 'р' 'ї' 'ч' 'з' 'ц' 'х' 'ю' 'ш' 'й' 'ь' 'ф'

Both of the clustering methods divide our objects into two separate groups: vowels and consonants with the space.



$$A = \begin{pmatrix} 0.0007 & 0.9993 \\ 0.6302 & 0.3698 \end{pmatrix}$$

From the A matrix we may conclude that there's a very high probability to go into second state after the first one (vowel → consonant).

Let's also analyse several other texts...

Марко Вовчок. Інститутка.

Parameter	Value
N	2
Epsilon (precision)	2e-3
Min iterations	20
$P^{(0)}$ distribution	evenly
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	normal
Observation space size	63715

Converged after **144 iterations**.

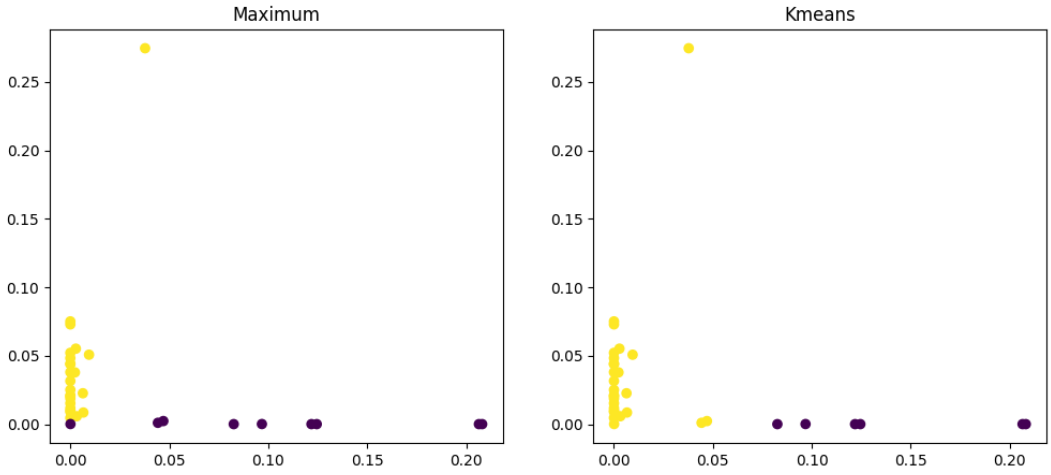
[Clustering by maximum]

1. 'е' 'у' 'і' 'и' 'ь' 'я' 'о' 'а' 'ф'
2. ' ' 'т' 'г' 'ш' 'в' 'ч' 'н' 'к' 'л' 'ю' 'д' 'с' 'щ' 'й' 'р' 'б' 'з' 'ж' 'м' 'ц' 'п' 'х' 'є' 'ї'

[Clustering by kmeans]

1. 'е' 'у' 'і' 'и' 'о' 'а'
2. ' ' 'т' 'г' 'ш' 'в' 'ч' 'н' 'к' 'л' 'ю' 'д' 'ь' 'с' 'я' 'щ' 'й' 'р' 'б' 'з' 'ж' 'м' 'ц' 'п' 'х' 'є' 'ї' 'ф'

The results are pretty the same except the presence of letter 'ф' in the first group while clustering by maximum.



$$A = \begin{pmatrix} 0.0011 & 0.9989 \\ 0.6147 & 0.3853 \end{pmatrix}$$

1.2 Прихована марківська модель з трьома станами

Марко Вовчок. Інститутка.

Parameter	Value
N	3
Epsilon (precision)	2e-3
Min iterations	20
$P^{(0)}$ distribution	evenly
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	normal
Observation space size	63715

Converged after **271 iterations**.

[Clustering by maximum]

1. 'т' 'г' 'в' 'ч' 'н' 'к' 'л' 'д' 'с' 'щ' 'р' 'б' 'ж' 'м' 'ц' 'п' 'х'
2. 'е' 'у' 'і' 'и' 'ь' 'я' 'о' 'а'
3. ' ' 'ш' 'ю' 'й' 'з' 'є' 'ї' 'ф'

[Clustering by kmeans]

1. 'т' 'г' 'ш' 'в' 'ч' 'н' 'к' 'л' 'ю' 'д' 'ь' 'с' 'я' 'щ' 'й' 'р' 'б' 'з' 'ж' 'м' 'ц' 'п' 'х' 'є' 'ї' 'ф'
2. 'е' 'у' 'і' 'и' 'о' 'а'
3. ' '

$$A = \begin{pmatrix} 0.0114 & 0.9886 & 0.0000 \\ 0.4226 & 0.0000 & 0.5774 \\ 0.6476 & 0.0420 & 0.3104 \end{pmatrix}$$

The third state cluster in a case of maximum clusterization may seem a weird one, but my opinion is that this class is a *one-in-the-middle*. If we look at the A matrix we can see that the transitions from the third class are all > 0. We cannot go into the third class from the first one (the consonants defined there).

But the K-Means clustering method clearly defined a space in a separate class with vowels and consonants in a two different classes.

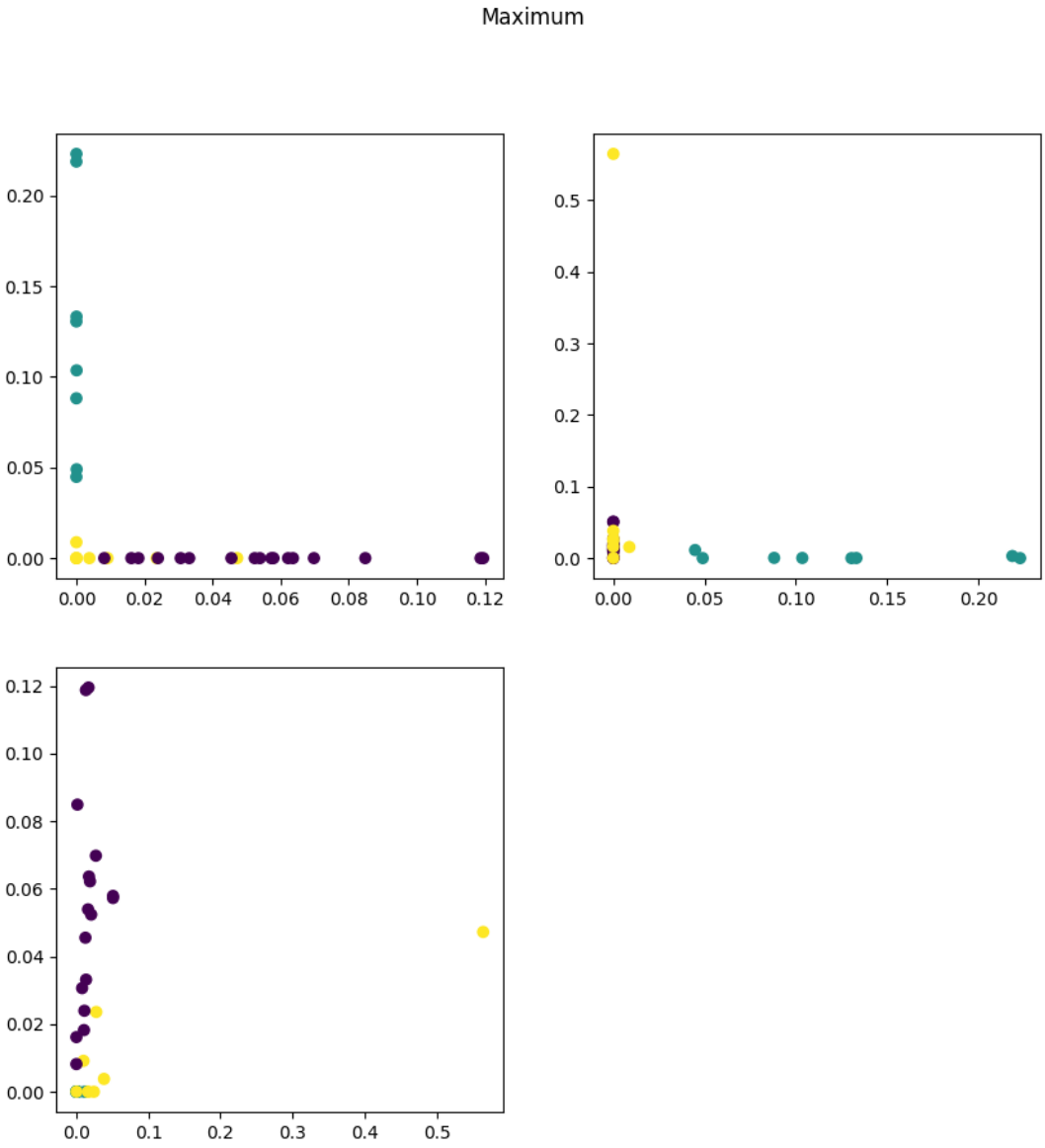


Figure 1: Clustering by maximum of HMM with 3 states trained on M.Vovchok 'Instytutka'

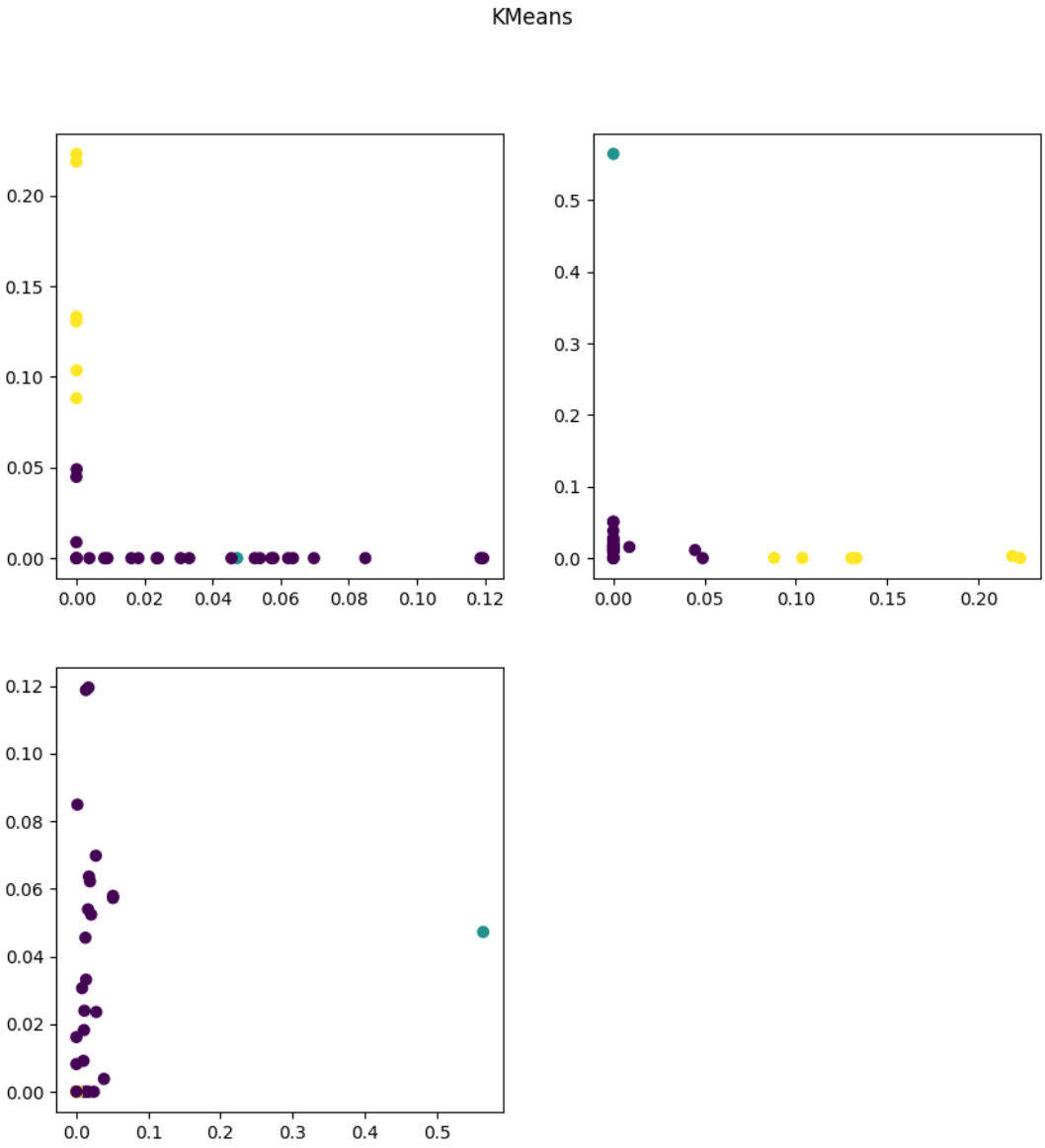


Figure 2: Clustering by kmeans of HMM with 3 states trained on M.Vovchok 'Instytutka'

1.3 Прихована марківська модель з чотирма станами

Марко Вовчок. Інститутка.

Parameter	Value
N	4
Epsilon (precision)	1e-5
Min iterations	20
$P^{(0)}$ distribution	normal
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	normal
Observation space size	63715

Converged after **999 iterations**.

[Clustering by maximum]

1. 'е' 'у' 'і' 'и' 'ь' 'о' 'а'
2. 'т' 'г' 'ш' 'ч' 'н' 'к' 'л' 'д' 'щ' 'р' 'б' 'ж' 'м' 'ц' 'п' 'х'
3. ' ' 'ю' 'є'
4. 'в' 'с' 'я' 'й' 'з' 'ї' 'ф'

[Clustering by kmeans]

1. 'е' 'у' 'і' 'и' 'о' 'а'
2. 'г' 'ш' 'ч' 'ю' 'ь' 'я' 'щ' 'й' 'з' 'ж' 'ц' 'х' 'є' 'ї' 'ф'
3. ' '
4. 'т' 'в' 'н' 'к' 'л' 'д' 'с' 'р' 'б' 'м' 'п'

$$A = \begin{pmatrix} 0.0000 & 0.3996 & 0.5983 & 0.0021 \\ 0.9968 & 0.0009 & 0.0023 & 0.0000 \\ 0.0295 & 0.5959 & 0.0002 & 0.3744 \\ 0.0000 & 0.6115 & 0.0000 & 0.3885 \end{pmatrix}$$

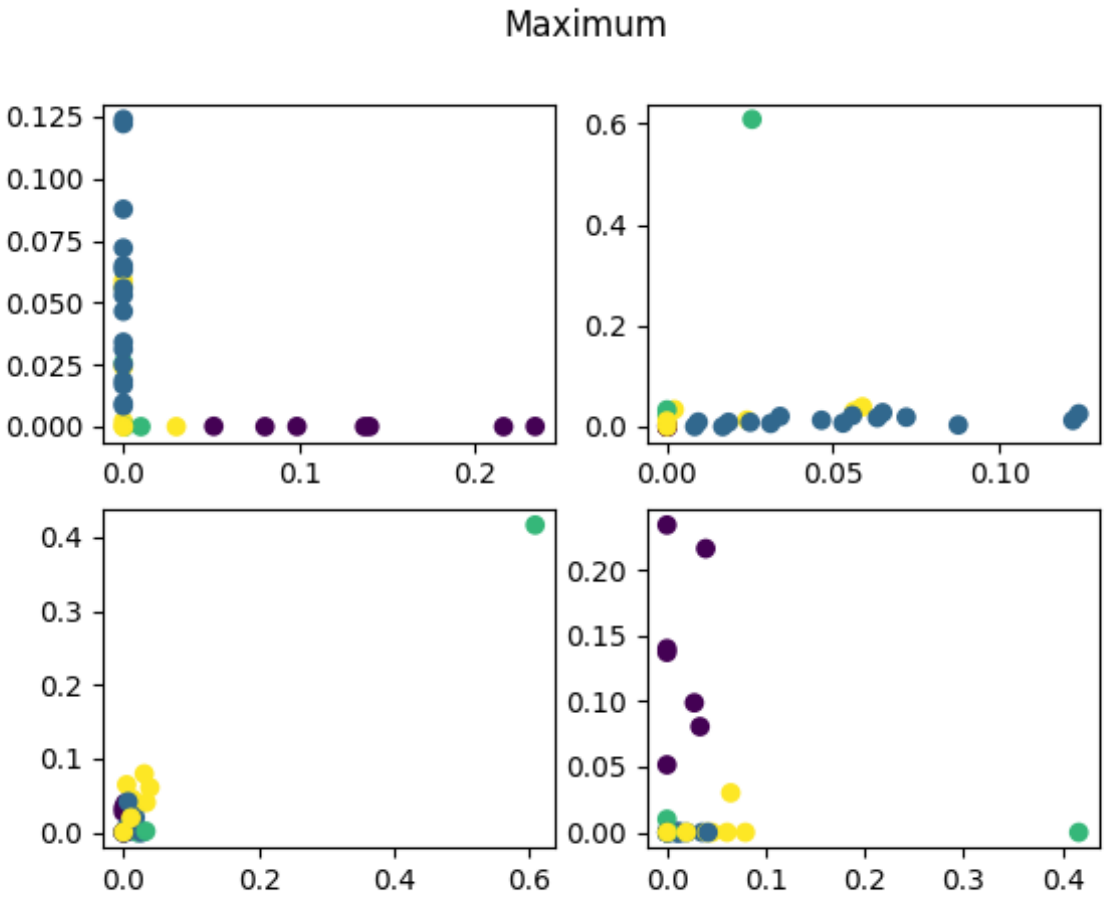


Figure 3: Clustering by maximum of HMM with 4 states trained on M.Vovchok 'Instytutka'

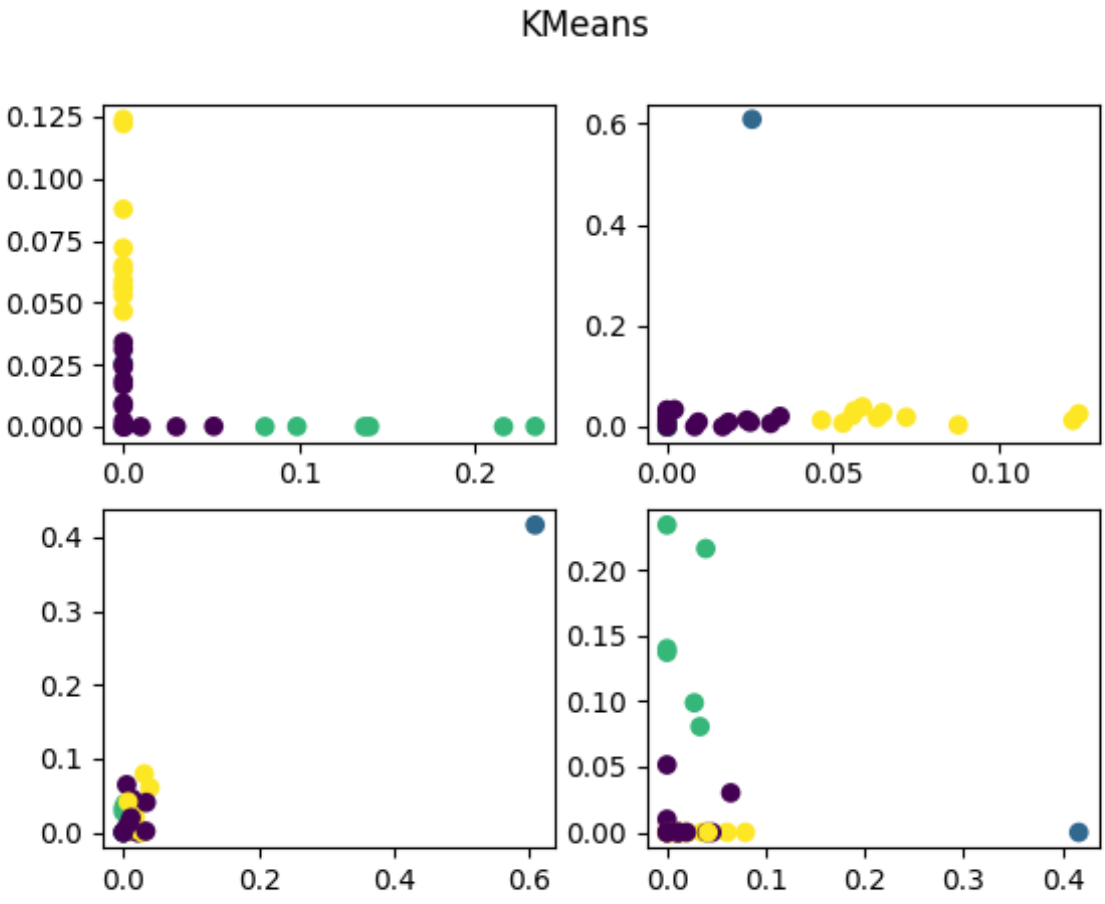


Figure 4: Clustering by kmeans of HMM with 4 states trained on M.Vovchok 'Instytutka'

П. Куліш. Чорна Рада.

Parameter	Value
N	4
Epsilon (precision)	1e-2
Min iterations	35
$P^{(0)}$ distribution	normal
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	normal
Observation space size	286868

Converged after **141 iterations**.

[Clustering by maximum]

1. ' '
2. 'о' 'е' 'і' 'у' 'и' 'а' 'я' 'ь'
3. 'в' 'к' 'є' 'ж' 'х' 'з' 'ш' 'м' 'й' 'ю' 'ї'
4. 'п' 'с' 'н' 'р' 'д' 'б' 'л' 'г' 'т' 'ц' 'щ' 'ч' 'ґ' 'ф'

[Clustering by kmeans]

1. ' '
2. 'п' 'с' 'н' 'р' 'д' 'є' 'ж' 'х' 'б' 'я' 'л' 'ь' 'г' 'ш' 'т' 'ю' 'ц' 'ї' 'щ' 'ч' 'ґ' 'ф'
3. 'о' 'е' 'і' 'у' 'и' 'а'
4. 'в' 'к' 'з' 'м' 'й'

$$A = \begin{pmatrix} 0.0195 & 0.1296 & 0.0922 & 0.7588 \\ 0.3620 & 0.0000 & 0.1890 & 0.4490 \\ 0.6830 & 0.0000 & 0.0345 & 0.2826 \\ 0.0000 & 0.8853 & 0.0000 & 0.1147 \end{pmatrix}$$

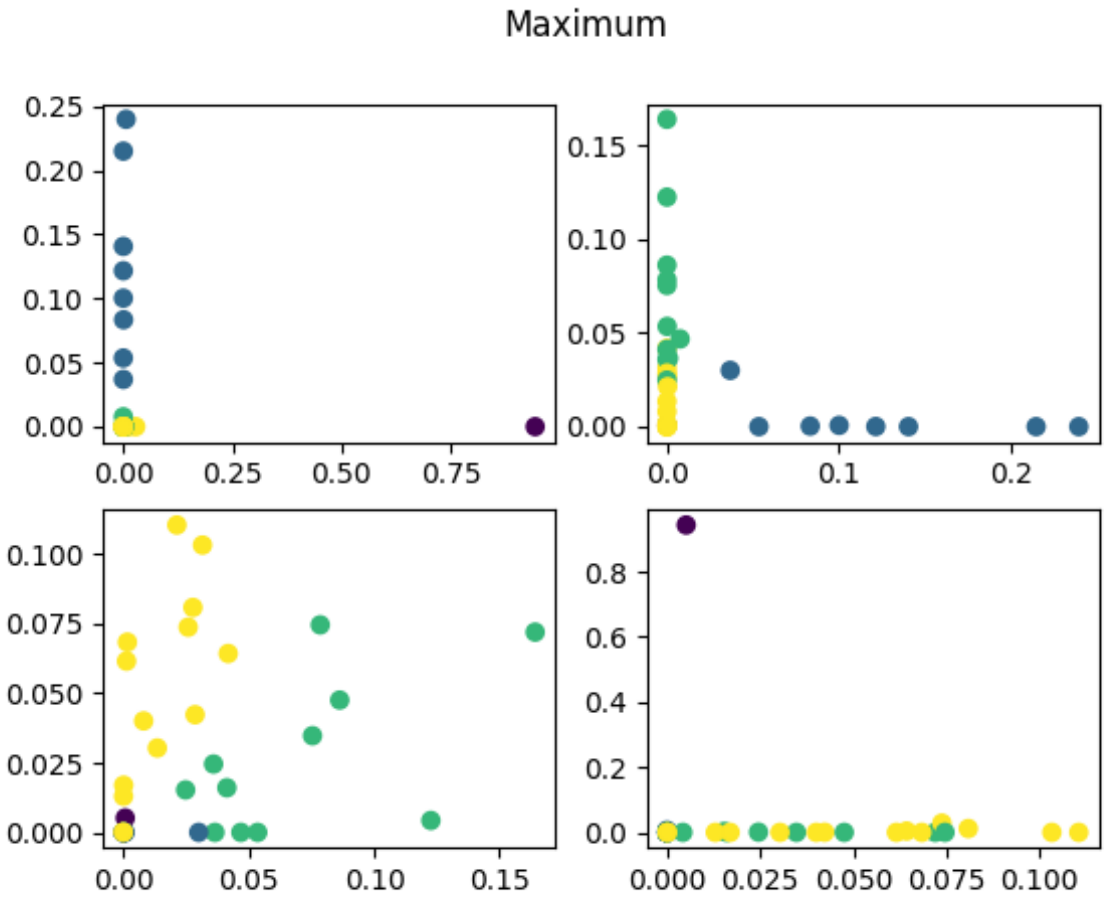


Figure 5: Clustering by maximum of HMM with 4 states trained on P.Kylish 'Chorna Rada'

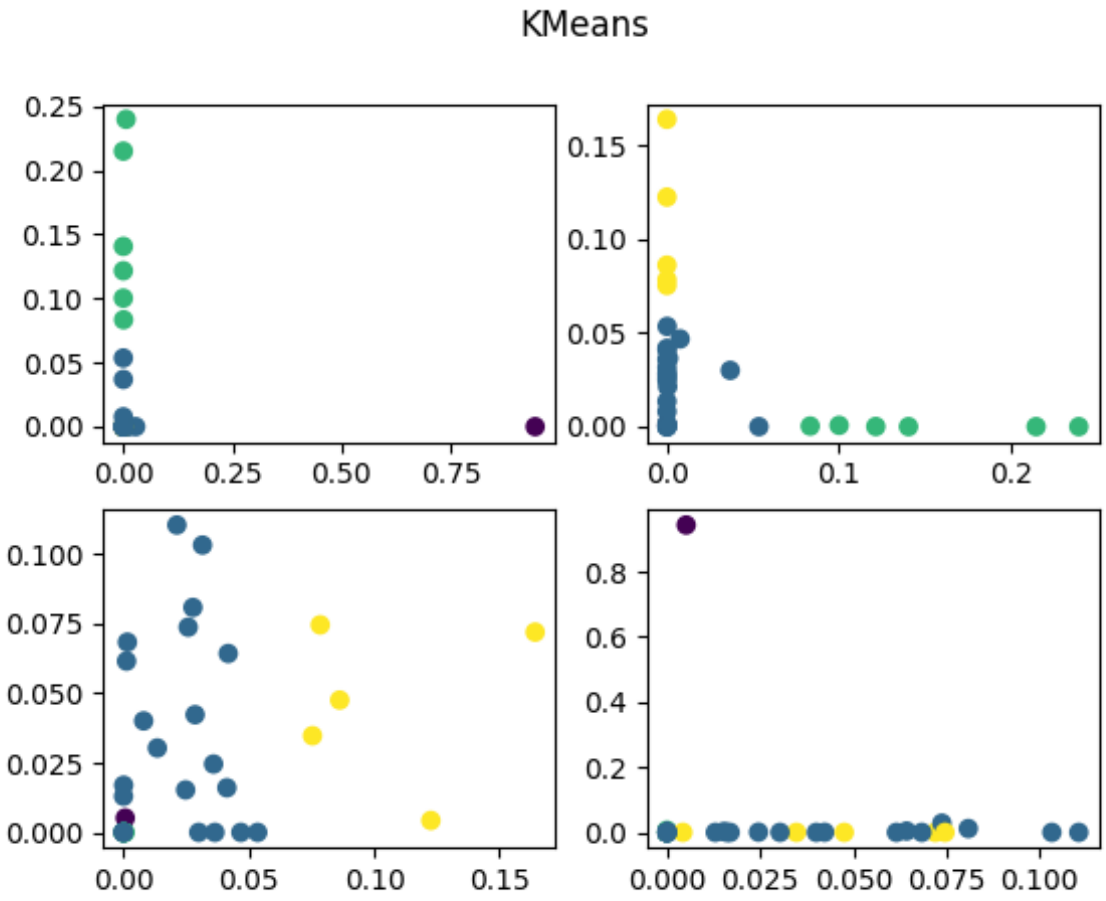


Figure 6: Clustering by kmeans of HMM with 4 states trained on P.Kylish 'Chorna Rada'

П. Куліш. Чорна Рада. (не враховуючи пробіли)

Parameter	Value
N	4
Epsilon (precision)	1e-2
Min iterations	35
$P^{(0)}$ distribution	normal
$A^{(0)}$ distribution	normal
$B^{(0)}$ distribution	normal
Observation space size	234671

Converged after **194 iterations**.

[Clustering by maximum]

1. 'о' 'е' 'і' 'у' 'и' 'а'
2. 'с' 'к' 'я' 'ь' 'ш' 'г'
3. 'п' 'н' 'р' 'д' 'ж' 'б' 'л' 'г' 'м' 'т' 'ц' 'щ' 'ч' 'ф'
4. 'в' 'є' 'х' 'з' 'й' 'ю' 'ї'

[Clustering by kmeans]

1. 'с' 'ь'
2. 'є' 'ж' 'х' 'б' 'я' 'з' 'г' 'ш' 'й' 'ю' 'ц' 'ї' 'щ' 'ч' 'ґ' 'ф' 'щ' 'ч' 'г' 'ф'
3. 'о' 'е' 'і' 'у' 'и' 'а'
4. 'п' 'в' 'н' 'р' 'к' 'д' 'л' 'м' 'т'

$$A = \begin{pmatrix} 0.0195 & 0.1296 & 0.0922 & 0.7588 \\ 0.3620 & 0.0000 & 0.1890 & 0.4490 \\ 0.6830 & 0.0000 & 0.0345 & 0.2826 \\ 0.0000 & 0.8853 & 0.0000 & 0.1147 \end{pmatrix}$$

I think that the most clear results of HMM with 4 states are observed with the deletion of spaces.

Here we can clearly see a set of sonorus letters *'п' 'в' 'н' 'р' 'к' 'д' 'л' 'м' 'т'* in a case of the clustering by K-Means.

Funny thing that the first class *'с' 'ь'* is composed. Also useful to notice that this composition has a much higher frequence of 'сь'.

Чорна рада	0.00663371306663692
Дев'ять братів	0.003378557422226933
Кармелюк	0.003376617962773829

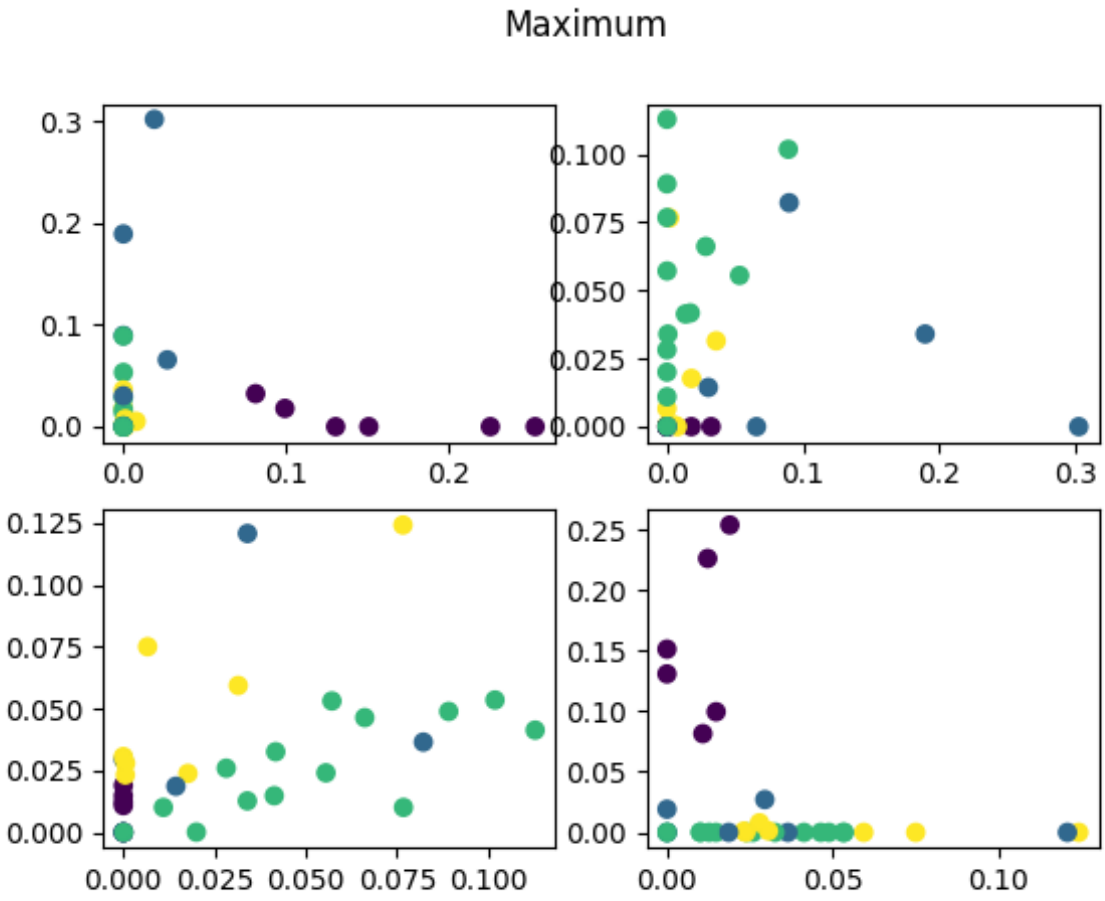


Figure 7: Clustering by maximum of HMM with 4 states trained on P.Kylish 'Chorna Rada'

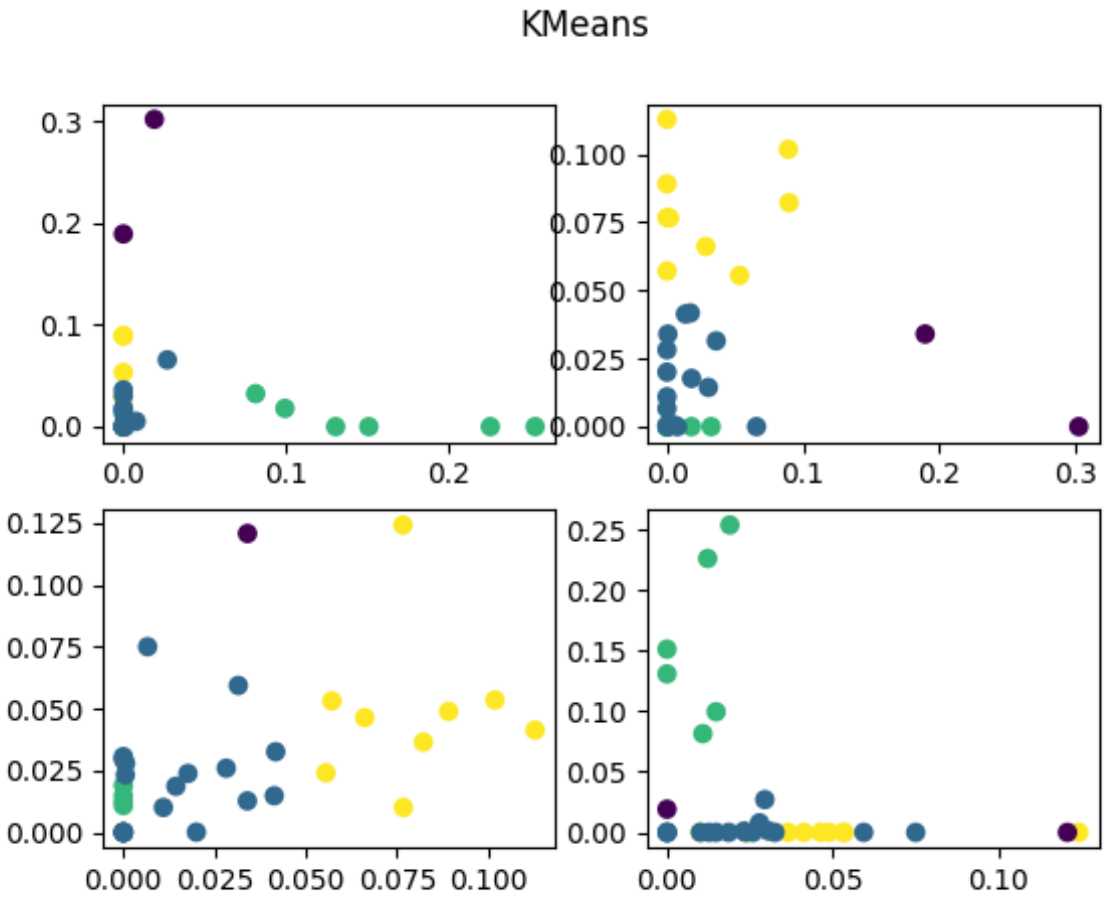


Figure 8: Clustering by kmeans of HMM with 4 states trained on P.Kylish 'Chorna Rada'

2 Декодування зашифрованого тексту

As a encoding algorithm a simple alphabet shift was used.

Alphabet: ’а’, ’б’, ’в’, ’г’, ’ґ’, ’д’, ’е’, ’є’, ’ж’, ’з’, ’и’, ’і’, ’ї’, ’й’, ’к’, ’л’, ’м’, ’н’, ’о’, ’п’, ’р’, ’с’, ’т’, ’у’, ’ф’, ’х’, ’ц’, ’ч’, ’ш’, ’щ’, ’ь’, ’ю’, ’я’

```
encoded = ''
l = len(alphabet)
for s in text:
  encoded += alphabet[(alphabet.index(s) + shift) % l]
return encoded

```

Марко Вовчок. Шифрування та декодування на основі одного тексту

Використаємо текст **Кармелюк** Марка Вовчка: зашифруємо його та використаємо для побудови параметрів для дешифрування.

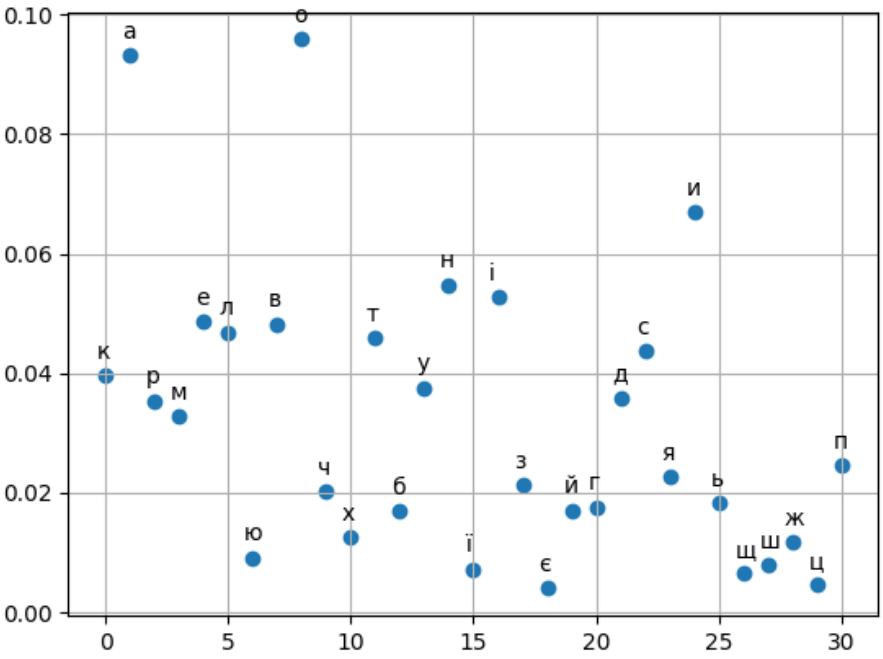


Figure 9: Частоти літер твору *Кармелюк*

Шифруємо текст зі зсувом 1: *лбснємялгпгшплн...*

Для побудови початкових параметрів для ПММ використаємо цей же твір.

Матриця A будується на біграмах з тексту, що будуть відповідати індексам літер з алфавіту (i, j) відповідаючи рядкам та стовпцям матриці; обчислюємо кількість таких переходів літер на основі цих біграм, додаємо та нормуємо, додаючи перед цим до всіх значень у матриці 5 для перетворення нульових ймовірностей у відносно близькі до нуля.

	0	1	2	3	4	5	...
0	0.005726	0.019001	0.078605	0.019261	0.001301	0.045809	...
1	0.120192	0.009615	0.009615	0.006010	0.006010	0.006010	...
2	0.115031	0.015950	0.019333	0.014500	0.002417	0.025133	...
3	0.138211	0.006969	0.010453	0.005807	0.005807	0.009292	...
4	0.030303	0.030303	0.030303	0.030303	0.030303	0.030303	...
5	0.127054	0.007585	0.022756	0.003793	0.003161	0.008217	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱
30	0.012387	0.020270	0.037162	0.012387	0.005631	0.041667	...
31	0.024809	0.051527	0.040076	0.015267	0.009542	0.127863	...
32	0.013195	0.016023	0.063148	0.025448	0.004713	0.042413	...

За вектор P візьмемо стаціонарний розподіл:

$$\mu P = \mu$$

$$\begin{aligned}\mu = [&0.08554185, 0.01852065, 0.04605712, 0.01916611,\\ &0.00367294, 0.03521623, 0.04643526, 0.00719039,\\ &0.01397959, 0.02239429, 0.06246084, 0.05001869,\\ &0.01001713, 0.01865465, 0.03848894, 0.04485574,\\ &0.03250084, 0.05180049, 0.08790287, 0.02526563,\\ &0.03470422, 0.0420283, 0.0440098, 0.03663937,\\ &0.00367294, 0.0146474, 0.00770207, 0.02148163\\ &0.01057371, 0.00934949, 0.01976749, 0.01166478, 0.02361854].\end{aligned}$$

Навчаємо ПММ на перших 3000 символах зашифрованого тексту по алгоритму Баума-Велша з 33 станами, точністю в $1e-2$ та мінімальною кількістю ітерацій в 200.

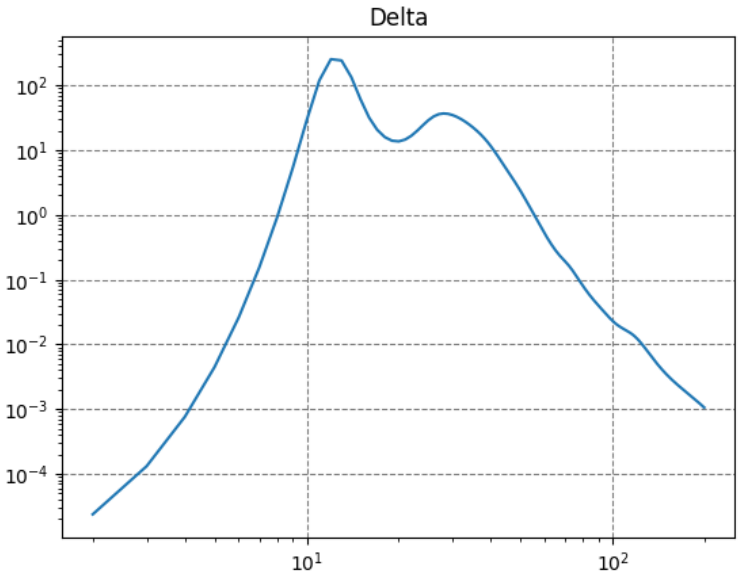


Figure 10: Еволюція дельти під час процедури навчання по параметрам текста Кармелюк на зашифрованому тексті Кармелюк

Отримана у результаті навчання матриця B:

	0	1	2	3	4	5	6	...
а	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
б	0.847381	0.000000	0.000000	0.000000	0.195377	0.000000	0.000000	...
в	0.000000	0.837261	0.000000	0.000000	0.000000	0.000000	0.000000	...
г	0.000000	0.000000	0.865880	0.000000	0.000000	0.000001	0.000000	...
ґ	0.000000	0.114226	0.000000	0.815231	0.000000	0.000000	0.000000	...
д	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
е	0.000000	0.048512	0.000000	0.000000	0.000000	0.771587	0.000000	...
є	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	...
ж	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
з	0.000000	0.000000	0.000033	0.013084	0.000000	0.000268	0.000000	...
и	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
і	0.031037	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
ї	0.046833	0.000000	0.000000	0.000000	0.804481	0.000000	0.000000	...
й	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
к	0.000000	0.000000	0.000000	0.034353	0.000000	0.000000	0.000000	...
л	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
м	0.000000	0.000000	0.000000	0.137331	0.000000	0.000000	0.000000	...
н	0.000000	0.000000	0.133100	0.000000	0.000000	0.000000	0.000000	...
о	0.000000	0.000000	0.000000	0.000000	0.000000	0.054901	0.000000	...
п	0.074706	0.000000	0.000000	0.000000	0.000142	0.000000	0.000000	...
р	0.000000	0.000000	0.000986	0.000000	0.000000	0.000000	0.000000	...
с	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
т	0.000000	0.000000	0.000000	0.000000	0.000000	0.169358	0.000000	...
⋮								
ю	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
я	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...

У такому вигляді вона є транспонованою через технічні причини. Але ми можемо побачити, певну залежність у цій матриці: літера 'а' кодується літерою 'б', 'б' кодується 'в' і так далі. Тобто побачили шифр.

Прогнавши через алгоритм Вітербі отримуємо послідовність станів на цьому тексті.

Перші 200 символів розшифрованого текста:

кармелюквовчокмаркохтобувавнаукраїніхтознаукраїнухтобувавізнаєтойнехайзгадає ахтонебувавінезнаєтойнехайсобіуявитьщотамскрізьбіліхатиувишневихсадкахівесною весноюламдужегарноякусісадочкизъцвітутьл'усісоло...

Порахувавши кількість вгаданих літер на всьому тексті отримуємо 98.01160110438461 %.

Протестуємо весь процес на різних текстах.

Нагадаю, що тексти мають таких авторів:

1. Марко Вовчок. Дев'ять братів.
2. Марко Вовчок. Інститутка.
3. Марко Вовчок. Кармелюк.
4. Марко Вовчок. Три долі.
5. Пантелеймон Куліш. Чорна рада.
6. Пантелеймон Куліш. Огненний змій

Інший текст для параметрів моделі.

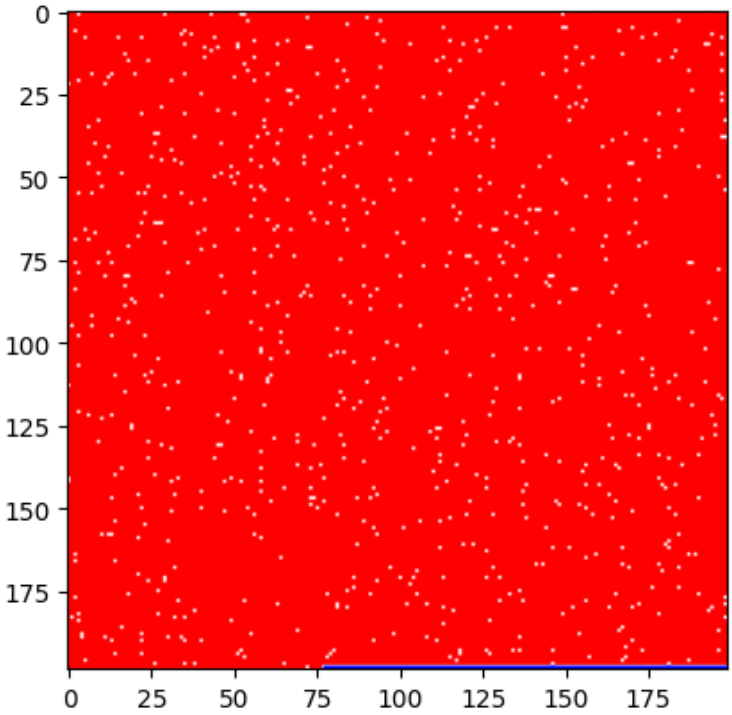


Figure 11: Мапа правильного розкодування текста Кармелюк на основі текста Кармелюк

Візьмемо текст **Кармелюк** для шифрування та текст **Дев'ять братів** як основу для параметрів моделі.

	Розміри тексту для навчання ПММ					
	1000	2000	3000	5000	8000	10000
Точність декодування	88.49	92.07	0.60	0.04	97.95	97.67

Все ж можемо помітити наявність помилок у моделі на кроках у 3000 та 5000 розмірів вибірки.

Та візьмемо інший текст: **Дев'ять братів** для шифрування та текст **Три долі** як основу для параметрів моделі.

	Розміри тексту для навчання ПММ					
	1000	2000	3000	5000	8000	10000
Точність декодування	0.25	89.90	0.20	93.30	0.20	0.20

Отже наша модель могла залишитися у якійсь точці локального мінімуму та видавати поганий результат для задачі декодування. Але загалом зі збільшенням розміру вибірки для навчання ПММ ми бачимо зростання точності декодування.

Як важливий крок для себе я б хотів додати, що спроба не залишатися на мові Python та використати щось нове для себе дала чіткі плоди: я зміг не витрачати багато часу на навчання ПММ (для прикладу на 33 станах та з вибіркою в 3000 символів для досягнення точності в $1e-2$ знадобилося 200 ітерацій та 31 секунда часу, що дає 6.7 в середньому ітерацій на секунду, а при взагалі простих умовах досягалося і 16-18 ітерацій на секунду) та використати його для збільшення розміру вибірки на задачі декодування. Оптимізації у математиці беззаперечно дають добрий результат, як от *Forward-Backward procedure*, але й інструменти, які ми використовуємо, мають відповідати нашим цілям.