Pilot Study and Proof of Concept: Multilabel Kolam Dataset for applying Machine Learning Models

1. Introduction

Kolam generation using machine learning and deep learning first require a dataset. There is no dataset for Kolams as this is a novel concept. There are several ways to create a dataset for applying algorithms but there is no guarantee that those ways can be used for a meaningful Kolam sequence generation. A pilot study and proof of concept is necessary to know the feasibility of the method used in creating the dataset before using it on a large scale, otherwise the models will give completely wrong predictions. One way which can give meaningful Kolam sequences has been found to apply the algorithms. A pilot study and a proof of concept has been conducted to know its feasibility, as when a large dataset is used for complex algorithms for best generation, it costs a lot of time, and the process may fail after investing so much of time. In our Kolam simulation work, Kolams are encoded into sequences using an encoding method with symbols a, b, c and d. As we obtained sequences, we can use them in creating dataset for applying machine learning to predict a Kolam sequence. A Multilabel dataset has been created with 160 data points for this pilot study and proof of concept to know whether this method of dataset creation is useful for applying machine learning algorithms to provide a meaningful Kolam sequence generation.

2. Methodology

2.1 Explanation on how dataset is created and how it can be augmented using different Kolam sequences

The inputs sequences are treated like a text and words and so are given spaces between the shapes so that we can apply text classification and document classification models. The multilabel is considered in this dataset, as for an input sequence, there is a possibility of occurrence of different symbols (it can be 'a' or 'b' or 'c' or 'd'). Then it can be treated like a text classification and document classification with multiple labels (say a document belongs to fiction and comedy and horror and it will have 1's at these labels).

The Kolam sequence generation problem needs some more steps to be solved. Here a Kolam sequence has 1's for different labels but with OR relationship not AND relationship like we have in document classification with multiple labels.

Example 1:

input	a	b	c	d
dcab b b	1	1	0	1

Here 'dcab b b' is a small input sequence and the next possible symbols are 'a or b or d' but not 'a and b and d'. The space is used between 'dcab', 'b' and 'b' to make the shapes look like words (say 'dcab' is a word, 'b' is a word and 'b' again is a word). We need another step to determine what to select from the possible outcomes (a, b, d). Then after determination, we can add that symbol to the sequence ‘dcab b b’ which now becomes ‘dcab b b [predicted and determined symbol]’. If the next symbol is varying, say a-->b, then space should not be given as this shape is growing to become another shape now. If the succeeding symbols are also varying, say a-->b-->d-->c, no space should be given as this shape is growing to become another shape now. If the succeeding symbols are the same, say c-->c-->c, then space should be given as these are different shape-I’s. If the succeeding symbols predicted are like a-->c-->c, space is given after ‘ac’ as it is a shape-II and after that we got shape-I. It becomes ‘ac c’. A small logic is to be written to achieve this space adding task. Such an updated sequence is now again fed to the model to predict the next symbol and this process can be iterated until it gives our prescribed length of sequence (say for example, we need a 10-element string including space as an element).

Only some shapes are possible after some shapes. For example, ‘bdca acd’ will next become either ‘bdca acdb’ or ‘bdca acd d’. When adding more data points based on Kolam sequences, the Kolam rules must also be kept in mind. Otherwise, it will lead to incorrect data which will further lead to incorrect predictions.

Shape modification probabilities:

1. Shape-I can be modified to any other shape. e.g. ‘a’ can be modified to ‘ab’ upon prediction of ‘b’ after ‘a’.
2. Shape-II can be modified to shape-III and shape-IV. e.g. ‘ab’ can be modified to ‘abd’ upon prediction of ‘d’ after ‘ab’.
3. Shape-III can be modified to shape-IV. e.g. ‘abd’ can be modified to ‘abdc’ upon prediction of ‘c’ after ‘abd’.
4. Shape-IV cannot be modified. E.g. ‘dbac’ cannot be modified to any other shape as there are no more shapes after shape-IV in this work.

Care should be taken while giving the predicted sequence as input to the model for further prediction. If ‘c’ is predicted after ‘abd’, then the input for the next prediction should be ‘abdc’ but not ‘abd c’, as ‘c’ cannot come after ‘abd’. Some more code will be required to differentiate them before giving it to the model for further prediction. Otherwise, it leads to incorrect data and incorrect predictions. A logic is found from the data. If there is a change in succeeding element (predicted symbol and last symbol of the input sequence are different), no space should be given and if there is no change in succeeding element (predicted symbol and last symbol of the input sequence are the same), a space should be given between them. For example, if we get ‘b’ upon prediction for an input sequence ‘dcab’, as both are ‘b’, space should be given, and the output sequence will be ‘dcab b’. If we get ‘b’ upon prediction for an input sequence ‘dca’, as ‘b’ and ‘a’ are different, no space should be given. Like this the code should look at the sequence and maintain valid spaces before giving it to the model for further predictions.

Shape following probabilities:

Every shape can be followed by only particular shapes and their orientations.

	Only possibilities

a	**a, b, c, (a, b), (a, c), (a, c, d), (a, b, d), (a, b, d, c), (a, c, d, b)**
b	**b, a, d, (b, a), (b, d), (b, d, c), (b, a, c), (b, a, c, d), (b, d, c, a)**
c	**c, a, d, (c, a), (c, d), (c, a, b), (c, d, b), (c, d, b, a), (c, a, b, d)**
d	**d, b, c, (d, c), (d, b), (d, b, a), (d, c, a), (d, c, a, b), (d, b, a, c)**
(a, b)	**b, (b, a), (b, a, c), (b, a, c, d)**
(b, a)	**a, (a, b), (a, b, d), (a, b, d, c)**
(c, d)	**d, (d, c), (d, c, a), (d, c, a, b)**
(d, c)	**c, (c, d), (c, d, b), (c, d, b, a)**
(a, c)	**c, (c, a), (c, a, b), (c, a, b, d)**
(c, a)	**a, (a, c), (a, c, d), (a, c, d, b)**
(b, d)	**d, (d, b), (d, b, a), (d, b, a, c)**
(d, b)	**b, (b, d), (b, d, c), (b, d, c, a)**
(b, a, c)	**c, (c, a, b)**
(a, c, d)	**d, (d, c, a)**
(d, b, a)	**a, (a, b, d)**
(c, d, b)	**b, (b, d, c)**
(b, d, c)	**c, (c, d, b)**
(a, b, d)	**d, (d, b, a)**
(d, c, a)	**a, (a, c, d)**
(c, a, b)	**b, (b, a, c)**
(b, d, c, a)	**a, (a, c), (a, b), (a, c, d)**
(a, b, d, c)	**c, (c, d), (c, a), (c, d, b)**
(d, c, a, b)	**b, (b, a), (b, d), (b, a, c)**
(c, a, b, d)	**d, (d, b), (d, c), (d, b, a)**
(a, c, d, b)	**b, (b, d), (b, a), (b, d, c)**
(b, a, c, d)	**d, (d, c), (d, b), (d, c, a)**
(c, d, b, a)	**a, (a, b), (a, c), (a, b, d)**
(d, b, a, c)	**c, (c, a), (c, d), (c, a, b)**

Using the above rules and methods, a multilabel dataset has been created with 160 data points in .csv format for the pilot study. The format of the multilabel dataset is as given below.

Input column – It contains parts of Kolam sequences with shape-I, shape- II, shape- III and shape- IV with different orientations and repetitions.

Labels column – The label columns contain four labels, a, b, c, and d, which are the symbols present in a Kolam sequence.

2.2 Feature Extraction Methods:

As the input is considered as a text, we need feature extraction methods to transform the words in our dataset into vectors to apply machine learning algorithms. Three popular methods are considered for this pilot study. They are

- TF-IDF
- CountVectorizer
- Word2Vec

2.3 Machine Learning Algorithms:

For the three feature extraction methods, popular multilabel machine learning algorithms are used individually. They are

- Logistic Regression
- Support Vector Machines
- K-Nearest Neighbors
- Gradient Boosting Machines
- Random Forest Classifier

2.3 Feeding the predicted symbols to the model for a required length sequence:

Machine learning algorithms help us predict the labels for the given input sequence. But as the given dataset is multilabel in nature with OR relation, we need to select one symbol out of the predicted labels. We can utilize the maximum value of the probability from the positive class to select a label as it is more likely to happen compared to the other labels even though they are positive upon prediction. The selected label should be now attached to the input sequence. As explained in 2.1, it is observed that if the last symbol of the input sequence is the same as the predicted symbol, then space should be given while appending it to the input sequence and feeding it back to the model for further predictions. If it is different, then no space should be given while appending it to the input sequence and feeding it back to the model for further predictions. The number of times we want to append the symbols depends on the required length of the input sequence including spaces. The process is iterated until we reach the required length of the sequence.

3. Results

The accuracy scores of different machine algorithms for different feature extraction methods are given in the below Table.

ML Algorithms Accuracy (in %)	**Feature Extraction Methods**		
	TF-IDF	CountVectorizer	Word2Vec
Logistic Regression	40.63	46.88	NA
Support Vector Machines	34.38	46.88	46.88
K-Nearest Neighbors	31.25	34.38	21.88
Gradient Boosting Machines	**56.25**	37.5	40.63
Random Forest Classifier	53.13	37.5	43.75

Out of the applied combinations, Gradient Boosting Machines with TF-IDF gave better accuracy of 56.25%. For the input_sequence = ['dbac c'] and max_length = 10, we got the output as ['dbac c cd d'], which is a meaningful Kolam sequence part where ‘dbac’ is correctly followed by a prediction ‘c’ and correctly given space as it is a repeating symbol. ‘dbac c’ is then followed by a prediction ‘c’ which is again correctly given space as it is a repeating symbol. ‘dbac c c’ is then followed by a prediction ‘d’ which is a valid shape ‘cd’ which can occur after ‘c’ and the space is not given as it is not a repeating symbol.

Some partial sequences with certain length are given in the below Table.

Initial input sequence	Output with max_length= 10 and validity
[dbac c]	[dbac c cd d] - valid
[bdc]	[bdc ca ab b] - valid
[ab b]	[ab b b bd d] - valid
[c cd]	[c cd d dc c] - valid
[cabd]	[cabdb ba a] - invalid
[cabd d]	[cabd db ba] - valid
[ca]	[cab b ba a] - valid

4. Discussion

The dataset is showing decent accuracy when machine learning algorithms are applied for Kolam sequence generation, as we can see from the results, we obtained decent accuracy percentages and a meaningful sequence generation, even though the dataset is small. Moreover, the dataset is unbalanced where the count of the labels is 63 and 61 for 2 labels and 3 labels respectively, but 36 for 1 label.

5. Conclusion

A multilabel dataset has been created by taking the rules of the Kolam drawing process. The probability of occurrences of shapes and symbols are also taken into consideration. From the results and outputs of this pilot study and proof of concept, the method used for dataset creation can be considered useful for developing it further for applying machine learning algorithms for Kolam sequence generation.

6. Recommendations

To improve the quality of Kolam sequence generation in terms of length and validity, one or more of the following steps should be considered for further research.

- Increasing the dataset size
- Balancing the labels
- Applying other feature extraction methods
- Applying DL algorithms on the increased dataset

The multilabel dataset can be replaced with multiclass dataset, where the target now can be shapes of the Kolam. For example, the input shape ‘dbac’ will have a target shape ‘c’. But with this approach, the complexity of the prediction process increases as the total number of classes will be 28 with 28 Kolam shapes.