A Romanisation Method for the Bengali Language with Efficient Encoding Scheme

Paper ID: 263

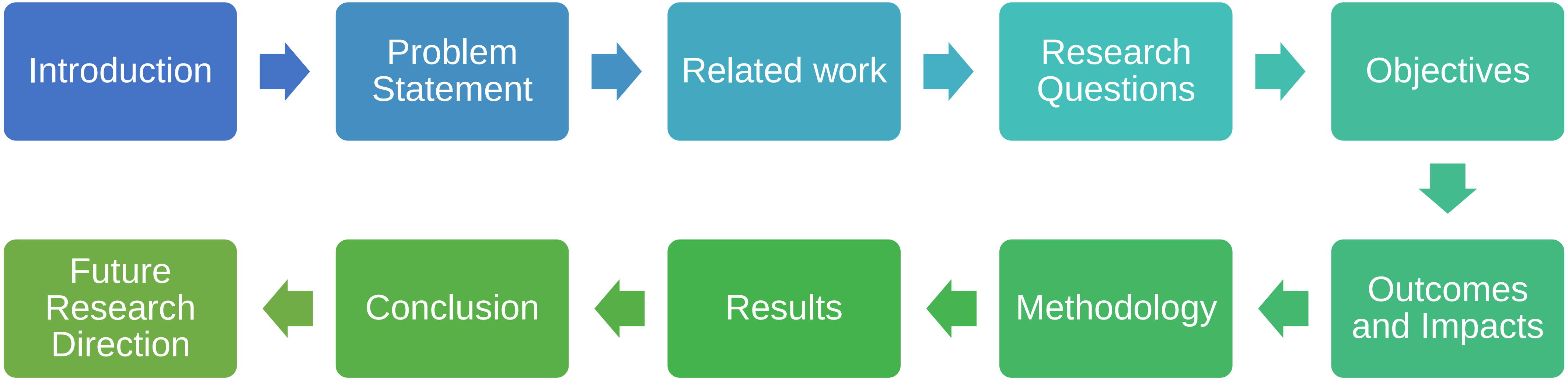
Date: 06 September, 2023

Amrita Das Tipu Md. Fahad Ashis Kumar Mandal

Department of CSE
Hajee Mohammad Danesh Science & Technology University, Dinajpur-5200

Authors Information

Outline


Introduction
Problem Statement
Related work
Research Questions
Objectives
Outcomes and Impacts
Methodology
Results
Conclusion
Future Research Direction

Introduction

Transliteration

- Converts the written form of a language
- From one language to another
- Retains phonetic meaning

Table 1: Bengali to English transliteration

Original	**Transliterated**
শিক্ষাবিদ	shikkhabid
বাংলা	bangla
ভাষা	bhasha
আমার	amar

Problem Statement

- Lack of standardization for Bengali language
- Limited publicly available datasets
- Variations in transliteration
- Resource in traditional ML method

Related Work

Table 2: Literature review

AUTHOR	CONTRIBUTION	LIMITATION	DIFFERENCE
Sarkar and Chatterjee [1]	One-hot coding for representing TU, used traditional ML model SVM and KNN	Evaluated on only 1000 NEs, private dataset	Our study uses binary coding in place of one hot coding for the TUs for cost reduction
Ekbal et al. [2]	6 n-gram based probabilistic models	Only used 6000 NEs, private dataset	Only 1 n-gram based model is used, evaluated on common words, provides algorithm for TU identification
Rathod et al. [3]	n-gram based feature selection, used SVM, for Hindi and Marathi languages	Only used NEs, private dataset	SVM and RFC is used, Bengali language, public datasets
Dasgupta et al. [4]	Joint source channel model, SMT model.	Backward transliteration approach	Forward transliteration or romanisation, TU identification algorithm

Research Questions

How to optimize computation resource usage in traditional machine learning models?

How to increase transliteration accuracy?

Objectives

- Optimize computational time
- Optimize memory usage
- Comparative or higher performance

Outcomes and Impacts

Expected outcome:

- Correct prediction with lower cost

Possible impacts:

- Automatic, reliable system
- Unhindered cross-lingual communication

Methodology

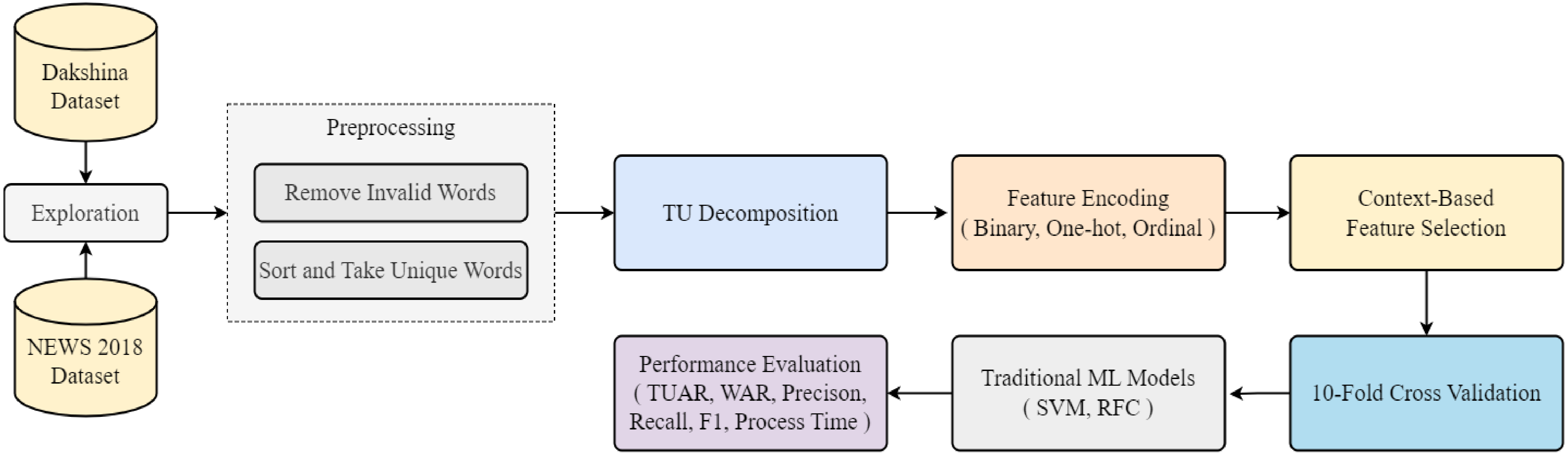


Figure 1: Proposed Methodology

Methodology

- Types of words in dataset
  - Dakshina: Dictionary, NE, Technical
  - NEWS 2018: NE
- Invalid words: Words containing null/empty, punctuations, numbers, out-of-script characters
- Sort and take unique Bengali words

Table 3: Dataset Statistics

Criteria	Dakshina	NEWS 2018
Total Words	130378	13623
Valid Words	113760	13514
Unique Words (Bengali)	25330	13214

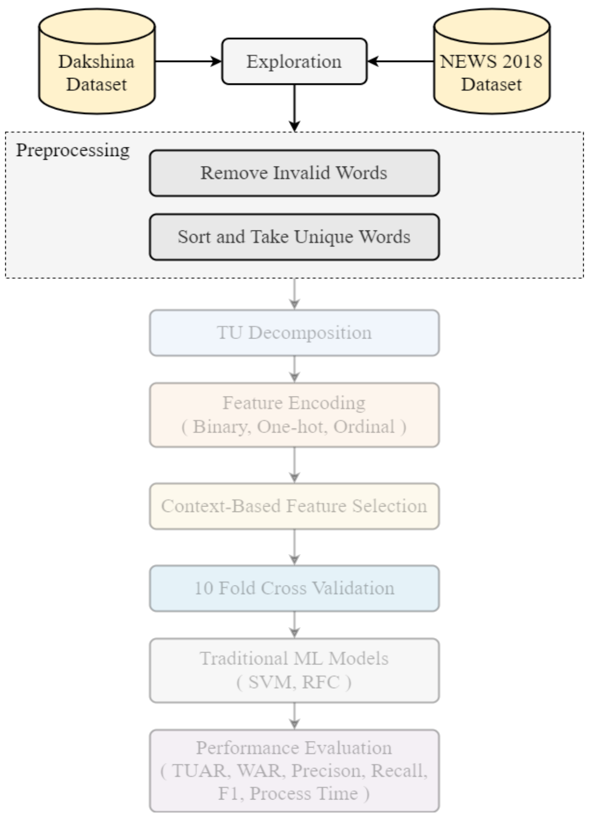


Figure 1 (a): Dataset Exploration and Preprocessing

Methodology

- Transliteration Unit (TU) decompose:
  - আমার → [আ | মা | র]
  - amar → [a | ma | r]

Table 4: TU Statistics

Criteria	Dakshina	NEWS 2018
TU aligned Words	15581	7563
Average number of TU per word	3.463	3.247
Maximum number of TU per word	9	8
Minimum number of TU per word	1	1
Unique TU (Bengali)	1449	947
Unique TU (English)	1605	1166

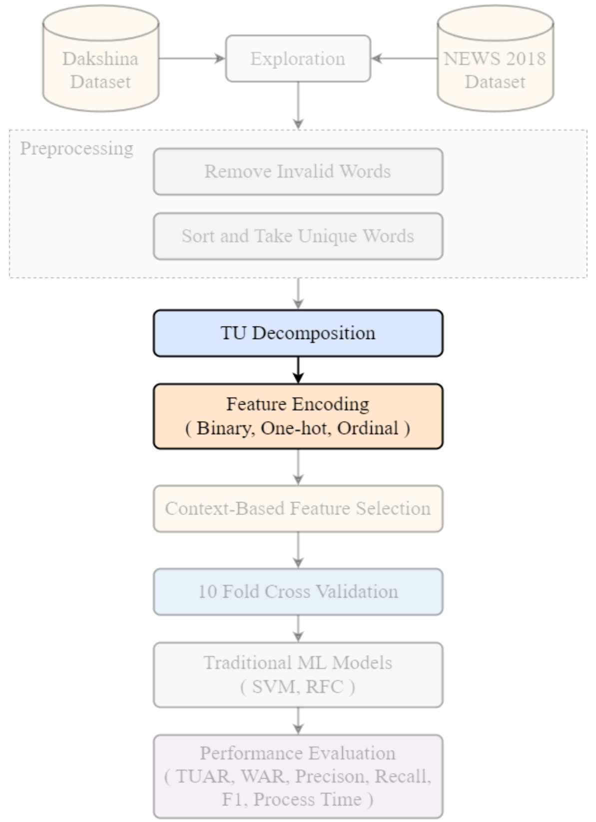


Figure 1 (b): TU decomposition and Feature encoding

Methodology

- Encode: converts string to numerical form

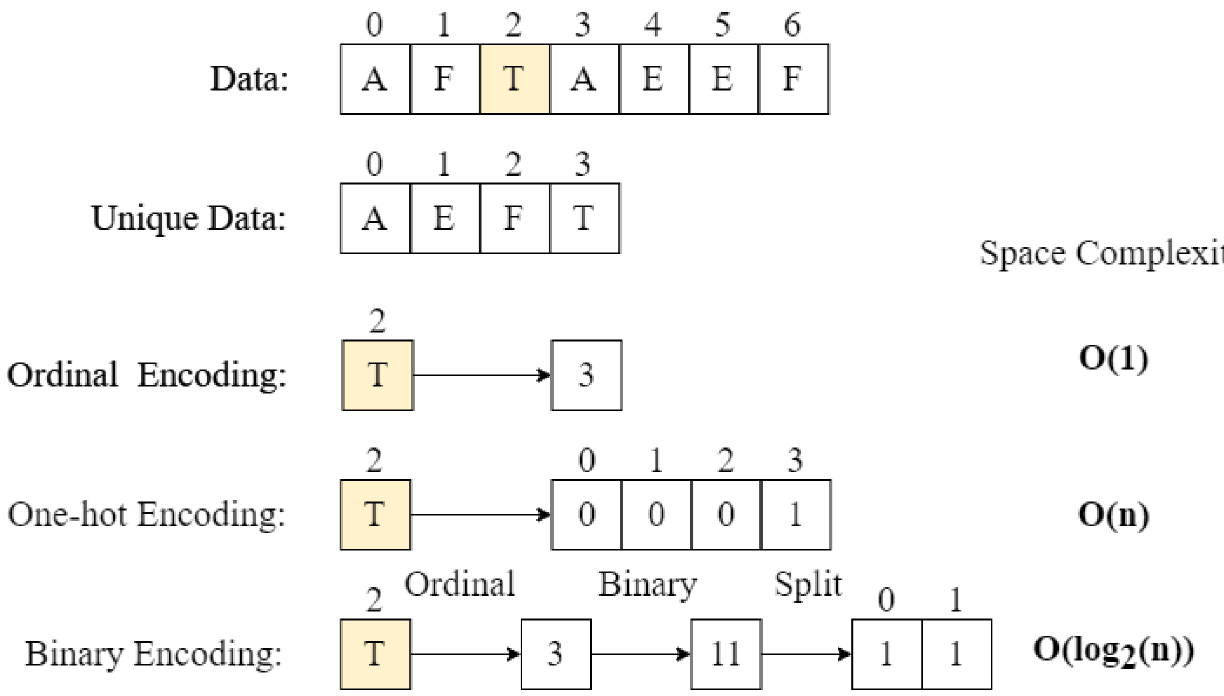


Figure 2: Encoding

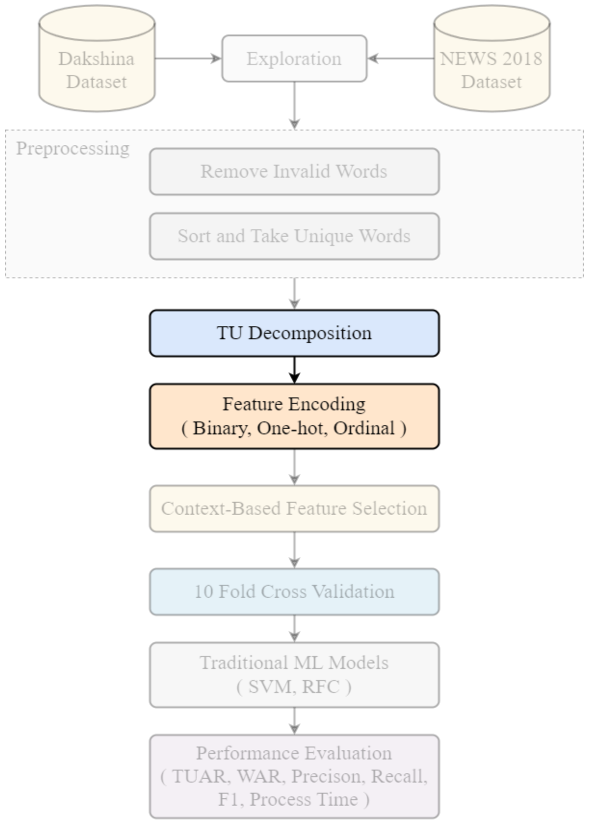


Figure 1 (b): TU decomposition and Feature encoding

Methodology

- Context: preceding or succeeding TU
- Feature selection
  - Preceding TU
  - Current TU
  - Succeeding TU

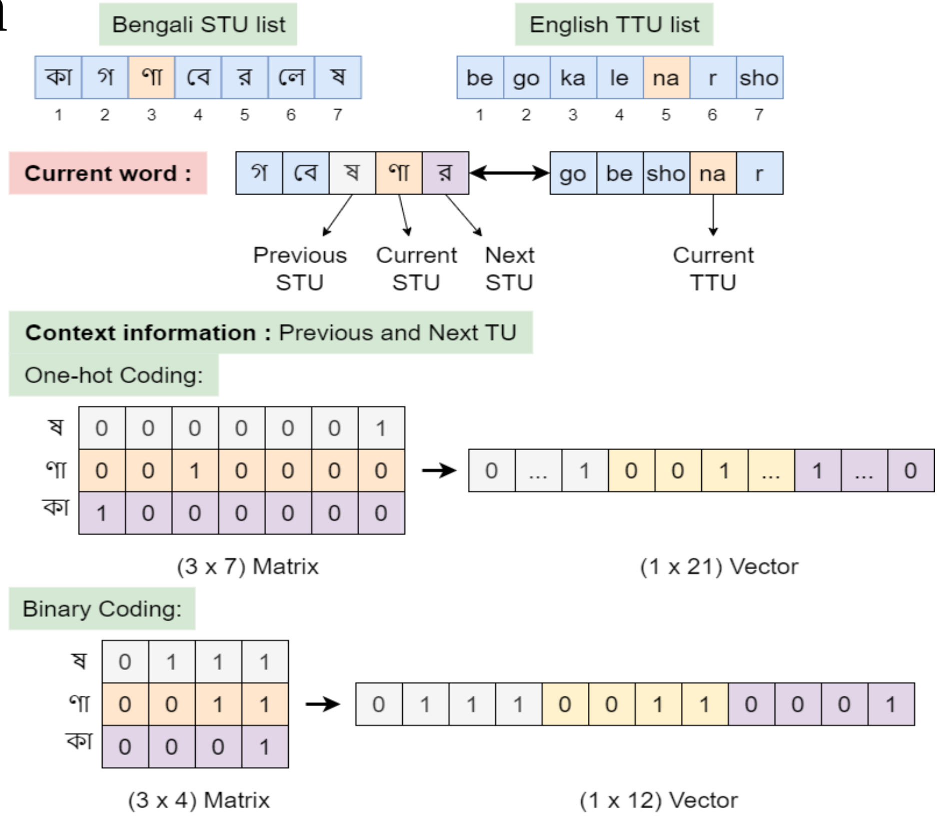


Figure 3: Context-based Feature Selection

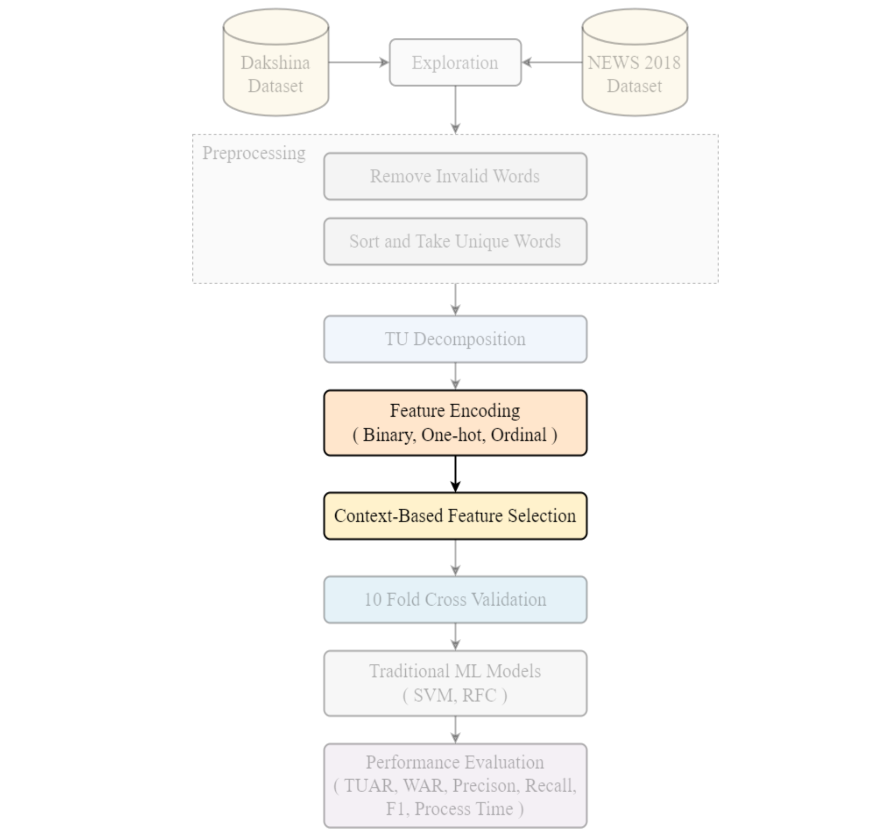


Figure 1 (c): Feature encoding and Feature selection

Methodology

Table 5: ML Models

ML Algorithms	Parameters
SVM	kernel = 'poly', degree = 2, decision_shape_function = 'ovr'
RFC	n_estimators = 200

- **Accuracy**
  - TU level accuracy (TUAR)
  - Word level accuracy (WAR / Top-1)
- **Precision**
- **Recall**
- **F1 score**
- **Process Time**
  - Ratio of one-hot and binary

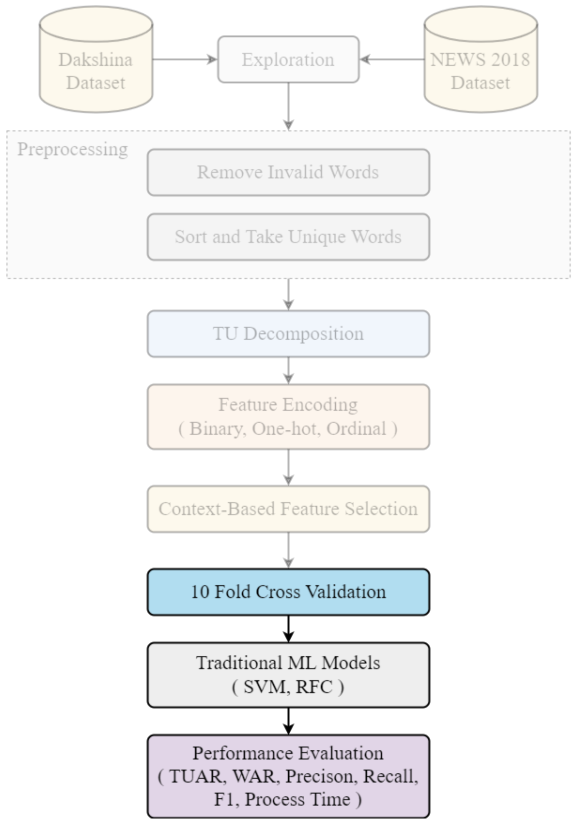


Figure 1 (d): Model preparation

Results

- Baseline: One-hot coding approach
- For Dakshina Dataset

Table 6: Accuracy analysis on Dakshina dataset

Model	SVM		RFC	
Encoding	One-hot	Binary	One-hot	Binary
TUAR (%)	79.5996	**80.1939**	78.2583	**80.7770**
WAR (%)	49.0020	**49.9840**	46.2166	**51.0109**

Table 7: Time analysis on Dakshina dataset

Model	Encoding	Train		Test	
		Time (sec)	Ratio	Time (sec)	Ratio
SVM	One-hot	34974.3703	**340.2229**	2159.3266	4.4014
	Binary	**102.7984**		**490.5969**	
RFC	One-hot	2611.1594	10.2847	39.3453	1.6193
	Binary	**253.8891**		**24.2984**	

Results

- Baseline: One-hot coding approach
- For NEWS 2018 dataset

Table 8: Accuracy analysis on NEWS 2018 dataset

Model	SVM		RFC	
Encoding	**One-hot**	**Binary**	**One-hot**	**Binary**
TUAR (%)	78.4124	**79.2669**	78.6131	**80.5596**
WAR (%)	49.3456	**50.2046**	49.6626	**52.3730**

Table 9: Time analysis on NEWS 2018 dataset

Model	Encoding	Train		Test	
		Time (sec)	**Ratio**	**Time (sec)**	**Ratio**
SVM	One-hot	2423.8516	**88.8825**	129.5156	1.5686
	Binary	**27.2703**		**82.5688**	
RFC	One-hot	498.4031	8.1559	5.5922	1.2038
	Binary	**61.1094**		**4.6453**	

Conclusion

- Grapheme-based approach
- Binary encoding technique
- Reduces memory usage, training, and testing time
- Achieves comparable performance to the one-hot coding

Future Research Direction

- Dataset creation for transliteration variation
- Applying Deep Learning techniques

References

1. Sarkar, K., Chatterjee, S.: Bengali-to-english forward and backward machine transliteration using support vector machines. In: J.K. Mandal, P. Dutta, S. Mukhopadhyay (eds.) Computational Intelligence, Communications, and Business Analytics, pp. 552–566. Springer Singapore, Singapore (2017).
2. Ekbal, A., Naskar, S.K., Bandyopadhyay, S.: A modified joint source-channel model for transliteration. In: Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pp. 191–198. Association for Computational Linguistics, Sydney, Australia (2006).
3. Rathod, P., Dhore, M., Dhore, R.: Hindi and marathi to english machine transliteration using svm. International Journal on Natural Language Computing (IJNLC) 2, 55–71 (2013).
4. Dasgupta, T., Sinha, M., Anupam, B.: Resource creation and development of an english-bangla back transliteration system. International Journal of Knowledge-based and Intelligent Engineering Systems 19, 35–46 (2015).
5. Roark, B., Wolf-Sonkin, L., Kirov, C., Mielke, S.J., Johny, C., Demirsahin, I., Hall, K.: Processing South Asian languages written in the Latin script: the Dakshina dataset. In: Proceedings of the Twelfth Language Resources and Evaluation Conference, pp. 2413–2423. European Language Resources Association, Marseille, France (2020).
6. Chen, N., Banchs, R.E., Zhang, M., Duan, X., Li, H.: Report of NEWS 2018 named entity transliteration shared task. In: Proceedings of the Seventh Named Entities Workshop, pp. 55–73. Association for Computational Linguistics, Melbourne, Australia (2018).

Thank You

Contributions

- First to use binary coding for Bengali language
- Reduced computational cost for SVM and random forest
- Provides algorithm for TU identification
- Performs well for frequently used words

Algorithms for TU Decomposition

Algorithm 3 Bengali TU Decomposition $(word)$

for $ch \in word$ **do**
    if $prev = hosonto$ or $prev \in prefixes$ or $ch \in matras$ or $ch = hosonto$ **then**
        $tu \Leftarrow tu + ch$
    else
        add tu to $tulist$
        $tu \Leftarrow ch$
    end if
    $prev \Leftarrow ch$
end for

Algorithm 4 English TU Decomposition $(word)$

for $ch \in word$ **do**
    if $prev \in vowels$ and $ch \in consonants$ **then**
        add tu to $tulist$
        $tu \Leftarrow ch$
    else
        $tu \Leftarrow tu + ch$
    end if
    $prev \Leftarrow ch$
end for

Algorithms for Encoding

Algorithm 5 Ordinal Encoding (*tu_list*, *all_tu*)

```
for tu ∈ tu_list do
    index ⇐ all_tu.index(tu)
    append index to tu_arr
end for
return tu_arr
```

Algorithm 6 One-Hot Encoding (*tu_list*, *all_tu*)

```
length ⇐ len(all_tu)
tu_arr ⇐ np.zeros((len(tu_list), length, ), dtype = int)
for tu ∈ tu_list do
    index ⇐ all_tu.index(tu)
    tu_arr[ind, index] = 1
end for
return tu_arr
```

Algorithm 7 Binary Encoding (*tu_list*, *all_tu*)

```
length ⇐ len(all_tu) + 1
total_bits = len(bin(length)[2 :])
tu_arr = np.zeros((len(tu_list), total_bits, ), dtype = int)
ind ⇐ 0
for tu ∈ tu_list do
    index ⇐ all_tu.index(tu) + 1
    tu_bin = bin(index)[2 :]
    number ⇐ total_bits − len(tu_bin)
    if number > 0 then
        prepend number of 0 to tu_bin
    end if
    for (i, c) ∈ enumerate(tu_bin) do
        tu_arr[ind, i] ⇐ int(c)
    end for
    ind ⇐ ind + 1
end for
return tu_arr
```

Results

Dataset	Model	SVM		RFC	
	Encoding	One-hot	Binary	One-hot	Binary
Dakshina	TUAR (%)	79.5996	**80.1939**	78.2583	**80.7770**
	WAR (%)	49.0020	**49.9840**	46.2166	**51.0109**
NEWS 2018	TUAR (%)	78.4124	**79.2669**	78.6131	**80.5596**
	WAR (%)	49.3456	**50.2046**	49.6626	**52.3730**

Dataset	Model	SVM		RFC	
	Encoding	One-hot	Binary	One-hot	Binary
Dakshina	Precision (%)	77.0093	**77.6260**	79.1452	**79.2024**
	Recall (%)	79.5996	**80.1939**	78.2583	**80.7770**
	F1 Score (%)	77.1255	**77.9520**	77.8466	**79.2719**
NEWS 2018	Precision (%)	**74.7564**	74.1313	**78.7315**	77.0546
	Recall (%)	78.4124	**79.2669**	78.6131	**80.5596**
	F1 Score (%)	75.1591	**75.5858**	77.5652	**77.8996**

Results

Dataset	Model	Encoding	Train		Test	
			Time (sec)	Ratio	Time (sec)	Ratio
Dakshina	SVM	One-hot Binary	34974.3703 **102.7984**	**340.2229**	2159.3266 **490.5969**	4.4014
	RFC	One-hot Binary	2611.1594 **253.8891**	10.2847	39.3453 **24.2984**	1.6193
NEWS 2018	SVM	One-hot Binary	2423.8516 **27.2703**	**88.8825**	129.5156 **82.5688**	1.5686
	RFC	One-hot Binary	498.4031 **61.1094**	8.1559	5.5922 **4.6453**	1.2038