






Semantic Term Weighting Representation for Kannada Document Classification

Ranganathbabu Kasturi Rangan^{1*}, Bukahally Somashekar Harish², Chaluvegowda Kanakalakshmi Roopa²

¹ Department of Information Science and Engineering, Vidyavardhaka College of Engineering, Mysore 570002, Karnataka, India

² Department of Information Science and Engineering, JSS Science & Technology University, Mysore 570006, Karnataka, India

Corresponding Author Email: rkrangan@vvce.ac.in

Copyright: ©2024 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/ria.380418>

ABSTRACT

Received: 3 November 2023

Revised: 12 December 2023

Accepted: 29 January 2024

Available online: 23 August 2024

Keywords:

Kannada documents classification, Natural Language Processing, positional encoding, Semantic Term weighting

In Natural Language Processing, the sequence order of terms plays a vital role in document categorization tasks. This positional sequence information aids in the natural language's semantic analysis. We proposed the semantic term weighting representation in response to the lack of semantic information in term weighting approaches. On the other hand, we created a collection of 11,045 Kannada documents dataset in response to the need for Indian regional language resources, particularly for the Kannada language. This dataset is asymmetrical and multilabel. The proposed dataset is subjected to the newly presented semantic term weighting representation techniques, like Term Frequency-Positional Encoding (TF-PE) and Term Frequency-Inverse Document Frequency-Positional Encoding (TF-IDF-PE). Further, the K-Fold and normal train-test split experimentations are carried out on the proposed dataset. Out of all the proposed representation techniques, Unicode encoded Term Frequency-Inverse Document Frequency-Positional Encoding (TF-IDF-PE) representation performed better than Term Frequency-Positional Encoding (TF-PE). In K-10 Fold experiments, the Unicode encoded TF-IDF-PE representation with the SVM classifier produces a greater average accuracy of 68.62%.

1. INTRODUCTION

Language is a great communication tool. People can share their expertise and express their feelings in an organized way by using natural language. The structures of these natural languages are intricate. It is therefore challenging for the machine to learn these. Natural language processing is composed of natural language understanding and natural language generation. In Natural Language Processing (NLP) tasks like sentiment analysis, text categorization, summarization, next-word prediction, etc., algorithms concentrate not only on morphological information or statistical information but also on semantic information. The semantic information enhances the text understanding level, following it leads to better results.

Irrespective of language, the sentence is a sequence of words that gives meaning. The sequence of order is rich in information. Bag-of-words or order-agnostic models are limited in their ability to use this word order information. As normal term weighting representations (vector space model) lacks in semantic information or input terms' sequence order, this information can be induced to term weights by positional encodings. This allows the algorithms to learn the relationships between the term vector space and semantic spaces.

Kannada is a regional language of India with limited resources. The corpus required for the several Kannada

language processing tasks is insufficient. Hence Kannada document dataset is proposed in this article. Further to categorize the Kannada documents, we explored and presented the new semantic based term weighting representation for better analysis of the documents.

1.1 Generic architecture

The Kannada document classification task is a subset of natural language understanding. The general architecture of this task is as shown in Figure 1. Document classification is the task of assigning the category c_i for the given document d_j , where $C = \{c_1, c_2, c_3, \dots, c_r\}$ are set of predefined "r" categories and $D = \{d_1, d_2, d_3, \dots, d_n\}$ are set of n documents.

The Figure 1 depicts the generic block diagram of Kannada document classification process. The raw Kannada text documents comprise of unwanted texts and removal of these is essential for better computation [1]. Stopwords removal, lemmatization, stemming, tokenization, and transliteration are few pre-processing techniques which aide in reducing the dimensionality and complexity of text processing. Further, mathematical representation methods are used for better representation of Kannada terms. In next phase, more informative and distinguishing features are selected by the feature selection methods. Later classifiers are applied to learn and classify the Kannada text documents. The evaluation of

the classification results is measured by various metrics like Accuracy, Precision, Recall, and F-measure.

In this paper, the main contribution is to create the dataset of 11,045 Kannada documents for Kannada text classification task. Further the hybrid representation method, which is the

agglutination of term weights with the position encodings, is discussed with empirical results. Section 2 is review of literature. Section 3 and 4 present the proposed methodology and experimentations carried out respectively. Finally, section 5 concludes with future scope and challenges.

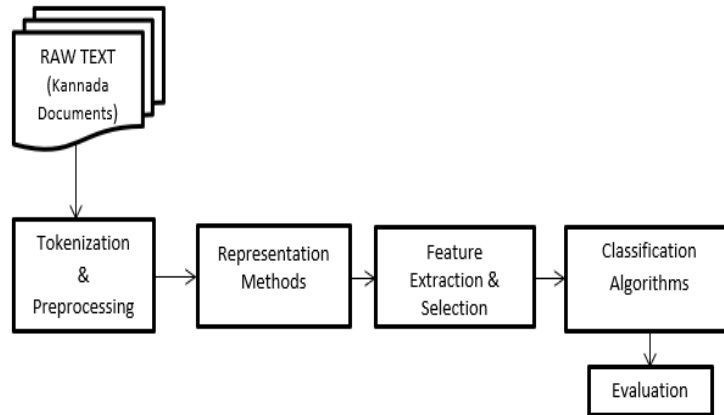


Figure. 1. The generic block diagram of Kannada document classification

2. RELATED WORK

In traditional term weighting scheme, the terms are represented in binary (or Boolean) but the term frequency (TF) in a document is apparently more precise and reasonable than binary 0 or 1, representing term presence or absence in the document [2, 3]. Term frequency (TF) based weighting may assign large weights to the terms with weak discriminating power. To overcome this shortcoming, a global factor Inverse Document Frequency (IDF) is embedded to form TF-IDF term weighting representation. The TF-IDF term weighting representation represents the importance of a term to a document in the dataset. Hence, TF-IDF is more reasonable than the binary and Term Frequency (TF) representations. For a term t_k , its TF-IDF weight is represented as follows:

$$w(t_k) = tf_k \cdot \log \left(\frac{N}{df_k} \right) \quad (1)$$

where tf_k is the term frequency of t_k in a document and df_k is the document frequency of t_k which represents the number of documents containing t_k , N is the total number of documents in the dataset. In Eq. (1), $\log \left(\frac{N}{df_k} \right)$ is the IDF factor and there are some other variants for IDF factor like $\log \left(\frac{N}{df_k} \right) + 1$ or $\log \left(\frac{N}{df_k + 1} \right)$. These variants address the null exceptions. In TF-IDF the known class information of train documents and position of terms information are not embedded in weights. On the other hand, term's significance in a document is overemphasized by TF-IDF. This leads to ignorance of terms contribution to text classification. To overcome the deficiency of TF-IDF, Debole et al. [4] worked on embedding the known class information in term weights and it is called Supervised Term Weighting (STW). In this Supervised Term Weighting (STW) methods, feature selection methods itself are used as global term weighting factor. For example, in Eq. (2), chi-square feature selection method is used, and it is named as TF-CHI method [4, 5]. Similarly,

Information gain in TF-IG [4, 3], gain ratio in TF-GR [4, 6], mutual information in TF-MI [7, 8] and so on. Further, addition to the term's specificity with respect to single document, Lertnattee and Theeramunkong [9] embedded the information of term's specificity to a class of documents and named as Inverse Class Frequency (ICF). The above-mentioned feature selection methods can reflect term's importance with respect to classes but still semantic information lacks in these weighting schemes.

$$w(t_k, c_j) = tf_k \cdot \chi^2(t_k, c_j) \quad (2)$$

Luo et al. and Wei et al. [10, 11] worked on semantic based term weighting. Here the term weighting is determined by term's semantic similarity with a specific class which can be represented with some key words. The semantic similarity or distance between terms is computed with the help of thesaurus known as WordNet. But accessing these external knowledge bases and exploring term similarity is complex. Hence this type of semantic based term weighting scheme has not shown significantly better performance than statistical schemes.

Further, in natural language the ordered sequence of terms exhibits the semantics of a sentence or document. Hence, the term's positional information can't be ignored. This positional encoding captures the vital information of term sequence order in a document. In major natural language generation tasks like machine translation, text summarization, question answering etc., transformer neural network architecture [12, 13] performs well and is most widely used encoder-decoder sequence-to-sequence model. This neural network architecture is presented by the authors Vaswani et al. [13]. It is a Self-attention-based network which relates different positions of a single sequence in order to compute a representation for the sequence of terms. Positional encoding extracts the absolute or relative [14] term's positional information. Further this information is used by many position-sensitive or attention language processing models like BERT [15], RoBERTa [16], and GPT-2 [17]. Positional encoding is also a part of transformer model. Shiv and Quirk [18] present a novel scheme of custom positional

encodings to extend transformers to tree domain tasks (especially binary trees). In the task of machine translation, Gehring et al. [19] presents convolutional sequence to sequence learning model. Here the positional encodings are used for gaining sequence information of terms.

Considering the task of Kannada text classification, Jayashree et al. [20] used a custom-built corpus called Technology Development for Indian Languages (TDIL) which is a comprehensive Kannada text resource, developed by Central Institute of Indian Languages (CIIL). They also experimented on the dataset built by fetching Kannada Wikipedia text. In the study [20], sentence level text classification is achieved. Similarly, summarization of categorized text documents in Kannada language is experimented in the study [21] based on sentence ranking technique. In the study [21], authors built their own dataset from Kannada webdunia web pages. At paragraph level of Kannada text, Jayashree et al. [22] worked on 1,791 Kannada text paragraphs. These paragraphs are categorized into 4 classes and belong to CIIL developed TDIL corpus and resulted with a precision of 0.74 using Multinomial naïve Bayes algorithm. Deepamala and Kumar [23] present a dataset consisting of 600 webpage Kannada text documents for classification task and there are 6 categories in the corpus. Further, 90% accuracy is claimed after applying the preprocessing techniques. Parameswarappa et al. and Caryappa et al. [24, 25] worked on algorithms to collect the Kannada web contents to build corpus and automatically check Kannada grammar, respectively.

Recently for Kannada language, researchers experimented more at character level recognition. As the availability of corpus is fewer, experimentations at the document level are moderate. The Kannada character recognition task is experimented on K-MNIST dataset by researchers [26]. In

their proposed work, CNN model outperformed with 98.77% of accuracy. At the preprocessing stage, lemmatization and stemming are two basic modules used for the normalization of text. Trishala and Mamatha [1] presented Unsupervised Stemmer and Rule-Based Lemmatizer for Kannada documents. Experimentation is carried out by building a dataset of 17,825 root words with the help of Kannada dictionary. The foregoing review exhibits the need of resources and computational methods in Indian regional languages specially Kannada for various document level Natural Language Understanding (NLU) tasks to perform.

3. PROPOSED METHOD

In Indian regional language text processing, the vital challenge is representation of the agglutinative and morphologically rich terms. Generally, the terms are transliterated to roman alphabet or entire text translated to English language. These two processes require a large corpus of more than one language and hence it is tedious and overdue. Other than formerly mentioned solutions for the term representation, Unicode based encoding can be performed and could represent the terms by unique number. Further in this paper, the incorporation of sequence order information to the document-term representation which overcomes the lack of semantic information in the vector space model is proposed. The schematic diagram of the proposed positional encoded vector space model is shown in the Figure 2.

From the Figure 2, the raw Kannada text documents are preprocessed by removing the roman numbers, punctuations, multiple spaces etc., Later the morphologically rich terms are encoded based on the Unicode term encoding. The details of these process are discussed in the following sections.

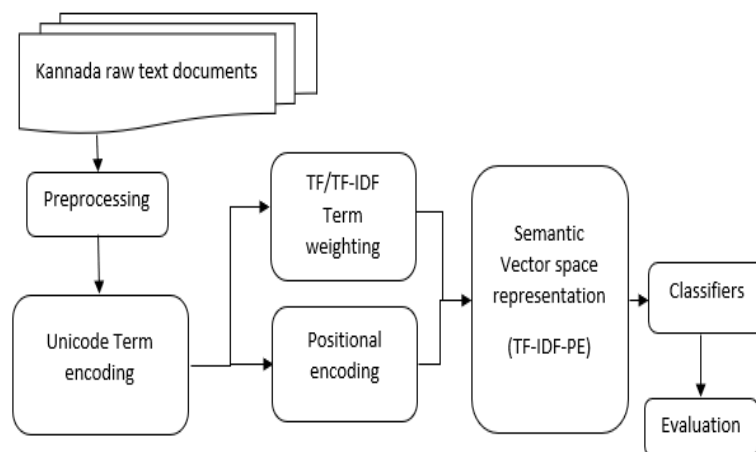


Figure 2. Proposed positional encoded based Kannada document classification model

3.1 Unicode term encoding

All natural language characters are identified uniquely by UNICODE [Universal coded character set] and Unicode helps to solve the incompatibility issue of ASCII encodings for natural language characters. The process of revamping the Unicode into sequence of bytes is called as encoding. In contrary, process of reversing sequence of bytes to Unicode is called as decoding. Every character of natural language is uniquely represented by code-point in Unicode. Code-points

are in the form of `\uXXXX`, where XXXX is the four-digit hexadecimal value and `\u` is an indicative for code-point. The sequence of these code-points makes Unicode strings to represent a term. As this sequence of code-points are too complicated to handle complex natural language terms, it is reduced to one unique decimal number by encoding the Unicode standard values / code points. Encoding byte addressing can be either big endian or little endian and further the encoded value is converted to decimal number which is unique for a term [27].

Table 1. An example for Unicode string Encoding and unique number generation

Characters	ಭ	ಾ	ರ	ತ
Unicode standard values (Code-Points)	\u0CAD	\u0CBE	\u0CB0	\u0CA4
Encoding (UTF-16)	b'\xff\xfe\xad\x0c\xbe\x0c\x0c\x0c\xa4\x0c'			
Decimal Representation	59694578067258861289215			

Ex: A Kannada (Indian regional language) term “ಭಾರತ” [English Translation: India] is Unicode encoded and represented by a unique decimal number as shown in the Table 1.

Kannada terms are agglutinative by nature, as is prevalent. Every phrase has morphologically rich characteristics due to the combination of vowels and consonants. Because of its complexity, the regular term representation omits the attached vowels. This decimal term representation is therefore necessary for every distinct term and addresses the loss of the original structure of the key terms.

3.2 Term weighting with positional encoding

Subsequently after the Unicode term encoding, document-term matrix is constructed based on the vector space model. In this document-term matrix, the term weighting is determined by term frequency (TF) or term frequency embedded with inverse document frequency (TF-IDF) or any other statistical measures for determining term weights as formerly discussed in related work section [28]. All these representations lack in the term’s sequence of order information which contributes more to the semantic input space. This term’s sequence order information can be determined by applying positional encoding technique.

In the recurrent neural networks, there is an inbuilt mechanism that takes care term’s sequence order through recurrence. In recurrent neural networks, challenges like gradient vanishing and exploding problems need to be addressed [18]. But in the transformer model [13], it is stateless sequence-to-sequence architecture which does not use recurrence or convolution and treats each term as independent of the other. Here, positional information is added to the transformer model explicitly to retain the information regarding the order of terms in a sentence. Positional encoding is the scheme through which the knowledge of sequence order of terms is maintained. Let us assume, ‘L’ is the sequence length of input, and we are finding the position information of ‘kth’ term in the sequence. The positional encoding is calculated by sine and cosine functions as mentioned in Eqs. (3) and (4).

$$P.E(pos_k, 2i) = \sin\left(\frac{pos_k}{n^{2i/d}}\right) \quad (3)$$

$$P.E(pos_k, 2i + 1) = \cos\left(\frac{pos_k}{n^{2i/d}}\right) \quad (4)$$

In Eqs (3) and (4), “pos_k” is position of the kth object in the input sequence, “d” is the dimension of output embedding space, “n” is user defined scalar which is set to 10000 based on empirical results [13], and “i” is used for index ranging between $0 \leq i < d/2$.

For an example, if we take a Kannada sentence “ನಾನು ಭಾರತ ದೇಶವನ್ನು ಪ್ರೀತಿಸುತ್ತೇನೆ”. Positional encoding for these terms is computed as shown in Table 2. As formerly mentioned, “n” and “d” are user defined, we considered n = 100, d = 4 and “i” ranges between $0 \leq i < 2$.

Table 2. Positional encoding matrix for a Kannada sentence

Term Sequence	Positional Index(k)	i = 0	i = 0	i = 1	i = 1
ನಾನು	0	P ₀₀ = sin(0)= 0	P ₀₁ =cos(0)= 1	P ₀₂ = sin(0)= 0	P ₀₃ = cos(0)= 1
ಭಾರತ	1	P ₁₀ =sin(1/1)= 0.84	P ₁₁ = cos(1/1)= 0.54	P ₁₂ =sin(1/10)= 0.10	P ₁₃ = cos(1/10)= 1
ದೇಶವನ್ನು	2	P ₂₀ =sin(2/1)= 0.91	P ₂₁ = cos(2/1)= -0.42	P ₂₂ =sin(2/10)= 0.20	P ₂₃ = cos(2/10)= 0.98
ಪ್ರೀತಿಸುತ್ತೇನೆ	3	P ₃₀ =sin(3/1)= 0.14	P ₃₁ = cos(3/1)= -0.99	P ₃₂ =sin(3/10)= 0.30	P ₃₃ = cos(3/10)= 0.96

From Table 2, we can infer that for the 4 kannada terms of a sentence, positional encoding values are calculated using Eqs. 3 and 4 where their positional values, and output embedding dimension are given as inputs.

As the sine and cosine trigonometric functions have values in the range of [-1, 1], the values of the positional encoding matrix will be in normalized range. Further due to the different sinusoidal value for each position, a unique way of encoding each position is achieved. In the proposed Kannada documents classification experiments, the “d” and “n” values are 2 and 10000 respectfully. The detailed empirical results are discussed in further sections.

Later the calculation of positional encoding values needs to be convoluted. This aide in embedding the sequence information in vector space model. For each term, there are sine and cosine positional encoded values, without losing the

semantic information we need to convolute. This convolution happens by sum of the trigonometric values. If the encoded values are represented in the wave forms and convoluted, this convolution leads to equivalent of 45-degree phase shift of resultant wave without losing any information. Hence this convolution can be acceptable. The mathematical proof for this convolution is as shown in below equations:

$$\sin(pos_k / n^{2i/d}) + \cos(pos_k / n^{2i/d}) = \sin(x) + \cos(x) \quad (5)$$

$$\sin(x) + \cos(x) = \sqrt{2} \left[\frac{1}{\sqrt{2}} \sin(x) + \frac{1}{\sqrt{2}} \cos(x) \right] \quad (6)$$

$$\sin(x) + \cos(x) = \sqrt{2} [\sin(45) \sin(x) + \cos(45) \cos(x)] \quad (7)$$

$$\sin(x) + \cos(x) = \sqrt{2} [\cos(x - 45)] \quad (8)$$

$$\sin(x) + \cos(x) = \sqrt{2} \left[\sin\left(x + \frac{\pi}{4}\right) \right] \quad (9)$$

From Eq. (5), “ x ” represents “ $pos_k/n^{2i/d}$ ”, which is common for both trigonometric functions. Multiply and divide Eq. (5) by $\sqrt{2}$ as shown in Eq. (5), later simplifications are followed in Eqs. (7), (8) and (9).

From the Eqs. (8) and (9), it's evident that sum of sine and cosine values leads to only a phase shift of 45° but not effecting any other aspects (wavelength and frequency). This preserves the term's positional information intact. Algorithm for positional encoding is presented in algorithm 1.

Algorithm 1: Positional encoding for Kannada document terms

Input: Sequence length of document, output embedding.

Data: $n=10000$ scalar value empirically determined [13], PE = Positional encoding

Output: Positional encoding matrix for Kannada document terms.

```

STEP 1: for  $k$  in range (sequence length)
STEP 2:   for  $i$  in range (output embedding / 2)
STEP 3:      $PE_{(k,2i)} = \sin\left(\frac{k}{n^{2i/output\ embedding}}\right)$ 
STEP 4:      $PE_{(k,2i+1)} = \cos\left(\frac{k}{n^{2i/output\ embedding}}\right)$ 
STEP 5:      $PE_k = PE_{k,2i} + PE_{k,2i+1}$ 
STEP 6:   end
STEP 7: end

```

In a document $d = \{t_1, t_2, t_3, \dots, t_k\}$, we can find the same term t_k in different positions. This information is also vital and cannot be ignored. Based on the normal Gaussian distribution analysis, the mean value of distribution of term's position values over the sequence order space is reckoned as shown in Eq. (11).

$$V_k = \sin(x_k) + \cos(x_k) \quad (10)$$

From Eq. (10), “ V_k ” is the added value of trigonometric positional encoding functions for a term.

$$\sum_{j=0}^m V_{jk} / m \quad (11)$$

The positional encoded value obtained by Eq. (11) for a term “ t_k ” in a document is embedded to its respective term weight “ tf_k ” in the vector space representation as shown in Eqs. (12) and (13). This enables the vector space term weights with semantic sequence information.

$$TW_k = tf_k + \left(\sum_{j=0}^m V_{jk} / m \right) \quad (12)$$

$$TW_k = tf_k \cdot \log\left(\frac{N}{df_k}\right) + \left(\sum_{j=0}^m V_{jk} / m \right) \quad (13)$$

From Eqs. (12) and (13), “ TW_k ” is the updated term weight with sequence information of a term “ t_k ”. Eq. (13) is the combination of TF-IDF with positional embedding. Similarly, Eq. (12) represents the combination of Term Frequency (TF) based term weights “ tf_k ” with positional encodings. The algorithm for the proposed representation is depicted in algorithm 2:

Algorithm 2: Positional encoded term weighting algorithm

Input: Kannada text documents $D = \{d_1, d_2, d_3, \dots, d_n\}$

Data and Functions: getPE() = positional encoding function, μ_{PE} = mean of positional encoded values, TW_k = Term weight matrix.

Output: Term Frequency-Positional encoded matrix or Term Frequency-Inverse document frequency-Positional encoded matrix.

```

STEP 1: In document  $d = \{t_1, t_2, t_3, \dots, t_k\}$  each raw
Kannada term is Unicode encoded.
STEP 2: for  $i \leftarrow$  to  $n$  by  $d_n$  do (for each document)
STEP 3: Sequence length = Total terms( $d_i$ ) (total terms in
each document)
STEP 4: getPE(Sequence length, output embedding =2,  $n$ 
= 10000)
STEP 5: if  $t_k = t_{k1}, t_{k2}, \dots, t_{km}$  (find the same terms
located in different positions)
STEP 6:  $\mu_{PE} = \frac{\sum_{j=1}^m t_{kj}}{m}$  (Positional encoding mean
of same terms)
STEP 7: for  $t_k$  in document term matrix (Count vectorized
matrix)
STEP 8: if  $t_{k(PE\ matrix)} = t_{k(document\ term\ matrix)}$ 
STEP 9: Final  $TW_k = \text{absolute}(\mu_{PE}) + tf_k$ 
(or)
=  $\text{absolute}(\mu_{PE}) + tf\_IDF_k$ 
STEP 10: end
STEP 11: end

```

The output of the positional encoded term weighting algorithm results in the document term weight matrix embedded with positional information. Further the machine learning algorithms are applied to learn and classify the Kannada documents. Finally, classification results are evaluated with accuracy metric.

3.3 The new dataset

As the Indian regional languages are less resourced, creation of large dataset for various regional language processing tasks have become essential. In the Kannada language, various researchers worked on creation of resource for different language processing tasks at different levels like at character, sentence, paragraph but at document level we are one of the few to build Kannada document corpus [20]. Authors [23] built their own Kannada text corpus of 600 webpage documents with 6 classes. Central Institute of Indian Languages (CIIL) is also building the Indian regional language corpus for various processing tasks but not publicly available to all. The proposed dataset is multiclass, large and expansion of our own dataset created formerly. The former dataset is of 1,007 documents collected from various sources, distributed among 8 classes. Details of this dataset is as shown in Table 3. The proposed dataset is of size 11,045 documents spread

across 10 classes and unbalanced. The detail of this dataset is presented in Table 4.

The dataset is created by the collection of Kannada news articles, Wikipedia articles which are available on internet. Another way is by digitizing the hard copy of open source publicly available Kannada articles using optical character recognition technique. The difficulty faced in scanning hard

copy is fabrication of terms i.e., some Kannada characters were recognized as other language characters [29]. This is due to the similarity of strokes of some Kannada characters as other language characters. Further there were challenges like unavailability of digital data, incorrect spellings, presence of multilingual words etc.

Table 3. The details of the former dataset

Label	Categories	No. of Documents	No. of Lines	No. of Terms	No. of Characters
1	Space & Science	70	3,219	36,332	2,63,893
2	Politics	244	13,836	1,36,035	9,89,346
3	Crime	55	2,390	27,584	2,00,692
4	Sports	60	2,406	30,343	2,03,424
5	Economics	50	2,106	25,143	4,12,617
6	Entertain-ment	100	6,538	57,488	4,12,617
7	Health	28	1,520	13,993	1,08,666
8	Stories	400	22,266	2,10,777	15,14,256

Table 4. The new dataset details

Label	Categories	No. of Documents
1	Space & Science	1697
2	Politics	744
3	Crime	136
4	Sports	457
5	Economics	904
6	Entertainment	2002
7	Health	486
8	Stories	794
9	Social Science	3009
10	Spiritual	816
TOTAL		11,045

4. EXPERIMENTS AND RESULTS

The proposed new dataset is used for the experimentations. Firstly, the Kannada documents are preprocessed by removing punctuations, roman numbers, irregular spaces, and single characters by using regular expressions. Followed by preprocessing, the Kannada terms are Unicode encoded resulting in creation of unique decimal number for each Kannada term. Now, the Kannada documents with Unicode encoded terms are count vectorized to have document-term matrix. In parallel the positional information is determined by using positional encoding algorithm which is presented in algorithm 2 of section 3. Major experimentations performed on new dataset can be categorized into two, like K-fold experimentations and random train-test split experimentations. Based on literature review, in both types of experimentations, classifiers like Support Vector Machine (SVM), K-Nearest Neighbor (K-NN), Decision Tree (DT) and Multi-Layer Perceptron (MLP) are considered, applied, and evaluated. In K-NN the ‘K’ value is set to 7, and in MLP, hidden layer size and iteration values are assigned to 1000 with RELU activation function. These values are considered empirically. Further at preprocessing stage, stop words are removed based on the term frequency in the documents.

In random train-test split experimentations, the ratio of train-test documents is 60:40, 70:30 and 80:20. As the

documents are randomly selected in these experiments, three consecutive experiments are conducted for each formerly mentioned ratio and average of these three empirical results is determined as presented in Table 5 and Table 6. In Table 5 the experimental results of proposed dataset are based on Term Frequency–Positional encoding (TF-PE) representation, similarly the empirical results in Table 6 are based on the Term Frequency-Inverse Document Frequency-Positional encoding (TF-IDF-PE) representations. The experimental results for non-Unicode encoded term weight representations of proposed kannada document dataset is less than 30%. On the other hand, Unicode encoded terms with TF-PE representations as presented in Table 5 performs better with 66.79% when MLP classifier is applied at the 80:20 train-test ratio. Further from Table 5, we can infer that SVM results at all train-test ratios are quite equivalent to MLP classifier results. Similarly in Table 6, experiments are based on Unicode encoded terms with TF-IDF-PE representations, the SVM classifier outperforms MLP in both 60:40 and 70:30 train-test ratios with 66.80% and 68.39% respectively. Even in 80:20 ratio MLP performs slightly better than SVM with 68.35% accuracy, it is same as SVM’s 70:30 ratio results. Hence, we can infer that SVM performs better across all train-test ratios of TF-IDF-PE representations. The Table 7 presents the comparative results for both TF-PE and TF-IDF-PE representations.

4.1 K-Fold experimentations

In normal train-test split experimentations, the train test split happens randomly and it’s difficult to determine and evaluate the results for whole dataset. Hence, in K-fold experimentations no document will be left over for testing. The K-fold experimentations bring better evaluation of learning algorithms on proposed dataset. As the proposed dataset is unbalanced and multilabel, Stratified K-fold validation is performed. The “K” parameter refers to number of splits to be formed for the given dataset. For an example, if K=2 then the dataset is divided into 2 folds, that is 11,045 documents is divided into 5,522 and 5,523 documents’ sets and each of it will be part of training and testing set. Later, the mean accuracy of those K-fold experiments is calculated.

Table 5. Experimental results for positional encoded term frequency representation (TF-PE)

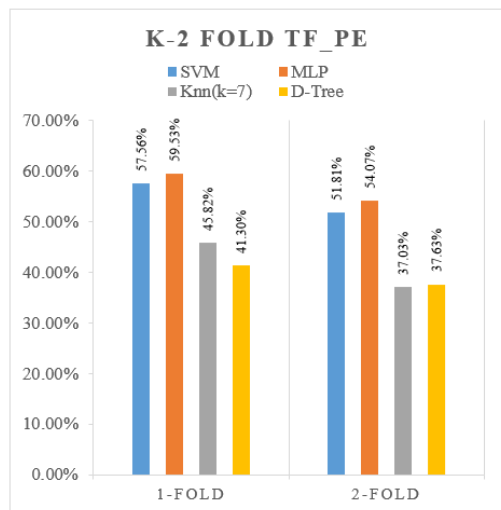
Train-Test Ratio	Features	SVM (%)	MLP (%)	KNN (%)	D-Tree (%)
60:40	746	64.69	63.98	46.06	42.44
	721	62.86	64.78	41.24	39.59
	736	64.31	65.64	45.02	44.91
Average Accuracy		63.95	64.80	44.11	42.31
70:30	968	64.94	65.84	47.40	44.81
	965	65.12	65.70	48.85	40.74
	970	65.39	65.72	47.83	45.20
Average Accuracy		65.15	65.75	48.03	43.58
80:20	1148	66.63	67.07	52.69	45.72
	1135	66.27	66.63	47.30	46.13
	1153	65.14	66.68	51.29	44.45
Average Accuracy		66.01	66.79	50.43	45.43

Table 6. Experimental results for positional encoded term frequency-Inverse document frequency representation (TF-IDF-PE)

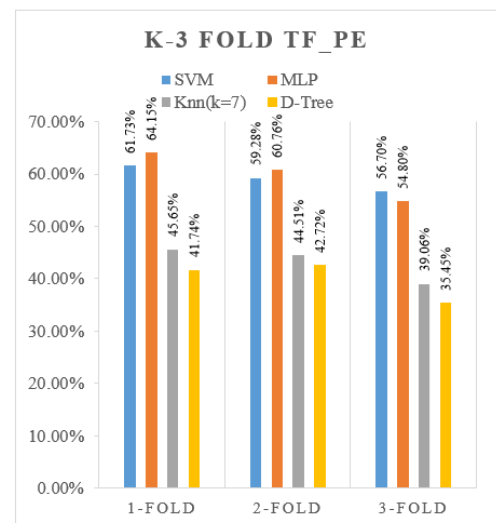
Train-Test Ratio	Features	SVM (%)	MLP (%)	KNN (%)	D-Tree (%)
60:40	756	66.70	65.14	41.01	41.83
	723	66.52	65.30	40.99	44.00
	741	67.18	65.41	40.38	44.48
Average Accuracy		66.80	65.28	40.79	43.44
70:30	968	68.44	66.42	39.50	45.10
	965	68.17	65.47	40.46	46.60
	966	68.56	67.11	43.40	45.50
Average Accuracy		68.39	66.33	41.12	45.73
80:20	1146	70.66	69.03	41.92	46.45
	1150	60.71	68.62	55.50	38.11
	1141	67.95	67.41	41.46	45.31
Average Accuracy		66.44	68.35	46.29	43.29

Table 7. Normal train-test split result comparisons for TF-PE and TF-IDF-PE representations

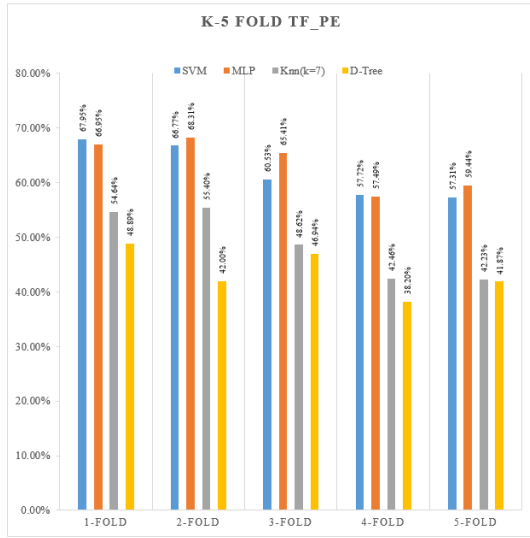
Train-Test Ratio	Representation	SVM (%)	MLP (%)	KNN (%)	D-Tree (%)
60:40	TF-PE	63.95	64.80	44.11	42.31
	TF-IDF-PE	66.80	65.28	40.79	43.44
70:30	TF-PE	65.15	65.75	48.03	43.58
	TF-IDF-PE	68.39	66.33	41.12	45.73
80:20	TF-PE	66.01	66.79	50.43	45.43
	TF-IDF-PE	66.44	68.35	46.29	43.29



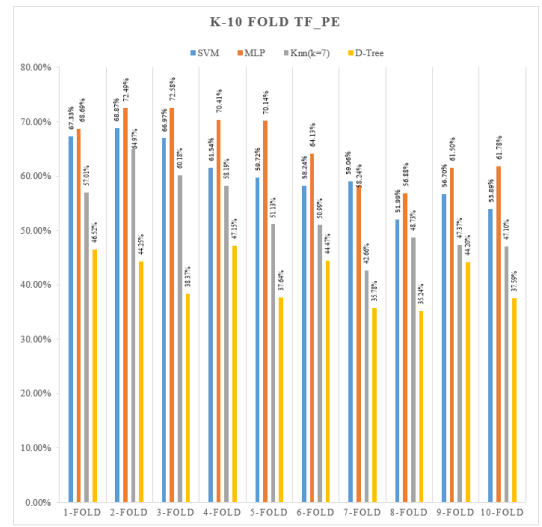
(a)



(b)

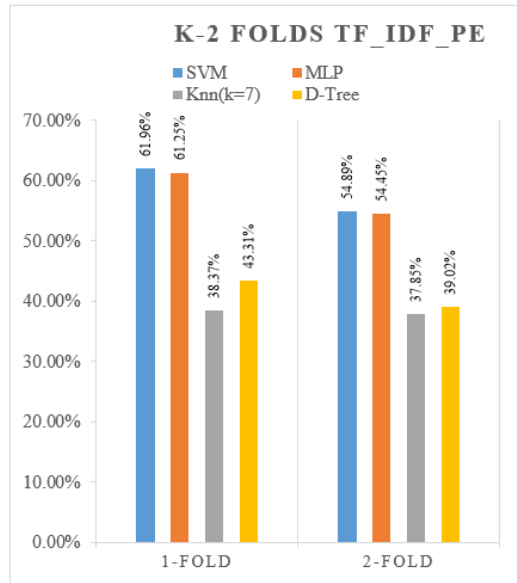


(c)

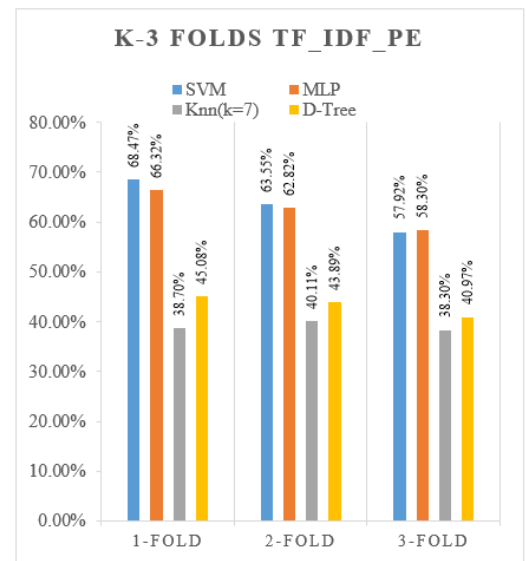


(d)

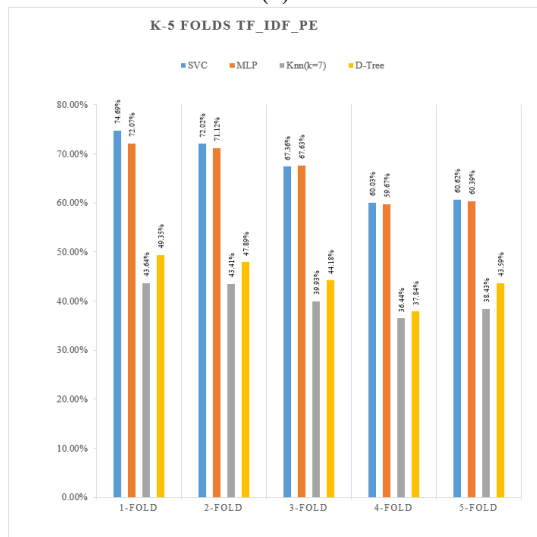
Figure 3. K-fold experimental results for TF-PE representation



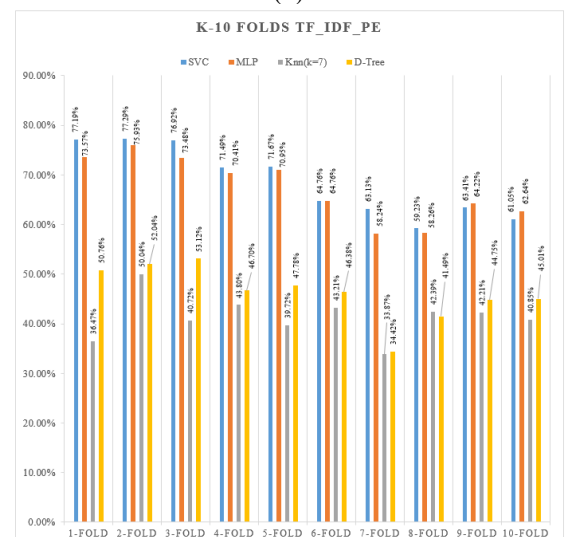
(a)



(b)



(c)



(d)

Figure 4. K-fold experimental results for TF-IDF-PE representation

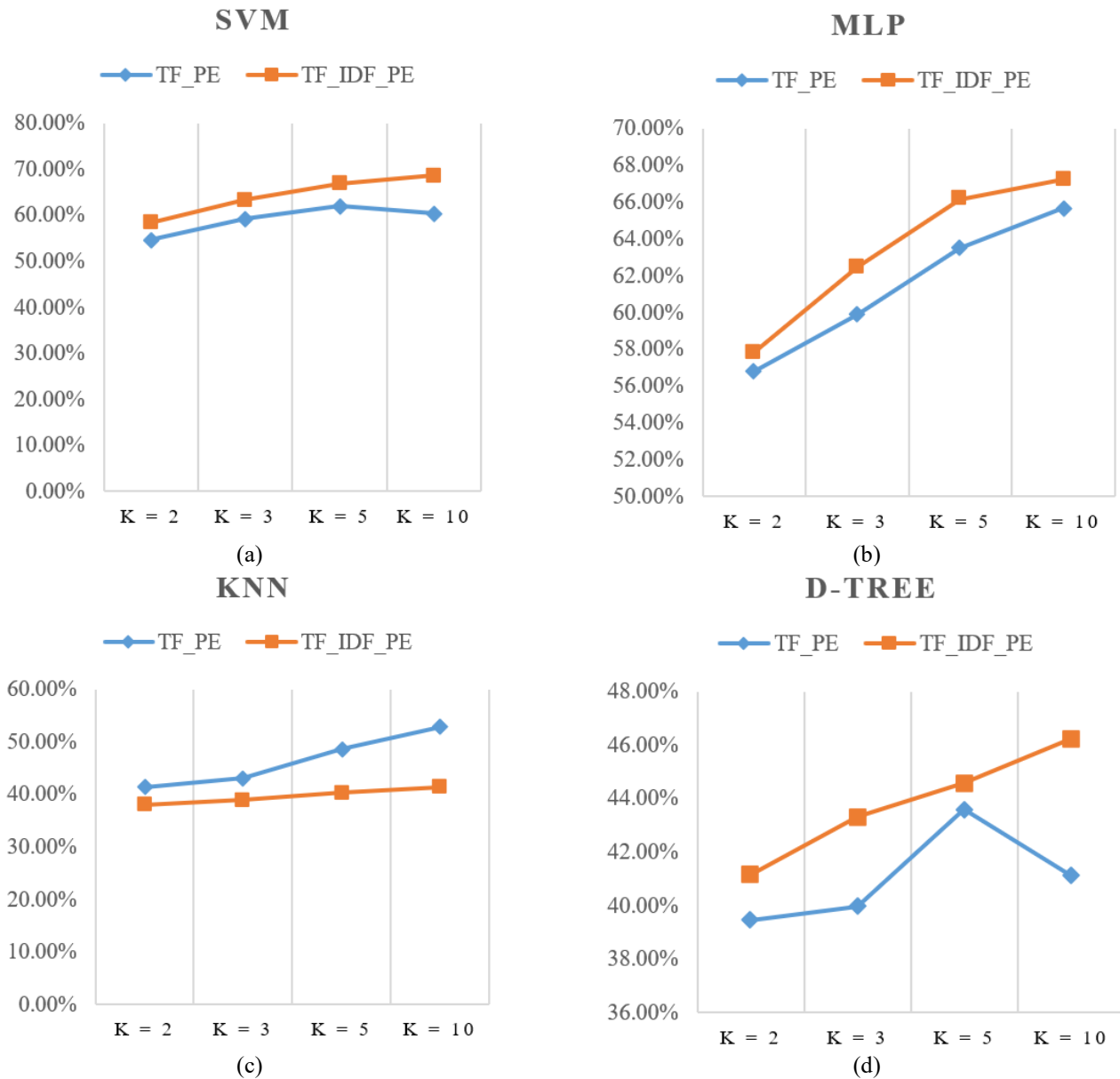


Figure 5. K-Fold result comparisons of TF-PE and TF-IDF-PE representations

In Figures 3 and 4, the various k-folds experimental results are presented with respect to TF-PE and TF-IDF-PE respectively. Among TF-PE and TF-IDF-PE, the TF-IDF-PE results are better from the perspective of all four classifiers. Specially for TF-PE representation with K-2 Fold experiments (Figure 3a), SVM classifier yields an average accuracy of 54.69% (Figure 5a, K=2), MLP yields an average accuracy of 56.80% (Figure 5b, K=2), K-NN (K=7) achieves an average accuracy of 41.43% (Figure 5c, K=2), and D-tree yields 39.47% of average accuracy (figure 5d, K=2). Further in K-3, K-5 and K-10 Fold experiments (Figure 3b, 3c, 3d), SVM classifier yields 59.24%, 62.06%, and 60.43% of respective average accuracy (Figure 5a). The MLP classifier yields 59.90%, 63.52%, and 65.68% of respective average accuracy (Figure 5b). The K-NN yields 43.08%, 48.67%, and 52.84% of respective average accuracy (Figure 5c). Lastly, D-Tree yields 39.97%, 43.59%, and 41.12% of respective K-Fold experimentations average accuracy (Figure 5d).

Similarly for TF-IDF-PE representations in K-2, K-3, K-5 and K-10 Fold experimentations, SVM classifier yields 58.42%, 63.31%, 66.94%, and 68.62% average accuracy of respective K-Fold experimentations (Figure 5a). The MLP

classifier yields 57.85%, 62.48%, 66.17%, and 67.25% average accuracy of respective K-Fold experimentations (Figure 5b). The K-NN yields 38.10%, 39.04%, 40.37%, and 41.33% of respective K-Fold experimentations average accuracy (Figure 5c). Finally, D-Tree yields 41.17%, 43.31%, 44.57%, and 46.25% average accuracy of respective K-Fold experimentations (Figure 5d).

In comparison with K-fold experimentations for TF-PE and TF-IDF-PE representations, TF-IDF-PE results are better with all learning algorithms (Figure 5). In K-2 Fold experimentations of TF-IDF-PE representation, except K-NN all other classifiers results are higher than K-Fold experimental results of TF-PE representation. Further, among all results of K-2 Fold experiments, SVM does better with 58.42% of accuracy (Figure 5a). Similarly, in TF-IDF-PE based K-3, K-5 and K-10 Fold experimentations, SVM does better with 63.31%, 66.94% and 68.62% of accuracy (Figure 5a). The notable observation made during the K-10 Fold experiments (TF-IDF-PE representation) is, the SVM classifier yielded 77.29% of accuracy for certain set of train-test document split, which is maximum when compared to overall experimental results (Figure 4d).

5. CONCLUSION

The new semantic based term weighting representation is proposed in this paper. Further to enrich the regional language documents' corpus, a new dataset of 11,045 Kannada documents is created. The new dataset is experimented with K-Fold experimentations, in addition to normal random train test splits. In term weighting representations, the sequence order of terms is not considered. Hence it leads to loss of semantic information, which is essential for better natural language understanding. To overcome this issue, the sequential information of the terms is embedded to the normal term weights by using the positional encoding technique. The sequential information contributes a lot in the regional language understanding like the Kannada document classification. In this regard, the representation techniques like TF-PE and TF-IDF-PE are applied on the newly created dataset. From the experimental results, it is found that in the spatial-semantic space SVM, and MLP learning algorithms performed better. In all the experiments, TF-IDF-PE term weighting representation performed well. Further in TF-IDF-PE representation, when compared to all K-Fold experimental results SVM performs slightly better than MLP with 68.62% average accuracy in K-10 Fold experiments applied on proposed dataset.

Hereinafter, the semantic representation could be improved by incorporating various statistical methods used for binding the positional encoding values to the term weights. As positional encoding method is based on continuous function it is arduous to embed with the term weighting representations which is of discrete. Hence it requires better empirically proven statistical methods. Further, the various other neural network classifiers with positional encoding embedded architecture could be explored on this new dataset. Finally, the volume of Kannada documents could be increased. This is more beneficial and essential for various regional language processing tasks.

REFERENCE

- [1] Trishala, G., Mamatha, H.R. (2021). Implementation of Stemmer and Lemmatizer for a Low-Resource Language—Kannada. In *Proceedings of International Conference on Intelligent Computing, Information and Control Systems*, pp. 345-358
- [2] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1): 1-47. <https://doi.org/10.1145/505282.505283>
- [3] Lan, M., Tan, C.L., Su, J., Lu, Y. (2008). Supervised and traditional term weighting methods for automatic text categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4): 721-735. <https://doi.org/10.1109/TPAMI.2008.110>
- [4] Debole, F., Sebastiani, F. (2003). Supervised term weighting for automated text categorization. In *Proceedings of the 2003 ACM symposium on Applied computing*, pp. 784-788. <https://dl.acm.org/doi/10.1145/952532.952688>
- [5] Deng, Z.H., Tang, S.W., Yang, D.Q., Li, M.Z.L.Y., Xie, K.Q. (2004). A comparative study on feature weight in text categorization. In *Asia-Pacific Web Conference*, pp. 588-597.
- [6] Quinlan, J.R. (1986). Induction of decision trees. *Machine learning*, 1(1): 81-106. <https://doi.org/10.1023/A%3A1022643204877>
- [7] Altançay, H., Erenel, Z. (2010). Analytical evaluation of term weighting schemes for text categorization. *Pattern Recognition Letters*, 31(11): 1310-1323. <https://doi.org/10.1016/j.patrec.2010.03.012>
- [8] Ren, F.J., Sohrab, M.G. (2013). Class-indexing-based term weighting for automatic text classification. *Information Sciences*, 236: 109-125. <https://doi.org/10.1016/j.ins.2013.02.029>
- [9] Lertnattee, V., Theeramunkong, T. (2004). Analysis of inverse class frequency in centroid-based text classification. In *IEEE International Symposium on Communications and Information Technology, 2004. ISCIT 2004*. Sapporo, Japan. <https://doi.org/10.1109/ISCIT.2004.1413903>
- [10] Luo, Q., Chen, E., Xiong, H. (2011). A semantic term weighting scheme for text categorization. *Expert Systems with Applications*, 38(10): 12708-12716. <https://doi.org/10.1016/j.eswa.2011.04.058>
- [11] Wei, B., Feng, B., He, F., Fu, X. (2011). An extended supervised term weighting method for text categorization. In *Proceedings of the International Conference on Human-centric Computing 2011 and Embedded and Multimedia Computing 2011*, pp. 87-99. https://doi.org/10.1007/978-94-007-2105-0_11
- [12] Wang, Y.A., Chen, Y.N. (2020). What do position embeddings learn? An empirical study of pre-trained language model positional encoding. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6840-6849. <https://doi.org/10.18653/v1/2020.emnlp-main.555>
- [13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.
- [14] Shaw, P., Uszkoreit, J., Vaswani, A. (2018). Self-attention with relative position representations. *North American Chapter of the Association for Computational Linguistics*. <https://doi.org/10.18653/v1%2FN18-2074>
- [15] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, pp. 4171-4186. <https://doi.org/10.18653/v1/N19-1423>
- [16] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. <https://doi.org/10.48550/arXiv.1907.11692>
- [17] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8).
- [18] Shiv, V.L., Quirk, C. (2019). Novel positional encodings to enable tree-based transformers. *Advances in Neural Information Processing Systems*, 32: 12081–12091.
- [19] Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, Australia*, pp. 1243-1252.
- [20] Jayashree, R., Srikantamurthy, K., Anami, B.S. (2015). Sentence level text classification in the Kannada

- language-A classifier's perspective. *International Journal of Computational Vision and Robotics*, 5(3): 254-270. <https://doi.org/10.1504/IJCVR.2015.071335>
- [21] Jayashree, R., Srikanta, M.K., Anami, B.S. (2012). Categorized text document summarization in the Kannada language by sentence ranking. In 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, India, pp. 776-781. <https://doi.org/10.1109/ISDA.2012.6416635>
- [22] Jayashree, R., Srikanta Murthy, K., Anami, B.S. (2013). Performance analysis of naïve Bayesian methods for paragraph level text classification in the Kannada language. In International Conference on Human-Computer Interaction, pp. 435-439.
- [23] Deepamala, N., Kumar, P.R. (2014). Text classification of Kannada webpages using various pre-processing agents. *Advances in Intelligent Informatics*, 235: 235-243. http://dx.doi.org/10.1007/978-3-319-01778-5_24
- [24] Parameswarappa, S., Narayana, V.N., Bharathi, G.N. (2012). A novel approach to build Kannada web Corpus. In 2012 International Conference on Computer Communication and Informatics, Coimbatore, India, pp. 1-6. <https://doi.org/10.1109/ICCCI.2012.6158824>
- [25] Caryappa, B.C., Hulipalled, V.R., Simha, J.B. (2020). Kannada grammar checker using LSTM neural network. In 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, pp. 332-337. <https://doi.org/10.1109/ICSTCEE49637.2020.9277479>
- [26] Gu, E.X. (2021). Convolutional neural network based kannada-mnist classification. In 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, pp. 180-185. <https://doi.org/10.1109/ICCECE51280.2021.9342474>
- [27] Rangan, R.K., Harish, B.S. (2021). Kannada Document classification using unicode term encoding over vector space. *Recent Advances in Artificial Intelligence and Data Engineering*. pp 387-400.
- [28] Chen, K., Zhang, Z., Long, J., Zhang, H. (2016). Turning from TF-IDF to TF-IGM for term weighting in text classification. *Expert Systems with Applications*, 66: 245-260. <https://doi.org/10.1016/j.eswa.2016.09.009>
- [29] Pal, U., Chaudhuri, B.B. (2004). Indian script character recognition: A survey. *Pattern Recognition*, 37(9): 1887-1899. <https://doi.org/10.1016/j.patcog.2004.02.003>