Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18527-y

Automated decryption of siri bhoovalaya using cryptography and natural language processing techniques

Jagadeesh Sai D[1]

Received: 27 February 2022 / Revised: 4 January 2024 / Accepted: 29 January 2024

Abstract
"The Siri Bhoovalaya is a seminal work of literature, believed to have been composed approximately a millennium ago, which encompasses diverse information encrypted using numerals of the Kannada language—a predominant language of southern India. Currently, only a portion of this enigmatic text is accessible, and deciphering its content remains largely a manual endeavor. This article presents a novel model designed to automate the conversion of these Kannada numerals into phonetic alphabets of the designated language. Subsequent to this conversion, algorithms rooted in Natural Language Processing (NLP) techniques are utilized to form coherent words. These algorithms adhere to the linguistic and grammatical structures of the target language. Through this research, we aim to establish an initial technical blueprint to shed light on the profound content encapsulated within this age-old masterpiece."

Keywords Software environments · Cryptographic algorithms natural · Language processing · Lexical semantics · Cryptographic primitives

1 Introduction

Siri Bhoovalaya is a renowned multi-lingual literary masterpiece (see [2, 9]) that dates back approximately a millennium, authored by the Jain monk Muni Kumudendu in Karnataka, India. One of its standout features is its unique composition—entirely in the numerals of the Kannada language. Furthermore, this work is so intricately designed that applying varied decryption methods unveils texts in different languages.

Each segment of this extensive work is termed a 'Chakra,' while the method to decode a chakra is known as a 'Bandha.' The text consists of an impressive 16,000 chakras, organized into 56 chapters and further grouped into 9 Khandas. Cumulatively, this amounts to 600,000 shlokas, encompassing roughly 1,400,000 characters. To put its magnitude into perspective, Siri Bhoovalaya is approximately sixfold the size of

✉ Jagadeesh Sai D
djsai@msrit.edu

[1] Department of Information Science & Engineering, Ramaiah Institute of Technology, Bangalore, India

the epic Indian tale, Mahabharata. It employs intricate patterns such as Chakrabandha, Hamsabandha, Varapadmabandha, Sagarabandha, Sarasabandha, Kruanchabandha, Mayurabandha, Ramapabandha, Nakhabandha, among others. Recognizing these patterns is crucial to determine the appropriate decryption technique. The chakras span diverse fields, from religious doctrines like Jainism, Vedas, Ayurveda, and astrology, to scientific disciplines including mathematics, physics, chemistry, and astronomy.

Each chakra aligns with the Saangathya metre, a hallmark of Kannada poetry. Specifically, every chakra presents a 27×27 matrix filled with integers, ranging between 1 and 64. Impressively, every integer corresponds to a phonetic alphabet in the Kannada language. When deciphered, these chakras translate into verses spanning 718 dialects prevalent across the Indian subcontinent. These dialects are comprised of 18 major languages, such as Sanskrit, Prakrit, Telugu, Tamil, Pali, Marathi, Apabhramsha, to name a few, in addition to 700 other minor dialects.

Despite its impressive scope, this vast composition has largely remained obscure, primarily because its numeric-centric nature makes decryption daunting. Consequently, there arose a prevailing belief that the original work, along with the supposed five extant copies, had vanished. This notion persisted until the 1950s when Pundit Yellappa Shastri unveiled the sole surviving copy. However, this version only encompasses 1,270 chakras from the Prathama Khanda, termed as Mangala Prabhruta. To date, merely about 8% of its content has been revealed, necessitating the application of diverse cryptographic techniques, including substitution, transposition, and steganography.

The prevailing sentiment among scholars is that Muni Kumudendu didn't encrypt this work for the sake of hiding its contents. Instead, he harnessed these methods to ingeniously embed content from various languages into a singular cipher text.

2 Cryptographic techniques

2.1 Mono-alphabetic substitution cipher

In this technique, there exists a substitution table that gives the mapping from every plain alphabet to a cipher alphabet. The plain alphabet is encrypted by replacing it with the cipher alphabet given by the substitution table. Similarly, decryption occurs by replacing the cipher alphabet with the plain alphabet in accordance with the table [5].

This concept is used in Siri Bhoovalaya where there exists a substitution table for every dialect. Figure 1 shows the substitution table for Kannada language (c.f [2, 13]).

The substitution table gives the mapping between a phonetic alphabet of the Kannada language and an integer between 1 and 64.

2.2 Transposition cipher

A transposition cipher is a encryption method where the cipher text is a permutation of the plain text and requires to be traversed in a particular order [6, 14].

There are a large number of transposition ciphers, two of which are described below.

1	ಅ	17	ಏಾ	33	ಚ್	49	ಫ್
2	ಆ	18	ಏಾ	34	ಛ್	50	ಬ್
3	ಆಾ	19	ಎ	35	ಜ್	51	ಭ್
4	ಇ	20	ಐೊ	36	ಝ್	52	ಮ್
5	ಈ	21	ಐೊೆ	ೂ37	ಞ್	53	ಯ್
6	ಈೀ	22	ಒ	38	ಟ್	54	ರ್
7	ಉ	23	ಒೊ	39	ಠ್	55	ಲ್
8	ಊ	24	ಒೋ	40	ಡ್	56	ವ್
9	ಊೂ	25	ಔ	41	ಢ್	57	ಶ್
10	ಋೂ	26	ಔಾ	42	ಣ್	58	ಷ್
11	ಋೂೃ	27	ಔಾಾ	43	ತ್	59	ಸ್
12	ಋೂೄ	28	ಕ್	44	ಥ್	60	ಹ್
13	ಳ	29	ಖ್	45	ದ್	61	ಂ
14	ಳು	30	ಗ್	46	ಧ್	62	ಃ
15	ಳೂ	31	ಘ್	47	ನ್	63	...
16	ಏ	32	ಙ್	48	ಪ್	64	::

Fig. 1 Substitution table for Kannada language

2.2.1 Chakra bandha

This is a deciphering technique introduced by Muni Kumudendu. The bandha gives a transposition matrix as shown in Fig. 2. The cells of the chakra must be traversed as illustrated in the figure from cells 1 to 729. Cell 1 is situated at row 1, column 14, cell 2 is situated at row 27, column 15 and so on until cell 729 which is situated at row 27, column 14.

2.2.2 Navmaank bandha

Here, a chakra can be divided into a set of 3×3 tiles. Each tile is a 9×9 matrix of cells as shown in Fig. 3. Cell traversal is in accordance with the transposition matrix shown in Fig. 4. The position of the tiles varies with chapters as shown in Fig. 5.

2.3 Steganographic schemes

Stenography is the technique of concealing one message within another [7, 15, 16]. This technique is widely applied in Siri Bhoovalaya. For instance, when Chakra 1–1-1 is deciphered using Chakra Bandha and the transposition table for Kannada language, the result is a Kannada text. When the first character of each line of Kannada text is assembled, it gives a Gatha in Prakrit. Similarly, when characters in the center of each line of Kannada text is aggregated, it results in a Sanskrit shloka.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	380	409	438	467	496	525	554	583	612	641	670	699	728	1	30	59	88	117	146	175	204	233	262	291	320	349	378
2	408	437	466	495	524	553	582	611	640	669	698	727	27	29	58	87	116	145	174	203	232	261	290	319	348	377	379
3	436	465	494	523	552	581	610	639	668	697	726	26	28	57	86	115	144	173	202	231	260	289	318	347	376	405	407
4	464	493	522	551	580	609	638	667	696	725	25	54	56	85	114	143	172	201	230	259	288	317	346	375	404	406	435
5	492	521	550	579	608	637	666	695	724	24	53	55	84	113	142	171	200	229	258	287	316	345	374	403	432	434	463
6	520	549	578	607	636	665	694	723	23	52	81	83	112	141	170	199	228	257	286	315	344	373	402	431	433	462	491
7	548	577	606	635	664	693	722	22	51	80	82	111	140	169	198	227	256	285	314	343	372	401	430	459	461	490	519
8	576	605	634	663	692	721	21	50	79	108	110	139	168	197	226	255	284	313	342	371	400	429	458	460	489	518	547
9	604	633	662	691	720	20	49	78	107	109	138	167	196	225	254	283	312	341	370	399	428	457	486	488	517	546	575
10	632	661	690	719	19	48	77	106	135	137	166	195	224	253	282	311	340	369	398	427	456	485	487	516	545	574	603
11	660	689	718	18	47	76	105	134	136	165	194	223	252	281	310	339	368	397	426	455	484	513	515	544	573	602	631
12	688	717	17	46	75	104	133	162	164	193	222	251	280	309	338	367	396	425	454	483	512	514	543	572	601	630	659
13	716	16	45	74	103	132	161	163	192	221	250	279	308	337	366	395	424	453	482	511	540	542	571	600	629	658	687
14	15	44	73	102	131	160	189	191	220	249	278	307	336	365	394	423	452	481	510	539	541	570	599	628	657	686	715
15	43	72	101	130	159	188	190	219	248	277	306	335	364	393	422	451	480	509	538	567	569	598	627	656	685	714	14
16	71	100	129	158	187	216	218	247	276	305	334	363	392	421	450	479	508	537	566	568	597	626	655	684	713	13	42
17	99	128	157	186	215	217	246	275	304	333	362	391	420	449	478	507	536	565	594	596	625	654	683	712	12	41	70
18	127	156	185	214	243	245	274	303	332	361	390	419	448	477	506	535	564	593	595	624	653	682	711	11	40	69	98
19	155	184	213	242	244	273	302	331	360	389	418	447	476	505	534	563	592	621	623	652	681	710	10	39	68	97	126
20	183	212	241	270	272	301	330	359	388	417	446	475	504	533	562	591	620	622	651	680	709	9	38	67	96	125	154
21	211	240	269	271	300	329	358	387	416	445	474	503	532	561	590	619	648	650	679	708	8	37	66	95	124	153	182
22	239	268	297	299	328	357	386	415	444	473	502	531	560	589	618	647	649	678	707	7	36	65	94	123	152	181	210
23	267	296	298	327	356	385	414	443	472	501	530	559	588	617	646	675	677	706	6	35	64	93	122	151	180	209	238
24	295	324	326	355	384	413	442	471	500	529	558	587	616	645	674	676	705	5	34	63	92	121	150	179	208	237	266
25	323	325	354	383	412	441	470	499	528	557	586	615	644	673	702	704	4	33	62	91	120	149	178	207	236	265	294
26	351	353	382	411	440	469	498	527	556	585	614	643	672	701	703	3	32	61	90	119	148	177	206	235	264	293	322
27	352	381	410	439	468	497	526	555	584	613	642	671	700	729	2	31	60	89	118	147	176	205	234	263	292	321	350

Fig. 2 Chakra Bandha transposition table

Rows : 1 to 27	Columns : 1 to 27 of the Chakra		
	Tile 1 9x9 Matrix	Tile 2 9x9 Matrix	Tile 3 9x9 Matrix
	Tile 4 9x9 Matrix	Tile 5 9x9 Matrix	Tile 6 9x9 Matrix
	Tile 7 9x9 Matrix	Tile 8 9x9 Matrix	Tile 9 9x9 Matrix

Fig. 3 Chakra divided into 9 tiles

The chakra's exclusive numerical composition and the complex cryptographic techniques re- quired for decryption have warranted the involvement of computers [2]. has attempted decryption of chakras using Microsoft Small Basic. This program takes the chakra as input and gives the deciphered characters as output as shown in Fig. 6.

This paper proposes a model to extend the automated decryption of chakra using bandha as given in [2] by incorporating an automated association of these characters to form words. This takes a step closer in understanding the shloka originally encrypted.

Relative Rows \ Relative Column Positions	1	2	3	4	5	6	7	8	9
1	47	58	69	80	1	12	23	34	45
2	57	68	79	9	11	22	33	44	46
3	67	78	8	10	21	32	43	54	56
4	77	7	18	20	31	42	53	55	66
5	6	17	19	30	41	52	63	65	76
6	16	27	29	40	51	62	64	75	5
7	26	28	39	50	61	72	74	4	15
8	36	38	49	60	71	73	3	14	25
9	37	48	59	70	81	2	13	24	35

Fig. 4 Navmaank Bandha transposition table

CH-2			CH-3			CH-4			CH-5			CH-6			CH-7			CH-8		
3	4	5	2	3	4	9	2	3	8	9	2	7	8	9	6	7	8	5	6	7
2	1	6	9	1	5	8	1	4	7	1	3	6	1	2	5	1	9	4	1	8
9	8	7	8	7	6	7	6	5	6	5	4	5	4	3	4	3	2	3	2	9

Fig. 5 Tile transposition scheme

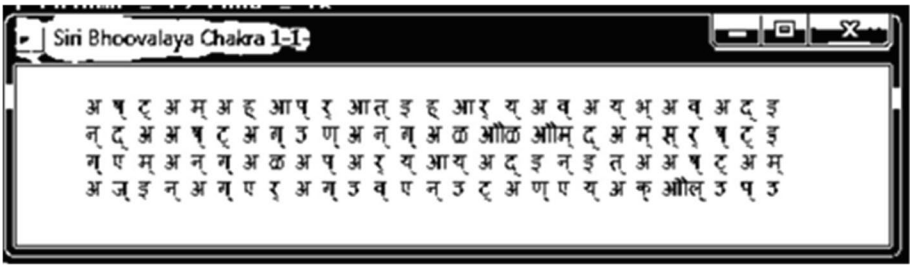

Fig. 6 Screen shot of chakra decrypted by [2]

3 Model

The proposed model for automated decryption and logical association of alphabets into words is shown in Fig. 7.

The model consists of the following components:

3.1 Decryptor

This component of the model takes three inputs:

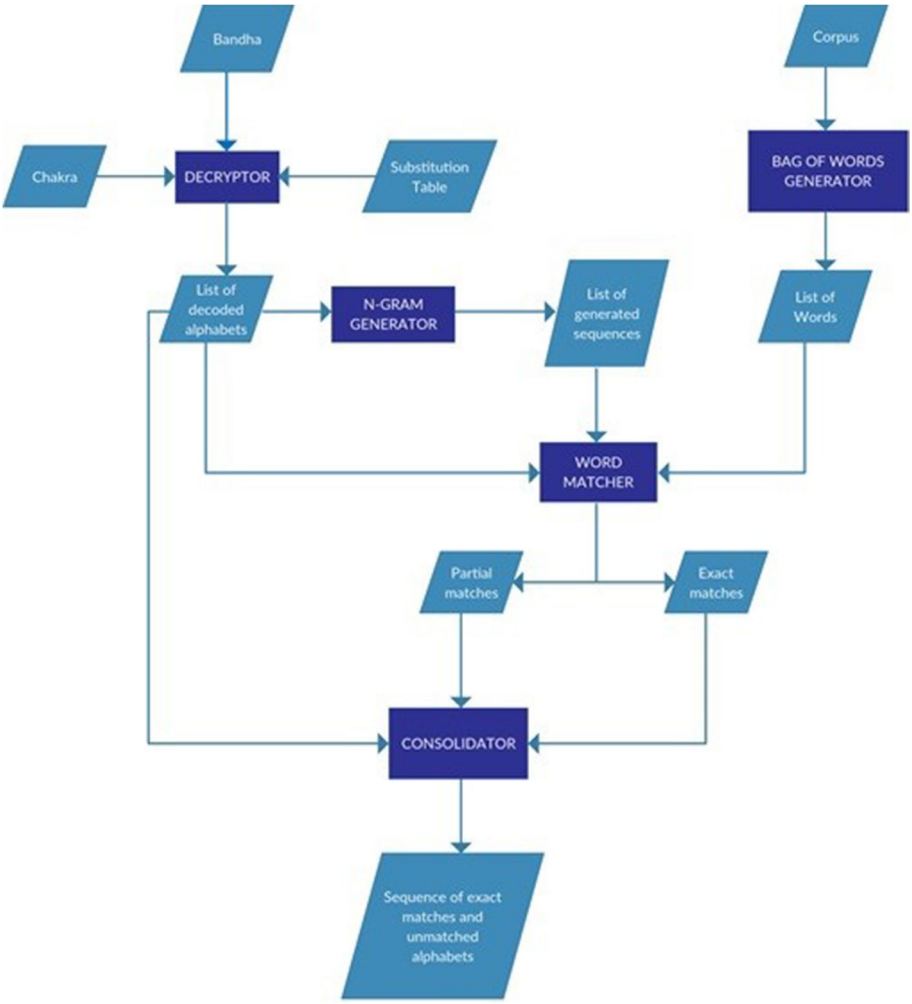

Fig. 7 Model for automated decryption and alphabet association

3.1.1 Chakra

The chakra is a 27×27 matrix of integers in the range of 1 and 64 (as described in Section 1). An instance of chakra is given in Fig. 8.

3.1.2 Bandha

The bandha is a technique to decrypt a chakra using transposition. There are several bandhas to decrypt a chakra (as discussed in Section 2.2). The algorithm for Chakra Bandha is given in Algorithm 3.

3.1.3 Substitution table

The numerals obtained on application of bandha on chakra is replaced with its corresponding alphabets using the substitution table. An instance of this is given in Fig. 1.

In summary, the *decryptor* applies the input *bandha* on the input *chakra* and substitutes the output numerals with the corresponding alphabets using the input substitution table. Thus, resulting in a list of decoded alphabets.

3.2 N-gram generator

It takes as input the list of decoded alphabets rendered by the decryptor and returns a list of 1-g[1] to 25-g sequences of alphabets.

59	23	1	16	1	28	28	1	1	56	59	4	56	1	1	47	16	34	1	7	16	1	1	7	56	1	60
53	54	47	28	1	47	45	28	7	4	59	41	4	45	1	30	47	47	45	42	53	28	51	1	52	1	1
1	22	1	30	2	1	2	55	30	1	7	45	47	52	1	4	1	47	1	1	1	1	53	1	52	59	52
59	30	2	55	55	13	16	2	53	60	1	4	16	47	48	45	16	56	56	43	45	1	56	1	4	1	13
47	45	1	1	22	30	51	1	2	56	38	30	4	1	1	56	1	1	16	1	57	7	56	56	1	22	1
54	52	52	45	1	7	55	48	1	58	52	35	28	55	1	38	45	30	55	4	47	7	45	38	45	38	1
1	1	1	28	13	56	55	51	54	1	1	1	1	42	2	4	4	1	43	16	47	7	1	13	4	51	4
28	53	47	22	8	1	53	59	38	7	43	40	1	52	59	54	30	1	45	16	1	28	23	50	7	43	43
1	2	45	51	30	1	52	58	48	59	47	54	4	4	1	47	45	47	56	28	1	45	1	13	7	7	7
55	1	53	47	56	1	1	7	1	1	2	60	48	56	1	1	16	1	1	54	1	52	17	30	54	45	45
59	56	52	1	45	1	55	28	52	28	1	2	1	52	54	4	43	60	48	28	1	16	23	8	53	7	1
2	1	53	52	43	23	2	4	16	52	44	54	1	2	42	7	1	7	47	30	28	48	47	1	54	52	16
45	54	23	4	28	45	45	30	1	59	1	56	28	2	54	53	38	2	2	1	28	55	40	60	4	50	28
2	13	47	1	1	4	17	45	1	56	1	52	56	51	1	47	55	55	45	7	2	54	1	56	7	1	1
23	4	53	54	59	48	13	56	1	47	23	1	2	55	16	1	1	47	40	54	16	52	1	47	60	43	60
45	16	43	1	7	47	1	7	1	4	54	54	1	43	28	28	7	1	2	7	52	30	1	4	47	4	13
42	1	54	13	1	28	1	45	42	5	48	56	1	1	1	52	54	7	1	1	2	56	56	2	43	1	1
56	43	22	45	56	43	2	2	56	1	8	48	59	59	7	16	53	55	53	48	1	1	46	2	30	53	1
47	45	1	2	54	56	56	2	55	51	4	16	7	13	30	16	1	1	4	52	52	4	54	47	2	38	1
1	54	60	56	54	1	60	1	1	16	40	38	17	1	47	56	33	55	1	1	59	48	1	53	7	1	1
1	52	16	1	60	1	30	53	30	7	47	13	13	22	8	13	45	59	54	1	2	42	54	47	53	52	53
16	30	1	4	52	47	56	1	28	16	1	22	59	51	1	1	7	28	53	60	7	1	16	16	1	1	58
4	53	56	1	52	2	13	52	38	30	45	7	1	30	56	16	1	1	1	30	48	56	54	54	55	28	45
1	47	47	1	28	22	1	47	1	1	45	46	1	1	47	53	55	52	1	1	7	43	2	1	1	1	43
1	4	53	1	45	43	16	55	52	4	47	55	45	22	51	56	1	38	13	30	2	28	56	13	56	28	55
4	16	46	1	1	16	1	1	1	1	1	47	59	4	8	38	58	1	1	48	1	7	22	1	1	1	60
52	4	30	56	53	52	54	1	30	52	1	16	54	7	58	1	30	54	1	56	51	53	56	57	56	4	60

Fig. 8 Chakra1-1–1

3.3 Bag of words generator

This component takes a corpus containing documents relating to a particular language (language used by substitution table) and returns a dictionary for each word in the corpus with its corresponding frequency of occurrence.

3.4 Word matcher

This takes the following inputs:

1) List of generated sequences from the N-gram generator
2) List of words from the bag of words generator
3) List of decoded alphabets from the decryptor

These inputs are passed as parameters to the following procedures:

3.4.1 Finding partial matching words for alphabet sequences. Algorithm 1 requires the predefined function

(1) str_search_list(sequence, word_list) — Takes a string, sequence as a regular expression. Searches for this regular expression in the list of strings, word_list. Returns a list of strings in word_list that match sequence. If no matches are found, returns ϕ.

Algorithm 1 Find partial matches for alphabet sequences

Procedure FindPartialMatch(*sequence_list,word_list*):

partial_matches ← {ϕ };

for *sequence in sequence_list* do

partial_matches[sequence] ← [ϕ];

results ← *str_search_list*(sequence,word_list);

if *results* != ϕ then

partial_matches[sequences] ← results;

end

end

return *partial_matches*

3.4.2 Finding exact matching words for alphabet list. Algorithm 2 necessitates the following predefined functions

(1) as.string(char_list) — Returns a string formed from the combination of all the characters in. char_list
(2) myarray.append(myelement) — Appends *myelement* to the array, *myarray*.
(3) str_search_str(substr,str) — Returns True if string, substr is a substring of string, str. Else, returns False.

Algorithm 2 Find exact matches for alphabet list

Procedure FindExactMatch(*alphabet_list,word_list*):

exact_matches ← [ϕ];

all_string ← *as.string*(alphabet_list);

for *word in word_list* do

results ← *str_search_str*(word,all_string);

if *results* == *True* then

exact_matches.*append*(results);

end

end

return *exact_matches*

The component returns a dictionary of partial matches and a list of exact matches.

3.5 Consolidator

This is the final component of the model that takes as inputs:

1) List of decoded alphabets from the decryptor
2) Dictionary of partial matches, and
3) List of exact matches

Returns a sequence of exact matches that substitute the corresponding alphabets and unmatched alphabets.

4 Results

The model proposed in this paper has been implemented in R programming language [8, 10] on a 128 GB RAM, 64-bit Linux system running R version 3.3.1. This implementation resulted in the generation of text files for the final and intermediate output. To eliminate the need for si- multaneously viewing these text files, an interactive Shiny [3, 17] web application was developed and deployed. This application is hosted on shinyapps.io [4, 6] and can be accessed at the address: https://siribhoovalya.shinyapps.io/siribhoovalya/.

The functionality of the application is explained as follows:

1) A *chakra* for decryption must be selected in the *Input Chakra* drop down. The selected chakra is displayed on the *Input* tab.
2) The language is chosen in the *Substitution Table* drop down.
3) The desired *bandha* is selected in the *Decryption Algorithm* drop down.
4) Once the chakra, language and bandha are appropriately chosen, the *Process* button must be pressed.
5) This will result in a *Processing* pop-up window to be visible in the bottom-left. This pop-up window will reflect the progress of the processing. The processing can be halted at any time by selecting the close button in the pop-up window.
6) Completion of the processing will display the *Output* tab that contains the following:
    a) Decrypted Output pane that shows the list of decoded alphabets (as rendered by the Decryptor component of the model)
    b) Exact Matched Predicted words pane that depicts the exact matches (as presented by the
    c) Word Matcher component of the model)
    d) Unigram Predicted words pane that portrays the partial matches (as provided by the Word Matcher component of the model)
    e) Processed Output pane that provides the sequence of exact matches and unmatched alphabets (as given by the Consolidator component of the model)

Figure 9 shows the web application processing the given inputs, while Fig. 10 shows the output pane of the web application.

5 Conclusion

This paper presents a comprehensive model which when given a *chakra*, a *bandha* and a *substitution table* will not only return a list of decrypted alphabets but is also capable of returning words predicted from these alphabets. The model also provides the words that partially match the alphabets.

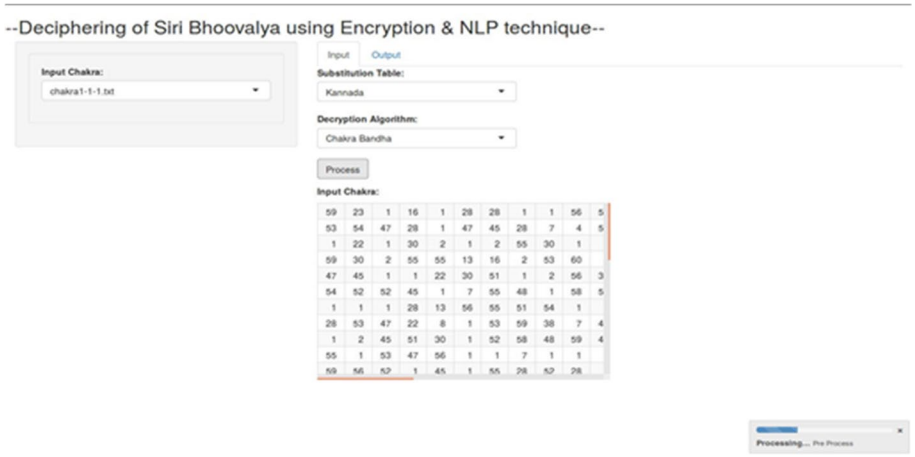

Fig. 9 Web application processing the inputs

Accounting for the fact that *Siri Bhoovalaya* is a work that encapsulates works from several fields of study by applying numerous encryption techniques to create a seemingly simple set of 729 numbers per page, the authors share the view of several contemporaries [2, 11, 12, 18] in believing that researchers of no particular area of research will be able to solely unravel the mysteries of this intriguing creation.

This paper makes an initial attempt towards associating alphabets to form words and portrays both the partially matched words and the exactly matched words with the intention of providing a common base for linguists, cryptographers, religious experts, etc.to work towards solving the intricacies of *Siri Bhoovalaya*.

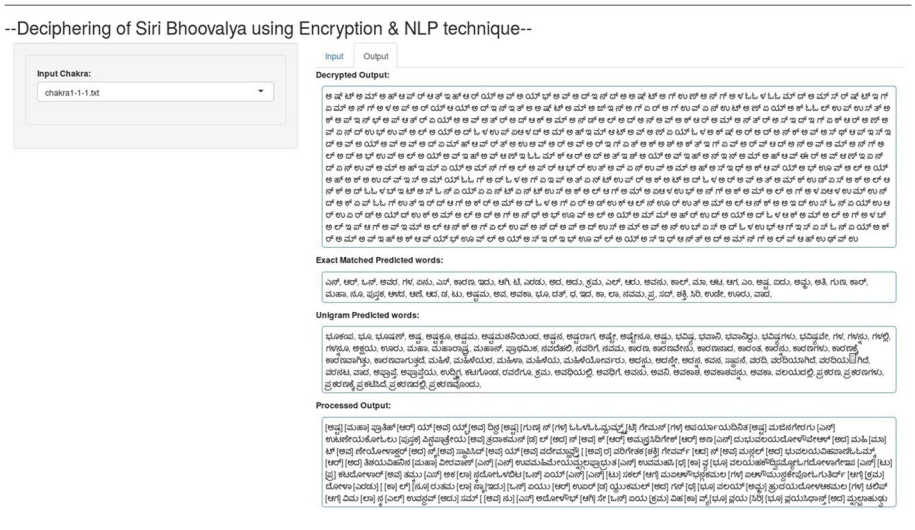

Fig. 10 Output pane of web application

6 Future work

Two primary challenges stand out in the study at hand. First, while the paper outlines two transposition techniques in Section 2.2, there's an understanding that Muni Kumudendu utilized several other, perhaps lesser-known, transposition methods as mentioned in Section 1. Comprehensive research is required to pinpoint these techniques and develop a bespoke approach for their decryption concerning this work.

Second, the challenge of deciphering ancient word associations in Siri Bhoovalaya is intensified by its age. Given that it was composed roughly a millennium ago, it likely contains archaic terms that have since fallen into obscurity. Compounding this challenge is the fact that many potentially helpful reference materials, which might contain these obsolete words, haven't been digitized. Considering that the chakras decode into 718 dialects, some with antiquated terms and texts not readily available digitally, creating a pertinent corpus appears to be a monumental task. Collaborative efforts among experts spanning various disciplines—from linguistics to computer science—are imperative [19, 20].

The introduced web application represents a pioneering attempt to simplify the deciphering process, obviating the need for intricate hardware and software. However, it has its limitations in terms of the number of chakras, the substitution table, and the decryption algorithms it currently supports [21, 22]. Efforts are in progress to digitize the available chakras. Additionally, substitution tables for numerous primary languages are in development and will soon be accessible. Notwithstanding, integrating decryption algorithms is intricate and demands a deeper, more nuanced understanding [23–25].

Appendix 1

Chakra Bandha Transposition algorithm

Algorithm 3 requires the following predefined functions:

1) myarray. append(myelement) — Appends myelement to the array, myarray.
2) mynumber + + — Increments the integer or float, mynumber by one.
3) mynumber1: mynumber2 — Returns array of all integers between the two integers, mynumber1 and mynumber2. If the mynumber1 and mynumber2 are floats, then array of all floats between these two floats will be returned.
4) rev(myarray) — Reverses the array, myarray.
5) len(myarray) — Returns the number of elements in the array, myarray.

Algorithm 3 Chakra Bandha transposition algorithm

Procedure ChakraBandha(*chakra*):

← n ←[ ] 1; out ϕ ;

for

col 14 to 25 do out.append(chakra[1,col]);

out.append(GetFirstLower(chakra,col));

out.append(GetFirstUpper(chakra,col,n));

out.append(GetJump(chakra,col,n));

n++;

end

out.*append*(chakra[1,26]);out.*append*(chakra[27,27]);out.*append*(chakra[26,1]);

out.*append*(GetSecDiag(chakra)); out.*append*(chakra[2,27]);

← n 1;

while *n* < 13 do out.*append*(GetSwap(chakra,n));

out.*append*(GetSecondLower(chakra,n));

out.*append*(GetSecondUpper(chakra,n)); n++;

end

out.*append*(GetSwap(chakra,n));

out.*append*(GetSecondLower(chakra,n));

return out

Algorithm 4 Chakra Bandha transposition algorithm (continued)

Function GetFirstLower(*chakra,col*)**:**

← [ ]

out ϕ ;

←

rows 27:(col+1);

←

cols *rev*(rows);

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Function GetFirstUpper(*chakra,col,n*)**:**

← [ ]

out ϕ ;

←

←

rows col:(2*n); cols 1:(col-(2*n-1));

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Function GetJump(*chakra,col,n*)**:**

← [ ]

out ϕ ;

←

←

rows (2*n+1):2; cols (col-2*n-1):col;

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Function GetSecDiag(*chakra*)**:**

← [ ]

out ϕ ;

←

Algorithm 4 (continued)

rows 27:1;

←

cols *rev*(rows);

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Function GetSwap(*chakra,n*)**:**

← [ ]

out ϕ ;

←

rows n:1;

←

cols *rev*(rows);

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Algorithm 5 Chakra Bandha transposition algorithm (continued)

Function GetSecondLower(*chakra*)**:**

← [ ]

out ϕ ;

←

←

rows 27:(2*n+1); cols (n+1):(27-n);

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Function Get SecondUpper(*chakra*)**:**

← [ ]

out ϕ ;

←

rows (2*n+2):(n+2);

←

cols (27-n):27;

←

for *i* 1 *to* len*(rows)* **do**

out.*append*(chakra[rows[i],cols[i]]);

end return *out*

Funding On Behalf of all authors the corresponding author states that they did not receive any funds for this project.

Data Availability All the data is collected from the simulation reports of the software and tools used by the authors. Authors are working on implementing the same using real world data with appropriate permissions.

Declarations

Conflict of Interest The authors declare that we have no conflict of interest.

References

1. Brown PF, Desouza PV, Mercer RL, Della Pietra VJ, Lai JC (1992) Class-based n-gram models of natural language. Comput Linguist 18(4):467–479
2. Jain AK (2013) An inimitable cryptographic creation: Siri Bhoovalaya
3. Shiny (2017) Shiny. https://shiny.rstudio.com/
4. shinyapps.io (2017) shinyapps.io. https://www.shinyapps.io/
5. Stallings W (2006) Cryptography and network security: principles and practices, 4th edn. Pearson Education India, pp 35–49
6. Stinson DR (2005) Cryptography: theory and practice. Chapman and Hall/CRC
7. Stallings W (2007) Network security essentials: applications and standards. Pearson Education India
8. R Core Team (2016) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna. https://www.R-project.org/
9. You Y et al (2018) A review of cyber security controls from an ICS perspective. In: 2018 international conference on platform technology and service (PlatCon). IEEE
10. Zhang H, Lin Y, Xiao J (2017) An innovative analying method for the scale of distribution system security region. In: 2017 IEEE power & energy society general meeting, IEEE
11. Bianchi T, Bioglio V, Magli E (2014) On the security of random linear measurements. In: 2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE
12. Do T, Gan L, Nguyen N, Tran T (2012) Fast and efficient compressive sensing using structurally random matrices. IEEE Trans Signal Process 60(1):139–154
13. Rao A, Jha B, Kini G (2013) Effect of grammar on security of long passwords. In: Proceedings of the third ACM conference on data and application security and privacy (CODASPY '13). ACM, New York, pp 317–324. https://doi.org/10.1145/2435349.2435395
14. Yan Y, Huang J (2017) Cooperative output regulation of discrete-time linear time-delay multi-agent systems under switching network [J]. Neurocomputing 241(7):108–114
15. Zhou L, Li C (2017) Out sourcing Eigen-decomposition and singular value decomposition of large matrix to a public cloud. IEEE Access 4:869–879
16. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed) Advances in cryptology—Eurocrypt. Springer, Berlin, pp 223–238
17. Jha DP, Kohli R, Gupta A (2016) Proposed encryption algorithm for data security using matrix properties. In: 2016 International conference on innovation and challenges in cyber security (ICICCS-INBUSH). IEEE
18. Patel B, Desai P, Panchal U (2017) Methods of recommender system: A review. In: 2017 international conference on innovations in information, embedded and communication systems (ICIIECS). IEEE
19. Thomas A, Sujatha AK (2016) Comparative study of recommender systems. In: 2016 international conference on circuit, power and computing technologies (ICCPCT). https://doi.org/10.1109/iccpct.2016.7530304
20. Yang S, Chen B (2023) SNIB: improving spike-based machine learning using nonlinear information bottleneck. IEEE Trans Syst Man Cybern: Syst 53(12):7852–7863. https://doi.org/10.1109/TSMC.2023.3300318
21. Sharma RK (2018) Title of the article. J Indian History Culture 2:11–35
22. Kumar SP, Sethi R eds (2021) Krishna Sobti: A counter archive. Taylor & Francis
23. University of Kerala (2000) International journal of Dravidian linguistics, vol 29. Department of Linguistics, University of Kerala

24. Hong Z et al (2021) Challenges and advances in information extraction from scientific literature: a review. JOM 73(11):3383–3400
25. Khurana D, Koli A, Khatter K et al (2023) Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl 82:3713–3744. https://doi.org/10.1007/s11042-022-13428-46