Enhancing Hindi Feature Representation Through Fusion of Dual-Script Word Embeddings

Lianxi Wang[1,2], **Yujia Tian**[1*], **Zhuowei Chen**[1*†]
[1]School of Information Science and Technology,
Guangdong University of Foreign Studies, Guangzhou, China, 510006
[2]Guangzhou Key Laboratory of Multilingual Intelligent Processing, Guangzhou, China, 510006
{wanglianxi, 20211003065, 20211003051}@gdufs.edu.cn

Abstract

Pretrained language models excel in various natural language processing tasks but often neglect the integration of different scripts within a language, constraining their ability to capture richer semantic information, such as in Hindi. In this work, we present a dual-script enhanced feature representation method for Hindi. We combine single-script features from Devanagari and Romanized Hindi Roberta using concatenation, addition, cross-attention, and convolutional networks. The experiment results show that using a dual-script approach significantly improves model performance across various tasks. The addition fusion technique excels in sequence generation tasks, while for text classification, the CNN-based dual-script enhanced representation performs best with longer sentences, and the addition fusion technique is more effective for shorter sequences. Our approach shows significant advantages in multiple natural language processing tasks, providing a new perspective on feature representation for Hindi. Our code has been released on https://github.com/JohnnyChanV/Hindi-Fusion.

Keywords: Hindi language models, Feature representation, Feature fusion, Dual scipts

1. Introduction

The emergence of pre-trained language models (PLMs) has ushered in a new era of natural language processing (NLP). These models have achieved remarkable advancements in languages with abundant resources. However, low-resource languages like Hindi, also exhibit clear limitations, including the lack of language processing tools, evaluation tasks, and datasets.

Hindi, as one of the official languages of India, is written in Devanagari script. The script has a unique structure, with each character representing a specific syllable. Apart from that, Hindi is also written in Roman script, the script is often used as a transliteration system to represent Hindi words and phrases using the Latin alphabet, which can represent one Hindi syllable with multiple letters. While Devanagari is the preferred script for writing and reading Hindi, the Roman script is commonly used in informal contexts, such as social media and messaging platforms.

The various scripts used within a single language essentially reinterpret the semantic content of sentences, placing different emphases on semantics. This notion encourages the idea that merging diverse scripts of a single language could enhance the effectiveness of feature representation. The idea that fusing characters and romanized sequence representation can enhance the expressiveness of NLP models has been proven by Sun et al. (2021), which incorporates character, glyph, and romanized sequence (pinyin) in Chinese language representation. Essentially, the Romanised sequence of Devanagari represents one Devanagari character with multiple Latin letters, which is similar to the Romanised sequence of the Chinese character, pinyin. These ideas motivate us that combining different scripts in Hindi will bring model performance improvement by capturing richer semantic information.

To the best of our knowledge, no existing Hindi language representations are built from dual scripts, and most Hindi language models are built on a single script by conversion. In the case that the Romanized script provides more accurate pronunciation information, and Hindi PLMs also perform differently from script to script, we consider that fusing the features from different scripts can bring advantages to the performance of language models for Hindi. Our research proposes an efficient dual-script representation method for Hindi. The major contributions of this paper can be summarized as follows:

- We propose a dual-script enhanced representation method for Hindi, which generates representation by fusing features from a single script Hindi Roberta.
- Incorporating four fusion techniques into the representation method effectively generates dual-script representation from single-script representation. These techniques include concatenation, addition, cross-attention, and con-

*Equal contributions.
†Corresponding authors.

LREC-COLING 2024, pages 5966–5976

volutional networks.

- Evaluate the representation methods by three different categories of NLP tasks, i.e. sequence generation, text classification, and natural language inference. Results show the effectiveness of dual-script representation methods.

2. Related Work

2.1. Hindi Language Models

In the field of NLP, representation models play a crucial role in converting text into numerical representations for various machine learning tasks, such as sentiment analysis, language translation, and named entity recognition. For Hindi, specific NLP models have been developed to process Hindi text, including Devanagari and Romanized texts.

To explore the representation of Hindi text, previous research by Modha and Majumder (2019) conducted a comprehensive comparison of multiple text representation schemes for Hindi, including bag-of-words (BoW) techniques, distributed word/sentence representations, and transfer learning of classifiers. Additionally, Hingmire et al. (2020) introduced the use of Message Sequence Charts (MSCs) as a representation to visualize Hindi narrative text. These works have significantly contributed to the advancement of Hindi representation models; however, they require substantial support from large corpora.

Huang et al. (2021) proposed HinPLMs, creating pre-trained language models (PLMs) for Hindi in Devanagari and Romanized scripts. Their models outperform existing models in various NLP tasks, especially part-of-speech tagging and named entity recognition. However, they have imbalanced performance in different tasks. The Romanized Hindi PLM excels in multi-label classification and natural language comprehension tasks, while the Devanagari Hindi PLM demonstrates superior performance in natural language inference and text classification tasks. In light of this, we aim to enhance model performance further by combining features from these two PLMs, seeking to leverage the strengths of both models. Therefore, this paper takes advantage of related pre-training language models and achieves good results.

Collectively, these studies have made significant contributions to the advancement and application of Hindi representation models in various NLP tasks and language-related challenges.

2.2. Feature Fusion Methods

Enhancing feature representation and performance of pre-trained language models by combining different forms of a single language has become an innovative and effective method (Zhang et al., 2022; Sun et al., 2021). These approaches usually have two crucial components including extracting separate features from a single form of the language and combining these features through feature fusion.

For many languages, there are different forms that can be used for communication and technical processing. For example, for Chinese, there are pinyin, glyph, and char (Sun et al., 2021). For Hindi, there are Devanagari chars and Romanized chars. Also, there is other information that can be used for feature representation enhancement, e.g., phonetic and grammatical information. In practice, Zhang et al. (2022) introduced disambiguate intonation for sentiment analysis (DISA), which combines text representation and phonetic information and leads to a SOTA performance of Chinese text representation. Moreover, Sun et al. (2021) introduced glyph, chars, and pinyin into the pretraining of BERT and resulted in a significant performance improvement. Additionally, Mutinda et al. (2023) combined sentiment lexicon, N-grams, and BERT features. For Hindi, Goyal et al. (2021) proposed enhanced word embeddings (EWE), which combined various grammatical text features including part-of-speech embeddings, word prefix embeddings, word suffix embeddings, and word length embeddings. These innovative approaches have significantly improved the computational power of deep learning methods for text analysis and inspired our idea of combining representations of Devanagari and Romanized script.

Feature fusion is a critical component in enhancing the performance of contemporary network architectures by integrating features from various layers or branches. Recent advancements have introduced more efficient feature fusion techniques, particularly in the realm of machine learning and deep learning. For instance, Fu et al. (2008) redefined multi-feature fusion as a subspace learning problem. Meanwhile, Mangai et al. (2010) conducted an extensive review of decision fusion and feature fusion techniques, providing valuable insights into pattern classification. Additionally, Li et al. (2017) proposed a deep fusion convolutional neural network (DF-CNN) tailored for multimodal 2D+3D facial expression recognition. Haghighat et al. (2016) introduced discriminant correlation analysis (DCA) for feature-level fusion, incorporating class associations into the correlation analysis of feature sets.

Multimodal information fusion also plays a big role in feature fusion. Previous classic works include Jiang et al. (2020), which provided a clear explanation of the scientific issues and future research directions of multi-modal information fusion in the field of data-driven emotion recognition, and then proposed a method based on deep denoising

volume. Multi-modal information fusion method of product autoencoder. Later, Gao et al. (2019) proposed a multi-modal feature fusion method based on the LSTM network. Similarly, Liang et al. (2020) proposed a new method for multi-modal information fusion and representation based on labeled multiple canonical correlation analysis (LMCCA).

Also, for fusing different feature representations of languages, conventional methods like dense connections, feature concatenation, and weighted element-wise summation have been extensively investigated in the context of image restoration, pattern classification tasks, and NLP, providing effective and robust fusion performance. Thus, Sun et al. (2021) applied concatenation and fully connected layer to fuse feature representations, Goyal et al. (2021) applied concatenation and GRU layer to fuse feature representations. Moreover, Mutinda et al. (2023) applied concatenation and CNN layer to fuse feature representations. All of these methods have reached a satisfying model performance improvement, therefore, we designed four different feature representation methods with the aim of investigating the fusion method for fusing Devanagari script and Romanized script features in Hindi.

2.3. Transliteration between Devanagari Text and Romanized Text

There have always been difficulties in converting text from Devanagari to Romanized scripts, and related work has been ongoing. Among them, Sodhar et al. (2019) noted, that efforts have been made to address the challenge of romanization of Denevagari texts. Modha and Majumder (2019) combined bag-of-words technology, distributed word/sentence representation, and classifier transfer learning to comprehensively compare various text representation schemes. This provides a preliminary idea for the conversion of Hindi.

Shiravale et al. (2021) performed an in-depth study on the recognition of Devanagari scene text using autoencoder CNN and encoder-decoder convolutional neural network models for text/background segmentation. Building on the success of their deep learning-based approach for English, they extended their work to establish scene text recognition benchmarks for three Indic scripts: Devanagari, Telugu, and Malayalam (Mathew et al., 2017).

In addressing challenges related to language identification (LID) of romanized text, Madhani et al. (2023) provided a simple yet effective solution. This solution is designed to address issues such as scarcity of training data and low LID performance when processing similar languages.

In the field of transliteration from Devanagari texts to Romanized texts, various methods have been developed. One well-known method is the lossless International Devanagari Transliteration Alphabet (IAST), which is widely recognized for its accuracy. Another commonly used method in NLP is the WXconv[1] transliteration method, which maps each vowel and consonant to a single Roman letter, which makes the computational process easy.

Velankar et al. (2021) explored contextual sentiment analysis of Hindi lyrics in Devanagari script using knowledge graph representation. Their situational analysis is stored as a knowledge base and updated through incremental learning methods.

HinPLMs, proposed by Huang et al. (2021), also adopted the WXconv transliteration method to train Roberta models for Devanagari Hindi and Romanized Hindi. Notably, the results show that Devanagari Roberta and Romanized Roberta perform well on different tasks, but neither is overwhelmingly superior.

The convergence of methods and research in transliteration and analysis of Indian scripts represents a significant contribution to the field, providing valuable insights and solutions to a variety of challenges.

3. Enhanced Hindi Feature Representation

3.1. Overview

In this section, we present the concept of dual-script enhanced feature representation for Hindi. As depicted in Figure 1, the dual-script generation process typically consists of two main stages: single-script representation generation and dual-script representation generation through feature fusion. In the single-script representation generation stage, we acquire separate single-script representations for Devanagari and Romanized Hindi text. In the subsequent dual-script representation generation stage, we merge these single-script representations using various fusion methods to obtain the dual-script enhanced Hindi feature representation.

3.2. Single Script Representation

3.2.1. Single Script Language Models

Text vectorization can be categorized into two fundamental methods: static and dynamic vectorization. Notably, two prominent models in this context are Word2Vec and Roberta. The Roberta model dynamically generates sentence vectors, while Word2Vec produces static vectors. Specifically, in Hindi, although the Roberta model demands greater computational resources, it leads to a substantial enhancement in text representation.

[1] https://pypi.org/project/wxconv/

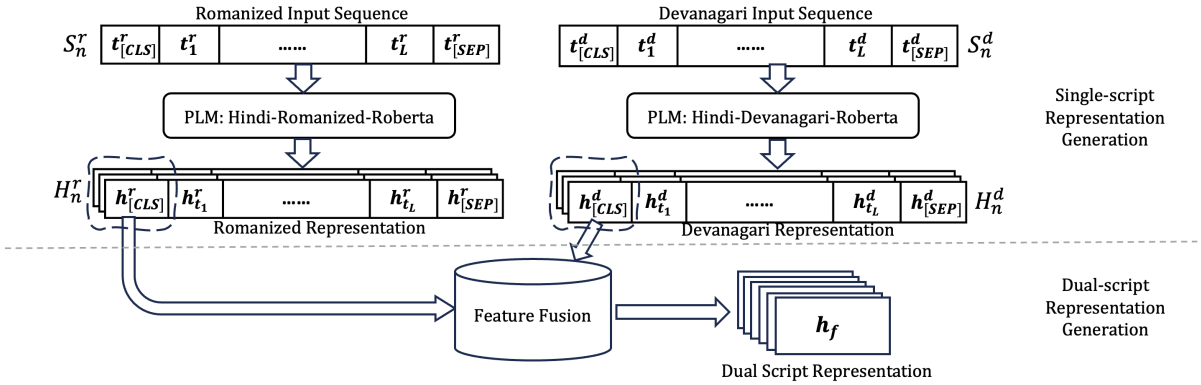

Figure 1: The dual-script sequence representation generation process.

For Hindi, researchers trained different language models on both scripts[2]. Among them, the following models are based on Roberta:

- **Hindi-Devanagari-Roberta**, which was trained on the original Devanagari corpus.
- **Hindi-Romanized-Roberta**, which was trained on the Roman corpus transformed from Devanagari.

Although these two PLMs reach best performances in several downstream tasks, they have imbalanced performance in different tasks. Therefore, we apply these two single-script Hindi PLMs in our dual-script representation method.

3.2.2. Representation Generation

In the realm of media and social media, sentences are predominantly either in Devanagari or Romanized Latin letters, with a few instances featuring a mixture of both scripts. To address this challenge and obtain representations in both scripts, we employ an open-source tool known as WXconv. This tool allows us to process, identify, and convert text from one script to the other as needed. It's worth noting that we have meticulously aligned this process with the corpus used in the PLM training to ensure consistency and prevent any performance degradation resulting from inconsistencies in data preprocessing methods.

After script transformation, we obtain two distinct representations for each sentence. To convert a sentence from the Devanagari script to a sequence of tokens, we employ the Hindi-Devanagari-Roberta tokenizer. Similarly, for the Romanized script, we use the Hindi-Romanized-Roberta tokenizer. This process yields two token sequences: $S_n^d = \{t_1^d, t_2^d, \ldots, t_L^d\}$ and $S_n^r = \{t_1^r, t_2^r, \ldots, t_L^r\}$. Here, S_n^d and S_n^r represent the $n-th$ sentence's Devanagari and Romanized token sequences, respectively. t_n^d represents a Devanagari token and t_n^r denotes a Romanized token, and L denotes the maximum sentence length. Additionally, we incorporate two special tokens, $[CLS]$ and $[SEP]$, to mark the beginning and end of each sequence, respectively.

Then we represent one sequence in two scripts using two PLMs mentioned above, which generates contextual representations $H = \{h_{[CLS]}, h_{t_1}, h_{t_2}, \ldots, h_{t_n}, h_{[SEP]}\}$:

$$H_n^r = \text{HindiRomanizedRoberta}(S_n^r) \quad (1)$$

$$H_n^d = \text{HindiDevanagariRoberta}(S_n^d) \quad (2)$$

where h_{t_n} denotes the hidden feature of token t_n. As shown in Figure 1, we use $[CLS]$ token as sequence-level pooling, $h_{[CLS]}$ denotes its hidden features. Additionally, $h_{[SEP]}$ denotes the hidden feature of the specific ending token. To be precise, $h_{t_n}^r$ and $h_{t_n}^d$ denote the hidden feature of the Romanized and Devanagari tokens respectively.

3.3. Dual-Script Representation

Dual-script representation is generated by fusing single script sentence representation $h_{[CLS]}^r$ and $h_{[CLS]}^d$. To achieve a better performance of Hindi language representation by fusing features from models in two scripts, this paper introduces four effective fusion techniques, concatenation, addition, attention mechanism, and convolutional neural network (CNN). As shown in Figure 1, we obtain single-script representation and then apply these fusion techniques to generate dual-script enhanced representation.

3.3.1. Concatenation

Concatenation fusion generates a dual-script representation with dimensions of 768*2, as illustrated in Figure 2. The representations from both scripts

[2]https://huggingface.co/GKLMIP

are straightforwardly concatenated to create a combined representation. The equation below illustrates the exact operation of these representations:

$$h_f = [h^d_{[CLS]}||h^r_{[CLS]}] \quad (3)$$

where || represents the concatenation method and h_f denotes the fused representation, i.e., the dual-script representation.

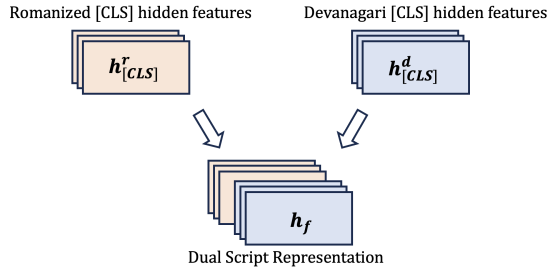

Figure 2: Concatenation feature fusion method.

3.3.2. Addition

In this method, aligned Devanagari and Romanized features are combined to produce a dual-script representation. The Romanized text representation is initially linearly transformed using a weighted matrix W and a non-linear activation function σ. The resulting dual-script representation is obtained by adding the Devanagari representation $h^d_{[CLS]}$ to the output of the activation function. Figure 3 provides a detailed illustration of this computational process. The mathematical representation of this fusion method is as follows:

$$h_f = h^d_{[CLS]} + \sigma(W \cdot h^r_{[CLS]}) + b \quad (4)$$

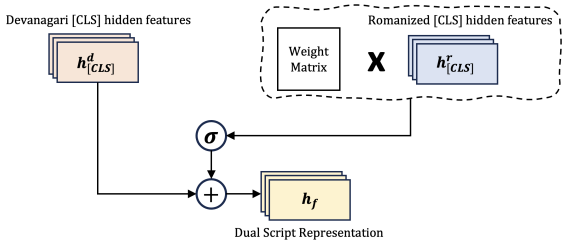

Figure 3: Addition feature fusion method.

3.3.3. Cross-attention

In the attention fusion method, we apply the cross-attention mechanism. In our method, the Romanized text representation acts as the query while the Devanagari representation acts as key and value. The weights of attention are calculated based on the similarity between query and key. Specifically, The calculation of multi-head self-attention is defined as:

$$\text{Att.}(Q, K, V) = \text{softmax}(\frac{QK^T}{d_k^{1/2}})V \quad (5)$$

where Q,K, and V stand for query, key, and value respectively. d_k denotes the dimension of K. Our dual-script representation h_f is generated by a cross-attention mechanism, its process is shown as follows:

$$h_f = \text{Att.}(h^d_{[CLS]}, h^r_{[CLS]}, h^d_{[CLS]}) \quad (6)$$

3.3.4. Convolutional Neural Network

After concatenation of $h^d_{[CLS]}$ and $h^r_{[CLS]}$, we further apply convolutional neural networks to extract high-level features to achieve the goal of compressing information, reducing dimensions, and enhancing robustness.

$$h_f = \sigma(\text{Conv}([h^d_{[CLS]}||h^r_{[CLS]}], W)) + b \quad (7)$$

where || signifies the concatenation operation, and Conv represents the convolutional operation applied to the concatenation of vectors $h^d_{[CLS]}$ and $h^r_{[CLS]}$. To be more precise, σ represents the activation function, W signifies the weight matrix, and b denotes the bias term.

4. Experiments and Analysis

4.1. Downstream Tasks and Datasets

We evaluate our methods on three categories of NLP tasks. Namely, text classification, sequence generation, and natural language understanding. Specifically, we evaluate our model on six datasets for Text Classification (TC), Natural Language Inference (NLI), Part-of-Speech Tagging (POS Tagging), and Named Entity Recognition (NER).

- **Text Classification**
  Text classification is a fundamental task in NLP where the goal is to predict the category or label of a given input text. In this context, we aim to validate the effectiveness of fusing features from two different scripts. To achieve this, we conduct experiments on three text classification datasets: BBC-articles (Kunchukuttan et al., 2020), which comprises mostly long texts, and IITP-movie-review, and IITP-product-review (Akhtar et al., 2016), both of which consist mainly of short texts.

- **Natural Language Inference**
  Natural Language Inference, also known as

textual entailment, has the goal of predicting the logical relation between two given texts, i.e., to judge whether the promise text can deduce the hypothetical text. Specifically, it has three situations: entailment, contradiction, and neutral. We experiment on a dataset called XNLI (Conneau et al., 2018). XNLI is a multilingual NLI dataset consisting of 15 languages. We extract Hindi data in XNLI for the Hindi NLI task.

- **Part-of-Speech Tagging**
  The goal of POS Tagging is to label each token with its corresponding Part-of-Speech. POS refers to the grammatical role that words play in a sentence, such as nouns, verbs, adjectives, adverbs, etc. We evaluate our methods on one POS Tagging dataset Hindi-HDTB, which is derived from the Hindi-Urdu Treebank Project (Bhat et al., 2017).

- **Named Entity Recognition**
  The NER task involves the extraction and classification of named entities within a provided text. In the context of the Hindi NER task, our assessment of models is based on a recognized dataset established as an integral component of the IJCNLP-08 NER Shared Task for South and Southeast Asian Languages. This dataset comprises 9466 training samples and 803 testing samples. Furthermore, our approach adheres to the data pre-processing technique introduced by Goyal et al. (2021), wherein the NEN, NEM, and NETI tags are consolidated into a miscellaneous category, denoted as MISC.

4.2. Parameter Settings

We apply the same optimizer parameters and methods for experiments, including learning rate and weight decay. We take cross entropy as our loss function, Adam as our optimizer, and the learning rate is manipulated as 4e-5.

For different tasks and datasets, we only change the batch size and the max sequence length according to their statistical features. Table 1 shows these parameters.

Task	Dataset	BatchSize	Length
TC	product-reviews	64	128
	movie-reviews	32	512
	BBC-articles	32	512
NER	IJNLP	16	256
POS	Hindi-HDTB	64	128
NLI	XNLI	128	64

Table 1: Parameter settings of each dataset

4.3. Results

To assess the effectiveness of our dual-script representation method, we employ several evaluation metrics, including area-under-curve (AUC) value, accuracy, and macro F1 score. In addition, we also discuss the training time cost in text classification tasks. AUC measures the area under the ROC curve. Accuracy and F1 score are metrics that consider the overall performance across all classes in classification tasks.

Moreover, we take single script models as our baselines, i.e., Hindi-Devanagari-Roberta and Hindi-Romanized-Roberta, which are donoted as Devanagari and Romanized in our experiments. Also, Dual-CAT, Dual-ATT, Dual-ADD, and Dual-CNN correspond to the enhanced dual-script representation generated by concatenation, cross-attention, addition, and convolutional network fusion techniques respectively.

4.3.1. Results of Text Classification Tasks

For text classification tasks, we evaluate model performances on three datasets. Table 2 shows model performances on TC datasets.

For short-sentence classification, corresponding to dataset IITP-product-review, results show an advantage of the dual-script method. Specifically, the addition-based fusion method overwhelmed single script methods and other fusion methods on accuracy and F1 values. The concatenation fusion method has a slightly higher value of AUC.

For long-sentence classification, corresponding to dataset IITP-movie-review and BBC-articles, results also show a significant advantage of the dual-script method. Specifically, on dataset IITP-movie review, the CNN-based fusion method overwhelmed single script methods and other fusion methods on accuracy and F1 values, while the attention-based fusion method has a slightly higher value of AUC.

Moreover, as shown in Table 2, on the long sentence dataset BBC-articles, the addition fusion method overwhelms other methods on AUC value.

4.3.2. Results of Sequence Generation Tasks

For sequence generation tasks, as shown in Table 3, we evaluate our model on Hindi-HDTB and IJCNLP datasets which correspond to POS Tagging and NER tasks. Moreover, we do not show the results of the CNN-based fusion method, as token-level classification tasks usually should consider contextual information while CNNs will smash the contextual semantic information which leads to a catastrophic failure.

In POS Tagging, the addition fusion method shows a dominant performance in which accuracy

Dataset	Model	AUC	ACC	F1
IITP-product-review	Devanagari	0.8878	0.7591	0.7062
	Romanized	0.8836	0.7457	0.7156
	Dual-CONCAT	**0.8904**	0.7438	0.7150
	Dual-ATT	0.8749	0.7514	0.7171
	Dual-ADD	0.8876	**0.7610**	**0.7350**
	Dual-CNN	0.8862	0.7457	0.7035
IITP-movie-review	Devanagari	0.7446	0.5355	0.5332
	Romanized	0.7318	0.5290	0.5209
	Dual-CONCAT	0.7408	0.5484	0.5390
	Dual-ATT	**0.7538**	0.5387	0.4368
	Dual-ADD	0.7427	0.5484	0.5532
	Dual-CNN	0.7371	**0.5903**	**0.5863**
BBC-articles	Devanagari	0.8439	0.7344	0.3670
	Romanized	0.8386	0.7356	0.3473
	Dual-CONCAT	0.8369	0.7240	0.3171
	Dual-ATT	0.8049	0.7471	0.3198
	Dual-ADD	**0.8779**	0.7564	0.3167
	Dual-CNN	0.8276	**0.7621**	0.3081

Table 2: Model performances on the text classification task.

Dataset	Model	ACC	F1
IJNLP-TFM-NER	Devanagari	0.7794	0.8576
	Romanized	0.6275	0.7651
	Dual-CONCAT	0.7735	0.8571
	Dual-ATT	0.6412	0.7539
	Dual-ADD	**0.7892**	**0.8726**
Hindi-HDTB-POS	Devanagari	0.9699	0.9702
	Romanized	0.9518	0.9520
	Dual-CONCAT	0.9698	0.9700
	Dual-ATT	0.8930	0.8935
	Dual-ADD	**0.9702**	**0.9704**

Table 3: Model performances on sequence generation tasks.

Model	AUC	ACC	F1
Devanagari	0.6977	0.5026	0.4975
Romanized	**0.7234**	0.5343	0.5241
Dual-CONCAT	0.7127	0.5258	0.5212
Dual-ATT	0.7133	0.5279	**0.5247**
Dual-ADD	0.7112	0.5144	0.5074
Dual-CNN	0.7220	**0.5487**	0.5169
GPT-3.5-Turbo	–	0.4263	0.3911

Table 4: Model performances on NLI task.

and F1 score reach the best position.

In the NER task, dual-script methods also achieve a satisfying performance, in which the addition method dominates other methods on accuracy and F1 values.

In sequence generation tasks, NER and POS Tagging, we fuse token-level representation by combining token features according to their indexes in the sequences. However, in practice, the Romanized script is transformed from Devanagari text, which is a sequence-to-sequence process. Meanwhile, the tokenization process of each script varies, which leads to a result that under an index, we can not always get aligned corresponding features in the two scripts.

To solve that problem, in addition fusion method, we apply a linear layer to the pre-fuse Romanized token features before summing them with the Devanagari token features. This process creates a mapping matrix to align features from different scripts which mitigates the referenced problem. In the results, we can easily reveal that the addition fusion

Dataset	Models	ACC	F1
IJNLP-TFM-NER	Dual-ADD	**0.7892**	**0.8726**
	m-Dual-ADD	0.7013	0.7963
Hindi-HDTB-POS	Dual-ADD	**0.9702**	**0.9704**
	m-Dual-ADD	0.9677	0.9682

Table 5: Results of ablation experiments focus on the linear mapping layer.

method shows a significant advantage compared with other methods.

4.3.3. Results of Natural Language Understanding Tasks

For the NLI task, as shown in Table 4, we observe a performance improvement in the CNN fusion method which reaches 0.5487 in accuracy.

In sentence-level classification tasks, TC and NLI, we fuse two sentence-level representations from different PLMs in two different scripts as a single dual-script representation. To be specific, we fuse $h^d_{[CLS]}$ and $h^r_{[CLS]}$. With sentence-level representations, we do sentence-level classification. Compared with single-script methods, dual-script methods shows generally better performances. However, in dual-script methods, different fusion methods do not show a result that one method dominates others, different fusion methods show similar performances.

As the creation of generic generative language models sweeps the entire NLP field, we also perform NLI experiments on gpt-3.5-turbo published by OpenAI. Results show that the gpt-3.5-turbo[3] model catastrophically failed on the NLI task in Hindi, with the lowest accuracy and macro F1 score between the models mentioned above. Specifically, we prompt the gpt-3.5-turbo model to do such a task, detailed prompting information is shown in Appendix A.

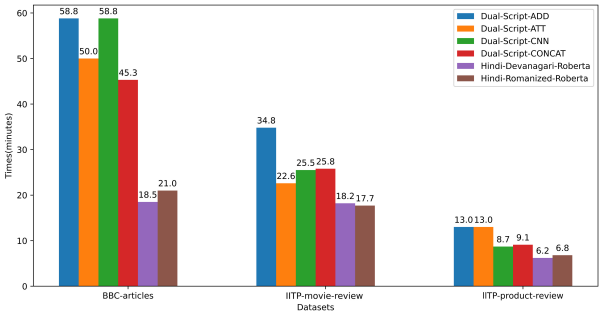

Figure 4: Training time cost of various models in the text classification task.

[3]https://platform.openai.com

4.3.4. Results of Training Time Cost

Figure 4 illustrates the divergences in training time costs for text classification tasks. Generally, dual-script models take a longer time to converge compared to single-script models. Specifically, for long sequence datasets like BBC-articles, dual-script models require over twice the time to converge compared to single-script models. However, for shorter sequence datasets, the additional training time for dual-script models is mostly less than 10 minutes longer than for single-script models.

The increase in training time cost can be attributed to several factors, including script transliteration, the utilization of two PLMs, and the expansion of parameters in fusion layers. Nonetheless, it's important to note that the dual-script text representation method delivers significant performance improvements, making the increase in cost acceptable.

4.4. Ablation Study

Our ablation study focuses on the effect of the linear mapping layer. This mapping layer is integrated in the addition fusion method, to eliminate the tokenization difference between two different scripts.

The tokenization difference problem arises when integrating PLMs for sequence classification. For instance, in English, the sequence "Hello Italy." can be tokenized as ["hello", "italy","[PAD]","[PAD]"], while its corresponding pronunciation mark sequence "hə'lou 'Itəli" can be tokenized as ["hə'", "lou", "'Itə", "li"]. Specifically, during sequence classification tasks on ["hello", "italy"], it becomes necessary to associate features of ["hə'", "lou"] with "hello", and ["'Itə", "li"] with "italy".

Table 5 shows the performance variances observed in linear mapping layer ablation experiments of the Dual-ADD fusion method. Models starting with m- in the table have the linear mapping layer removed. Specifically, the accuracy and macro F1 score dropped by eight percent while this layer was removed in the NER task. Moreover, the accuracy and macro F1 decreased by one percent while this layer was removed in the POS Tagging task.

The ablation study results examine that the linear mapping layer creates a mapping matrix to align features from different PLMs of different scripts. It also shows the significance of eliminating tokeniza-

tion differences in sequence classification tasks when combining features from different PLMs.

4.5. Discussion

In the proposed method for enhancing feature representation using dual scripts, we depart from the conventional approach where researchers typically combine various features from a single script within one language. Instead, we combine features from different scripts within the same language.

Experiment results on multiple datasets show the advantage of the dual-script feature representation method. Models with dual-script representations generally perform better in the tasks above. It also provides evidence for the effectiveness of combining different scripts of a single language in feature representation. It's worth noting that the dual-script approach outperforms the single-script method overall. In sequence generation tasks, the addition-based fusion method stands out as highly effective. In contrast, there isn't a significantly superior fusion method for sequence classification tasks, thus, features of datasets should be taken into consideration to get the best performance. To be more specific, these findings underscore the potency of the dual-script enhanced representation when combined with the addition-based fusion method in sequence generation tasks. Additionally, it demonstrates the capacity of this approach to enhance model performance in tasks such as TC and NLI.

Furthermore, we assessed the performance of GPT-3.5-turbo on the Hindi NLI task, where it encountered significant challenges. This underscores the importance of developing language-specific expertise models for addressing such issues and shaping the future of language understanding AI.

Nonetheless, employing two PLMs to create a dual-script representation inevitably results in a noteworthy escalation in both inference and training time. Furthermore, the incorporation of two PLMs into a single network leads to an expansion in the parameter space, feature dimensions, and network depth, posing substantial challenges in optimizing the dual-script network. Additionally, the heightened complexity and depth of the network introduce uncertainties and safety concerns.

5. Conclusion

In this paper, we propose dual-script enhanced word embedding representation methods for Hindi. We introduce four different fusion methods for dual-script word embedding fusion, namely, concatenation, cross-attention, addition, and CNNs. Moreover, we conduct experiments on TC, NLI, POS Tagging, and NER tasks, which in general show that dual-script methods have significant advantages and outperform single-script representation methods. It can be seen that different knowledge was learned by PLM in different scripts of one language, and combining them benefits language models. It excavates the potential of combining different scripts of one language with word representations. This inspiration can be applied to other languages which have more than one script, such as Chinese. It will provide new ideas to promote the integration of language models.

Acknowledgements

This work was supported by the National Social Science Fund of China (No. 22BTQ045).

6. Bibliographical References

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A hybrid deep learning architecture for sentiment analysis. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 482–493.

Riyaz Ahmad Bhat, Rajesh Bhatt, Annahita Farudi, Prescott Klassen, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, Ashwini Vaidya, Sri Ramagurumurthy Vishnu, et al. 2017. The Hindi/Urdu treebank project. *Handbook of linguistic annotation*, pages 659–697.

Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*.

Yun Fu, Liangliang Cao, Guodong Guo, and Thomas S Huang. 2008. Multiple feature fusion by subspace learning. In *Proceedings of the 2008 international conference on Content-based image and video retrieval*, pages 127–134.

Lei Gao, Rui Zhang, Lin Qi, Enqing Chen, and Ling Guan. 2019. The labeled multiple canonical correlation analysis for information fusion. *IEEE Transactions on Multimedia*, 21(2):375–387.

Archana Goyal, Vishal Gupta, and Manish Kumar. 2021. A deep learning-based bilingual Hindi and Punjabi named entity recognition system using enhanced word embeddings. *Knowledge-Based Systems*, 234:107601.

Mohammad Haghighat, Mohamed Abdel-Mottaleb, and Wadee Alhalabi. 2016. Discriminant correlation analysis: Real-time feature level fusion for multimodal biometric recognition. *IEEE Transactions on Information Forensics and Security*, 11(9):1984–1996.

Swapnil Hingmire, Nitin Ramrakhiyani, Avinash Kumar Singh, Sangameshwar Patil, Girish Palshikar, Pushpak Bhattacharyya, and Vasudeva Varma. 2020. Extracting message sequence charts from Hindi narrative text. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 87–96, Online. Association for Computational Linguistics.

Xixuan Huang, Nankai Lin, Kexin Li, Lianxi Wang, and Suifu Gan. 2021. HinPLMs: Pre-trained language models for Hindi. In *2021 International Conference on Asian Language Processing (IALP)*, pages 241–246. IEEE.

Yingying Jiang, Wei Li, M. Shamim Hossain, Min Chen, Abdulhameed Alelaiwi, and Muneer Al-Hammadi. 2020. A snapshot research and implementation of multimodal information fusion for data-driven emotion recognition. *Information Fusion*, 53:209–221.

Anoop Kunchukuttan, Divyanshu Kakwani, Satish Golla, Avik Bhattacharyya, Mitesh M Khapra, Pratyush Kumar, et al. 2020. Ai4bharat-indicnlp corpus: Monolingual corpora and word embeddings for Indic languages. *arXiv preprint arXiv:2005.00085*.

Huibin Li, Jian Sun, Zongben Xu, and Liming Chen. 2017. Multimodal 2d+ 3d facial expression recognition with deep fusion convolutional neural network. *IEEE Transactions on Multimedia*, 19(12):2816–2831.

Qi Liang, Ning Xu, Weijie Wang, and Xingjian Long. 2020. Multimodal information fusion based on lstm for 3d model retrieval. *Multimedia Tools and Applications*, 79:33943–33956.

Yash Madhani, Mitesh M. Khapra, and Anoop Kunchukuttan. 2023. Bhasa-abhijnaanam: Native-script and romanized language identification for 22 Indic languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 816–826, Toronto, Canada. Association for Computational Linguistics.

Utthara Gosa Mangai, Suranjana Samanta, Sukhendu Das, and Pinaki Roy Chowdhury. 2010. A survey of decision fusion and feature fusion strategies for pattern classification. *IETE Technical review*, 27(4):293–307.

Minesh Mathew, Mohit Jain, and CV Jawahar. 2017. Benchmarking scene text recognition in devanagari, telugu and malayalam. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 7, pages 42–46. IEEE.

Sandip Modha and Prasenjit Majumder. 2019. An empirical evaluation of text representation schemes on multilingual social web to filter the textual aggression. *arXiv preprint arXiv:1904.08770*.

James Mutinda, Waweru Mwangi, and George Okeyo. 2023. Sentiment analysis of text reviews using lexicon-enhanced bert embedding (lebert) model with convolutional neural network. *Applied Sciences*, 13(3):1445.

Sankirti S Shiravale, R Jayadevan, and Sanjeev S Sannakki. 2021. Recognition of devanagari scene text using autoencoder cnn. *ELCVIA: Electronic Letters on Computer Vision and Image Analysis*, 20(1):0055–69.

Irum Naz Sodhar, Akhtar Hussain Jalbani, Muhammad Ibrahim Channa, et al. 2019. Identification of issues and challenges in romanized Sindhi text. *International Journal of Advanced Computer Science and Applications*, 10(9):229–233.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. ChineseBERT: Chinese pretraining enhanced by glyph and Pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2065–2075, Online. Association for Computational Linguistics.

Makarand Velankar, Rachita Kotian, and Parag Kulkarni. 2021. Contextual mood analysis with knowledge graph representation for Hindi song lyrics in devanagari script. *ARXIV-CS.CL*.

Guobiao Zhang, Wenpeng Lu, Xueping Peng, Shoujin Wang, Baoshuo Kan, and Rui Yu. 2022. Word sense disambiguation with knowledge-enhanced and local self-attention-based extractive sense comprehension. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4061–4070.

Appendix A. Example Dialog of gpt-3.5-turbo in NLI Task.

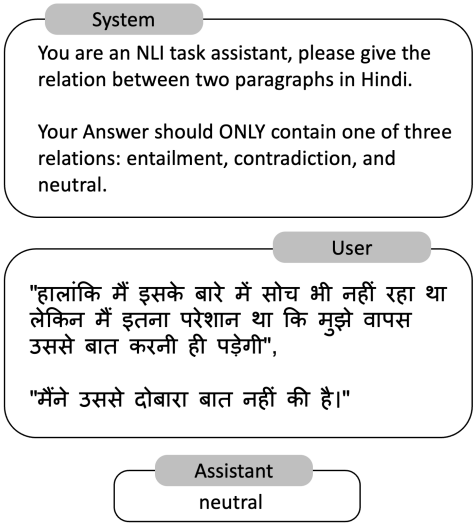

Figure 5: Example Dialog