International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 2, April 2022, pp. 1990~2000
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i2.pp1990-2000

Automatic text summarization of konkani texts using pre-trained word embeddings and deep learning

Jovi D'Silva, Uzzal Sharma
Department of Computer Science and Engineering, Assam Don Bosco University, Assam, India

Article Info

Article history:

Received Mar 14, 2021
Revised Nov 25, 2021
Accepted Dec 5, 2021

Keywords:

Automatic text summarization
Deep learning
Fasttext
Konkani
Low resource
Machine learning
Word embeddings

ABSTRACT

Automatic text summarization has gained immense popularity in research. Previously, several methods have been explored for obtaining effective text summarization outcomes. However, most of the work pertains to the most popular languages spoken in the world. Through this paper, we explore the area of extractive automatic text summarization using deep learning approach and apply it to Konkani language, which is a low-resource language as there are limited resources, such as data, tools, speakers and/or experts in Konkani. In the proposed technique, Facebook's fastText pre-trained word embeddings are used to get a vector representation for sentences. Thereafter, deep multi-layer perceptron technique is employed, as a supervised binary classification task for auto-generating summaries using the feature vectors. Using pre-trained fastText word embeddings eliminated the requirement of a large training set and reduced training time. The system generated summaries were evaluated against the 'gold-standard' human generated summaries with recall-oriented understudy for gisting evaluation (ROUGE) toolkit. The results thus obtained showed that performance of the proposed system matched closely to the performance of the human annotators in generating summaries.

Corresponding Author:

Jovi D'Silva
Department of Computer Science and Engineering, Assam Don Bosco University
Tepesia, Assam 782402, India
Email: jovidsilva@gmail.com

1. INTRODUCTION

Text summarization is the process of reducing the contents of a large text document into a concise summary that preserves the key elements and the meaning of the text [1]–[3]. Summarization is vital in providing the readers with a gist of the contents of a larger text document. However, manually summarizing a substantially large number of text documents can be a laborious and time-consuming task, especially when millions of text documents get uploaded to the internet every hour. It is a very cumbersome task for a user to sift through a large volume of files while looking for specific information. Automatically summarizing text documents is very critical under such circumstances, so that a gist of the contents of a document can be provided to the user. A substantial amount of research in natural language processing (NLP) is being done to achieve this task [1]. Automatic text summarization (ATS) is an approach towards developing summaries of text documents automatically to produce compact and meaningful summaries.

We can classify summaries into two broad types, extractive and abstractive [1], [4]–[6]. An extractive summary is generated by extracting relevant sentences or phrases from a text document [1], [4]–[6] An abstractive summary is built by interpreting the meaning and the context of the information conveyed in a document and expressing the content in words or phrases that may differ from those appearing in the original source [1]–[6]. In the domain of ATS, most of the research is being done on the popular and most widely

spoken languages in the world, like English. Through this experiment, we attempt to extend the domain of ATS to Konkani language, which is a low resource language. A low-resource language is the one with limited data, tools, speakers and/or experts in the concerned language. A Konkani language dataset was specially devised to perform these experiments. The dataset comprises of text documents, based on Konkani literature, written in Devanagari script [7]. Konkani language is a minority language spoken in some states along the Indian west coast. It is one of the oldest languages with the earliest found scripture dating back to 1187 A.D [8]. The number of Konkani speaking people has drastically declined since the last few decades, with currently only about 0.19% of the Indian population speaking Konkani. There are approximately 2.2 million Konkani speaking people as per the 2011 Census of India [9]. Given the current status of the language and the need for the proliferation of the language in the online digital realm, the availability of an automatic summarization tool could aid the digital growth of the language and provide readers with a summary of digitized Konkani texts. The primary objective of this research is to expand the domain of ATS to include a low resource language, like Konkani.

The techniques used for summarizing texts can be categorized as language dependent or language independent [10], [11]. Language independent approaches can be extended to more than one language. However, language dependent approaches are only applicable to the language in question. In this work, fastText pre-trained word embeddings are used, which are pre-trained and made available by Facebook in various languages. These embeddings are specific for each language and are, thus, language dependent. fastText is a library for efficiently learning text classifiers and text representations. It is based on the Word2Vec model, which is a shallow neural network that learns word embeddings in an unsupervised manner and generates embeddings for sentences using word embeddings of the constituent words. The models are trained on documents from Common Crawl and Wikipedia to obtain embeddings for words in the languages offered [12], [13]. With Konkani, the number of Konkani language articles is significantly low on Wikipedia compared to popular languages like English [14]. But, fastText can generate embeddings for out of vocabulary words.

In this paper, we perform automatic single document summarization of Konkani texts for producing extractive summaries using a deep learning approach. The methodology comprises two major parts; first, we use fastText's pre-trained Konkani model to generate word embeddings for every word in a sentence of a document. These are then used to generate a sentence embedding to represent the sentence as a feature vector. The second part is applying a deep multi-layer perceptron (MLP) network for performing Automatic Text Summarization. This is done as a supervised binary classification task using the sentences in the source document converted into feature vectors as input to the MLP network [12], [13]. Thus, we have modelled text summarization as a classification task using Deep MLP classifier and examined the use of words embeddings produced by fastText for text summarization. The system output was evaluated using recall-oriented understudy for gisting evaluation (ROUGE) toolkit. It was observed that the output closely matched the summaries produced by the language experts. Our major contributions through this work are summarized: i) this is the first study in a low-resource language, Konkani, to explore a deep learning text classification approach for extractive automatic text summarization of Konkani folk tales; ii) to examine the use of deep learning with fastText as feature vectors for text summarization in Konkani and the usage of fastText word embeddings to construct sentence embeddings, and iii) to evaluate the summarization system using ROUGE-1, ROUGE-2 and ROUGE-L and compare it with a lead baseline and human-annotated benchmark.

The previous work done using deep learning techniques is given in section 2. Section 3 gives the specifications of the dataset used. Section 4 gives a detailed description of multi-layer perceptron network, along with the activation functions used. Section 5 highlights the details of fastText word embeddings model. The detailed step-by-step methodology adopted for the automatic text summarization task is given in section 6. Thereafter, the results of the experiments are provided in section 7, followed by discussions in section 8. The conclusion of the study is presented in section 9.

2. RELATED WORKS

Deep neural networks (DNN) have multi-layer architecture with many hidden layers that help these networks learn. This section highlights some attempts at using deep learning techniques for performing ATS. In 2016, Mani explored ridge regressor and multi-layer perceptron for deriving important features for ATS using DUC 2001 dataset [15]. Sinha *et al*. proposed a data driven approach for extractive text summarization using neural networks with DUC dataset [16]. In the proposed model, they used a neural network with an input layer, one hidden layer, and an output layer. Facebook's fastText library was used for representing sentences as vectors. The model received English language sentences as input, along with the vector representation of words. A source document was divided into segments and then these segments were fed to the system recursively to get the document summary.

In the work presented by Mastronardo in 2019, pre-trained "deep conceptualized word embeddings" were used with pointer generator models for performing text summarization on the newsroom and CNN/Daily Mail datasets. The evaluation of the outcome using recall-oriented understudy for gisting evaluation (ROUGE) showcased impressive performance [17]. Suleiman and Awajan implemented deep learning techniques for performing text summarization on DUC2002 and Daily Mail datasets. They used Word2Vec model for representing words and Facebook's fastText library. ROUGE evaluation metrics were used for the assessment of the system performance [18].

Lwin and Nwet proposed using fastText sentence embeddings in conjunction with centroid based model for the summarization of Myanmar news articles. The positives of using fastText is that it presents vector representations of the words that are absent from a dictionary [19]. Nitsche and Halbritter presented comparative results of the study of various neural network constructs in performing text classification on a German language dataset. In the experiments, they used Facebook's fastText pre-trained word vectors and noticed a 1% to 2% drop in the accuracy when embeddings were not used [20]. Pittaras and Karkaletsis trained word embeddings from ground up, and also, evaluated the performance of Facebook's fastText pre-trained embeddings for extractive text summarization [21].

Lamsiyah *et al.* proposed using DNN for extractive single document summarization with sentence embedding. They presented the performance evaluations of employing three sentence embedding models on DUC 2002 dataset for ATS and show that deep learning methods display exceptional output [22]. Dudchenko and Kopanitsa compared the performance of various artificial neural network models for extracting text from Russian medical text documents with the help of word embeddings [23]. They compared multi-layer perceptron (MLP), convolutional neural networks (CNN) and long short-term memory networks (LSTMs); MLP and CNN presented similar performance with the embedding models. However, the performance of MLP exceeded that of CNN on the pipelines which used the embeddings trained using medical records with lemmatization but CNN achieved highest F-score.

Another system, presented by Alghamdi and Assiri, uses fastText word embeddings for performing Arabic text classification. The study compared two algorithms of fastText's Python implementations, viz., Gensim's native implementation and Facebook's official fastText in performing classification of text [24]. Afzal *et al.* proposed using deep neural network model for the summarization of biomedical texts. The objective of the work was to verify the performance of deep neural networks and recurrent networks in the field of ATS since these are previously known to outperform machine learning models in NLP tasks [25]. The results of these experiments indeed indicated that the deep neural network models performance showed greater accuracy. Suleiman *et al.* presented a technique for text summarization using deep learning and used ROUGE metric for the quality assessment [26]. Hailu *et al.* proposed ATS system and an automatic evaluation metric based on Word embeddings. They used pre-trained word embedding models Word2Vec, GloVe and fastText [27]. Li and Gong, in their work, have presented the use of word embeddings and compared the accuracy of various deep learning models for text classification [28]. Of the various deep learning models used in the experiments, MLP proved to be relatively simple in terms of its structure and yet presented respectable performance.

We observe that most of the related works have effectively used deep learning architectures like MLP, CNN and LSTM for text classification and ATS. Also, it can be seen that English is the language that is most preferred for such research compared to other languages. They have also shown the indispensable use of word embeddings, such as fastText, Word2Vec, that have contributed to making deep learning successful for these tasks. Since fastText word embeddings are available in various languages, enabling the use of these word embeddings in deep learning for languages other than English such as German, Arabic. fastText also makes available word embeddings for some low-resource languages like Konkani, thus, enabling us to use word embeddings along with deep learning approaches, such as MLPs, to perform text summarization modelled as a text classification task. In our experiment, we compute sentence embeddings, using fastText word embeddings, which act as feature vectors ideal to be used with MLPs [29]. Previous work using machine learning for text summarization in Konkani used k-means clustering on the same Konkani dataset [30]. We compare this system with the system presented in this paper. The details of the dataset used are presented in the next section.

3. THE DATASET

The Konkani language dataset used for the experiments was developed by the authors specifically to facilitate research in Konkani in the ATS domain [7]. Konkani language is spoken along the west coast of India and has different dialects. The version of Konkani on which the dataset is based is primarily spoken in the state of Goa, which is a small state in India. The number of Konkani speaking population has reduced drastically in the recent years. According to 2011 Census of India, there are only about 2.2 million Konkani

speakers in India [9]. This work is an attempt to bring Konkani language to the forefront of research in the domain of ATS.

The dataset comprises 71 folk tales procured from 5 books on Konkani literature, written in Devanagari Script. These books have folk tales from various geographical locations of India, Russia and also, Goa. Some books have stories that were translated from other languages to Konkani by the authors of the books. The books obtained for the creation of this dataset are rare and were difficult to find as their copies are limited. Our dataset essentially captures the crux of Konkani literature. The details of the dataset generation process are given in [7].

The average number of sentences in each story is 138, and each story has about 1,155 words on an average with the shortest story having 520 words and the longest story having 2,852 words. The dataset comprises 9,849 sentences in total. For each of the 71 folktales, an abstract of 300 words length was generated by two autonomous subject specialists. Ideally, a 300 words document would make up for one page and seemed like a suitable word count threshold to be applied to the length of the resulting summaries in addition, considering the length of the shortest story being 520 words, we had to set the standard length for the human-generated summaries to a value less than 520, therefore, the word limit was set to 300, and in consultation with the language experts, 300 words limit was found to be an ideal value. These abstracts served as the gold-standards for the system generated summary evaluations.

4. MULTI-LAYER PERCEPTRON NETWORK

A perceptron is a very basic learning mechanism. It can accept a couple of inputs, each with a weight to indicate its importance, and produces a decision output of 0 or 1. In a multi-layer perceptron model, there are several perceptrons assembled in layers to perform complex computations [31]. In the system developed for ATS in this paper, the first layer of MLP network is the input layer with 300 nodes, followed by two successive hidden layers as depicted in Figure 1. The first hidden layer consists of 300 nodes and the second hidden layer comprises 200 nodes, with the output layer having only a single node. The reason behind using 300 nodes at the input layer is that fastText has a vector length of 300 for each word. Likewise, each word in a sentence has a vector of length 300 and when all the values of each word vector in a sentence are combined, we obtain a sentence vector of length 300. Each of these values is then given as input to each of the 300 input nodes of the network. These vector values form a tabular structure with 300 vector values represented along the x-axis and the sentences represented along the y-axis as rows. There is an additional value in this tabular structure that denotes the class to which a sentence belongs and this is used for training the classifier. MLP model works well with such tabular structures. MLP network has a relatively simpler structure but yet obtains respectable results. The simpler structure has a direct impact on the time and space complexity of a system [23], [28].

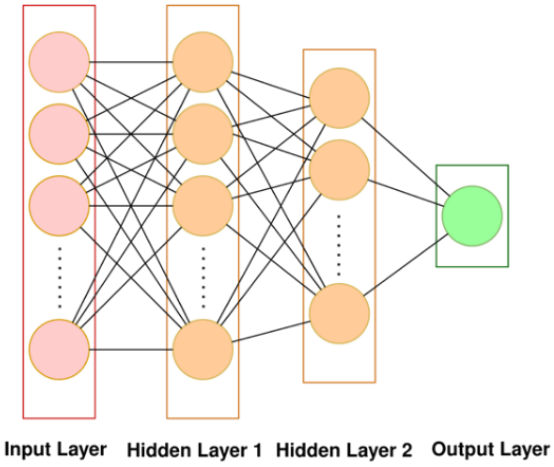

Figure 1. MLP architecture

The optimizer used in this experiment was Adam [32] and the loss function used was *binary_crossentropy*. For the model run, a batch size of 100 was used with 2 training epochs. This construction is a deep neural network because of the use of more than one hidden layers. The hidden layers make use of rectified linear units (ReLU) activation functions and the final output layer makes use of a

sigmoid activation function, as this is a binary classification task. The model was implemented with Keras and TensorFlow.

4.1. Activation functions

4.1.1. Rectified linear unit function

Rectified linear unit (ReLU) produces a basic non-linear transformation. If an element y is presented to the function, then the function is illustrated as the maximum of the element y and zero, given by (1). Positive elements are preserved in ReLU and all the negative elements are discarded by fixing the corresponding activations to 0. When a negative input is supplied, ReLU function produces a 0, and when a positive input is supplied, it outputs a 1 [33].

$$ReLU\ (y) = \max(y, 0) \quad (1)$$

4.1.2. Sigmoid function

The sigmoid function converts the input values supplied to it to output values ranging between 0 and 1, as depicted in (2) [16].

$$Sigmoid\ (y) = \frac{1}{1+\exp(-y)} \quad (2)$$

5. FASTTEXT WORD EMBEDDINGS

The fastText can produce vectors for words which are of a fixed 300 dimension, but it needs a very large dataset and lots of computation power in order to train and work. Fortunately, fastText provides pre-trained word vectors in 157 languages, including the language Konkani, these models have been trained on Common Crawl and Wikipedia [12]. The models were trained using continuous bag of words (CBOW) model, as depicted in Figure 2, with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives.

CBOW model predicts a target word as per its context. A fixed size window around the target word indicates the context in a bag of words representation. To illustrate the working of the CBOW model and how it predicts the target word with a window size of 5, consider the following sentence, "त्या गांवांत एक नुस्तेंकार आसलेलो" (There was a fisherman in that village). Consider 'एक' as the target word. Then, two words on either side of the target word along with the target form the window size of 5. Thus, the vectors of the context words (त्या, गांवांत, नुस्तेंकार, आसलेलो) are then used to predict the target word 'एक'.

The fastText pre-trained word embeddings can also predict out of vocabulary (OOV) word vectors it has not seen during the training phase. This has a huge implication for this work, since fastText was trained on a different dataset before being made available for download. fastText can do so by building a vector by summing the sub-word vectors which make up the OOV word. It can also generate better word embeddings when it encounters rare words because it considers the internal structure of words. These aspects of fastText are beneficial in this experiment.

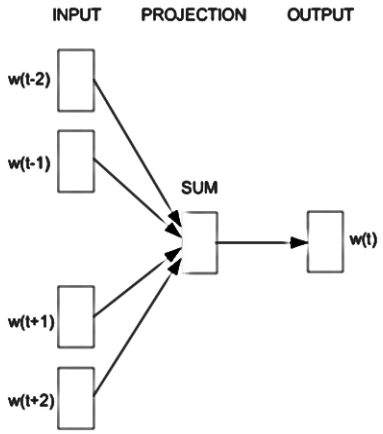

Figure 2. Continuous bag of words (CBOW) architecture

6. METHOD

In this section, the details of the proposed methodology are presented. Figure 3 illustrates the proposed method followed by the detailed description of the steps involved.

a) Pre-processing: This step involves placement of every sentence on a new line. Thereafter, the sentences are cleaned of the punctuation marks. The punctuations will, however, be preserved in the system generated summary. Each word is separated to compute word embeddings.

b) Compute sentence embeddings: The 300-dimensional fastText word embedding for each word in a sentence is retrieved and then L2 norm is calculated for each of the words. L2 norm is "a standard method to compute the length of a vector in Euclidean space". Given $x = [x1\ x2\ \dots\ xn]^T$, L2 norm of x is defined as "the square root of the sum of the squares of the values in each dimension" [34]. The word embeddings that have a positive L2 norm are then divided by the L2 norm and then averaged to generate a sentence embedding for that sentence which acts as a feature vector.

c) Classification with K-fold cross validation: The dataset comprises of 71 folk tales and it was divided into 5 folds. In each fold that was created, 80% of the fold was used for training and 20% was used for testing, as depicted in Figure 4. Five equal folds of 71 stories could not be obtained; hence, the testing batch composed of 14 stories for the first four folds and 15 stories for the last fold. In this manner, every document in the dataset partakes in the training and the testing phases. During the processing of each fold, MLP model is trained on the training data and is then applied to predict the class of the test samples provided to it. Thus, for each document in the test partition, predictions are made for each sentence; either it is 1 or 0. The sentences predicted to be significant i.e., the ones marked as 1 are considered for inclusion in the final summary.

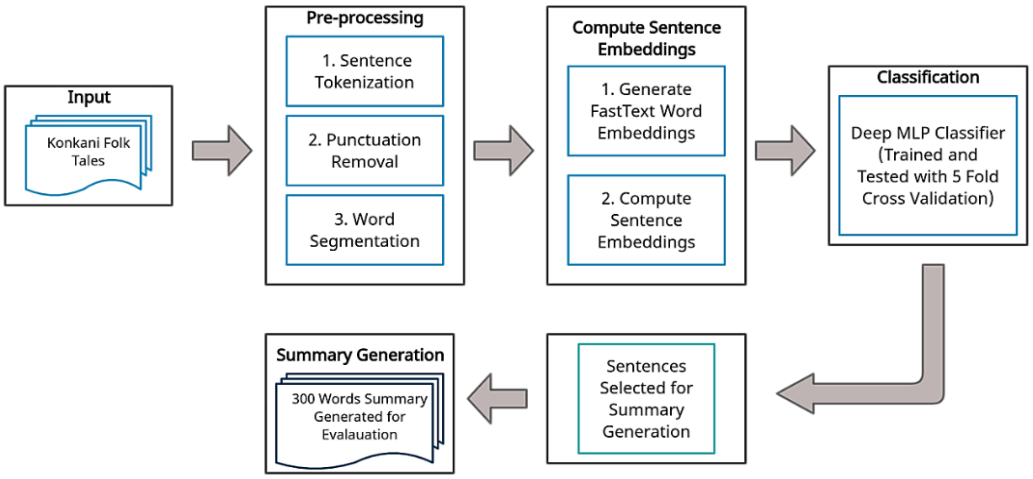

Figure 3. Proposed methodology for automatic text summarization using deep MLP

Figure 4. K-folds

d) Summary generation: A threshold of 300 words is applied to restrict the length of the generated summary. The value of the threshold is set to 300 for the purpose of evaluation since the length of human generated summaries is 300 words. Each story is summarized by two language experts and these are then compared with the system generated summaries with the help of the ROUGE toolkit to evaluate content overlap.

7. RESULTS

We have performed single document extractive summarization on a Konkani language dataset, consisting of 71 documents, using Deep MLP classifier with the help of pre-trained fastText word embeddings that represent sentences as feature vectors. To evaluate the system summaries generated, we employ the ROUGE toolkit. ROUGE has been found to work precisely with human assessments and adopts N-gram statistics [35]. The 300 words system generated extractive summary was compared with the two reference abstracts generated by two language experts [35]. ROUGE metric was used to quantify the overlap of uni-grams, bi-grams and longest common subsequence (LCS) between the human generated reference summaries and the auto-generated summary. ROUGE-1 depicts uni-gram scores, ROUGE-2 denotes bi-gram scores, and ROUGE-L mirrors LCS scores. The perception behind using the above measures was to verify the eloquence of the summaries generated, with growing levels of granularity shifting through uni-gram to bi-gram and then to LCS. If the arrangement of words in the resultant summary is akin to the human-summarized reference summaries, it is an implication of a rather eloquent system generated output. Specifics of the working of the evaluation measures employed are underlined by Lin [35]. Tables 1, 2 and 3, outline the metrics of the assessment of the corresponding human summaries with the system-generated summaries using deep learning. These tables also provide the scores of automatic text summarization system built with k-means clustering with 3 clusters with the output summaries generated with the same Konkani folk tales dataset provided as input to the system [30]. The performance of ATS systems can be compared against baseline systems, like using leading sentences from the input text document [2]. We have included the scores of lead baseline in the above three tables for comparison with the performance of our system. A lead baseline summary is constructed using the first 300 words from an input document. Also, a new benchmark is introduced which consists of 300 words extractive summaries constructed using sentences that are manually indicated as relevant by the language experts for inclusion in a summary. The system generated summaries and the human-annotated benchmark are extractive summaries which are compared with the pair of gold-standard summaries.

Table 1 shows ROUGE-1 (uni-gram) scores. Table 2 gives an account of ROUGE-2 (bi-gram) and Table 3 gives ROUGE-L (LCS) scores. The values of ROUGE-1 scores are higher compared to ROUGE-2 and ROUGE-LCS scores. The reason for this is that a matching term is in both ROUGE-1 and ROUGE-2 but this is not true the other way round. However, ROUGE-L values are greater than ROUGE-2 values since ROUGE-L keeps track of word matches that are in-sequence rather than checking for consecutive word matches. ROUGE-L naturally integrates longest common n-grams appearing in a sequence.

In the tables displayed below, the equivalent ROUGE metrics are estimated, where the quantitative interpretation of the overlap between the system-generated summary and the human-generated summaries are provided by Precision and Recall. Precision aims to determine to what extent the contents of a system generated summary are relevant. This is critical, as a generated summary may contain unimportant portions of the original text. Recall intends to analyse to what extent the content of the gold-standard human-generated reference summary was adequately captured by the automatically generated summary. The F-Score takes into consideration both, Precision and Recall, and thereafter, provides a combined report of the two measures as a unique score. The scores obtained lie within the range of 0 to 1. A score of 0 implies no overlap between the system-generated summary and human-generated reference summary, whereas a score of 1 suggests a very strong overlap between the two summaries. The graphical representations of the comparisons of Precision, Recall and F-Score of supervised fastText Deep MLP, k-means clustering with 3 clusters, lead baseline and human annotated benchmark are given below in Figures 5, 6, and 7 respectively. As seen from the tables and Figures 5, 6, and 7, the F-Scores of the system closely match the F-Scores of the benchmark.

Table 1. ROUGE-1 uni-gram scores

	ROUGE-1 (uni-gram)		
Systems	Precision	Recall	F-score
Supervised-FastTextDeepMLP	0.33589	0.32691	0.33054
K-Means-3Clusters	0.31408	0.31373	0.31388
Lead-Baseline300words	0.30147	0.30165	0.30154
HumanAnnotated-Benchmark	0.35844	0.35460	0.35608

Table 2. ROUGE-2 (bi-gram) scores

ROUGE-2 (bi-gram)			
Systems	Precision	Recall	F-score
Supervised-FastTextDeepMLP	0.09559	0.09247	0.09372
K-Means-3Clusters	0.07942	0.07927	0.07934
Lead-Baseline300words	0.08097	0.08103	0.08099
HumanAnnotated-Benchmark	0.11088	0.10908	0.10977

Table 3. ROUGE-L (LCS) scores

ROUGE-L (LCS)			
Systems	Precision	Recall	F-Score
Supervised-FastTextDeepMLP	0.32952	0.32064	0.32423
K-Means-3Clusters	0.30680	0.30644	0.30659
Lead-Baseline300words	0.29642	0.29659	0.29648
HumanAnnotated-Benchmark	0.35228	0.34847	0.34994

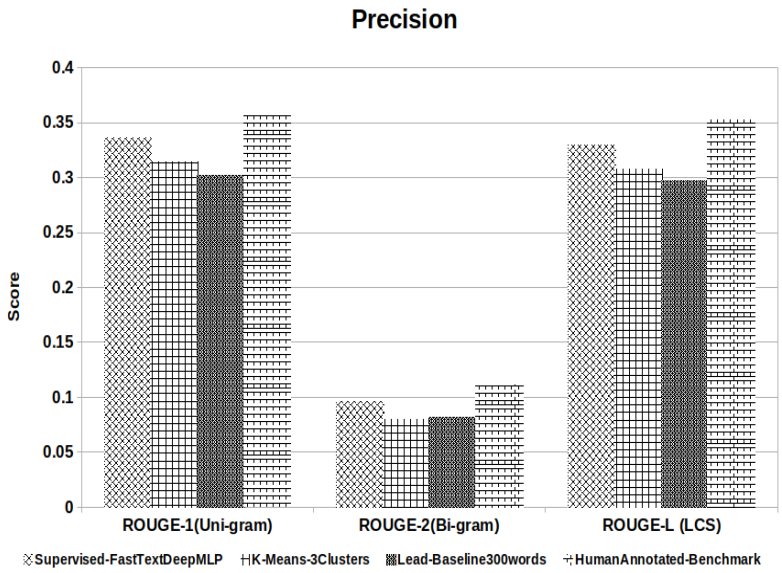

Figure 5. Precision for ROUGE-1, ROUGE-2 and ROUGE-L

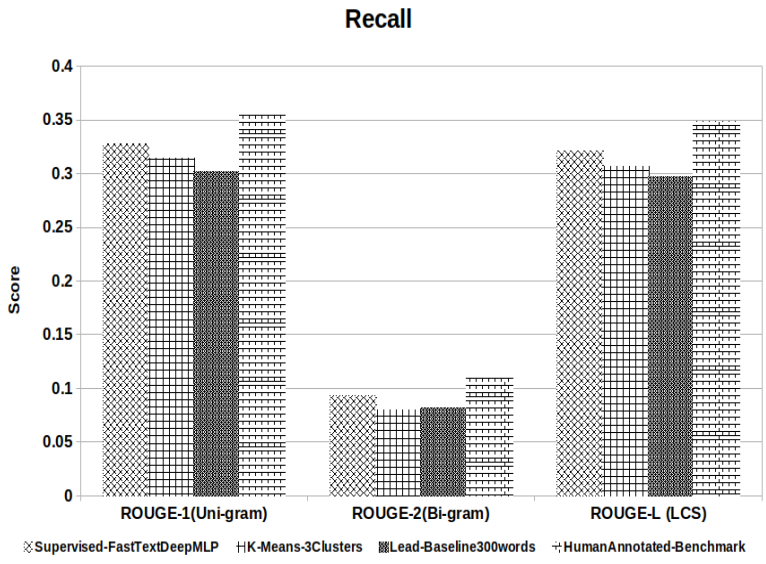

Figure 6. Recall for ROUGE-1, ROUGE-2 and ROUGE-L

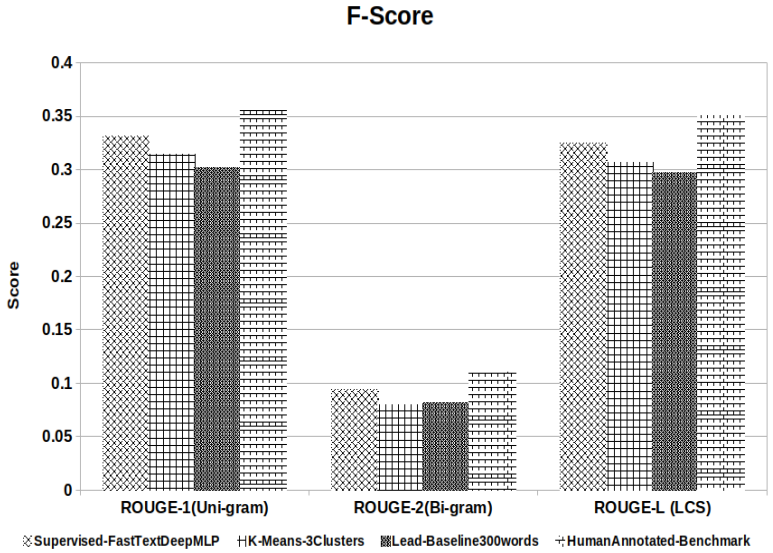

Figure 7. F-score for ROUGE-1, ROUGE-2 and ROUGE-L

8. DISCUSSION

The fastText based DeepMLP system can select important sentences to form summaries, as seen in the F-Scores reported by ROUGE. Supervised techniques, such as our system, require training data in order to learn how to produce a good summary. In case of deep learning, the size requirement of this training data is much greater than traditional machine learning approaches. Another dataset would have been required to learn word embeddings alone, but with fastText, this need was eliminated as it was pre-trained and available for download. This was possible since fastText was trained on Konkani documents from Common Crawl and Wikipedia. fastText offers pre-trained word embeddings in a total 157 languages [12]. Using fastText saves training time and eliminates the need for a separate dataset for learning word embeddings. Konkani, being a low resource language, having limited tools, data, speakers or language experts, not needing another dataset for training word embeddings proved to be an advantage. The articles on which the pre-trained word embeddings were trained were from different fields, despite this, fastText takes into account the morphological structure of a word, unlike other word embeddings like Word2Vec. fastText can produce vectors even for out-of-vocabulary words, which is a big advantage in case of our experiment. The system presented in this paper, being a supervised approach, showed better performance as compared to unsupervised extractive technique based on k-means clustering on the same Konkani dataset [30]. Therefore, this indicates that supervised deep MLP technique can produce better quality summaries in comparison with unsupervised approaches. The system also beat the Lead Baseline of 300 words.

This research can be further expanded by increasing the size of the summarization dataset available to the MLP classifier. Another consideration is using the same domain dataset for training the pre-trained word embeddings as the domain of the summarization dataset. There are a limited number of articles on Wikipedia that were used for training the pre-trained word embeddings for low-resource languages like Konkani, compared to the number of articles available for popular languages like English [14]. Using a larger corpus for training these word embeddings can be considered as another avenue for research. Also, comparison of fastText word embeddings with other classifiers can also be explored.

9. CONCLUSION

In this paper, automatic single document extractive summarization of Konkani texts using Facebook's fastText pre-trained word embeddings and deep MLP. One of the main challenges of using deep learning is the requirement of a large dataset, but since fastText pre-trained models are distributed in many languages, they allow us to save training time in generating word embeddings and also eliminate the need for a large training corpus. In addition, fastText can generate an embedding for a word that was unseen during the training phase. Word embeddings have been a key breakthrough in NLP and Deep Learning, allowing an efficient representation of the words in a document, and as is our case, the sentences in a document. Using these sentence embeddings as input features to deep MLP displayed very impressive results, demonstrating that the system's performance was comparable to the human annotated benchmark. This research lays the

foundation for other research in low-resource languages such as Konkani, although there is work on cross-lingual word embedding models. A majority of pre-trained word embedding models support only popular languages. There a need for more pre-trained models in other low resource languages given that pre-trained models provide impressive results when used as feature vectors in a classification task. Another point of significance is that despite the words embeddings being trained on data from different sources, i.e., Wikipedia and Common Crawl, and different domain from that of the dataset, the word embeddings provided excellent results. In future work, examining the effects of increasing the size of training data for the model could be considered. Another consideration that can be considered is using data from the same domain as the training data to train the fastText shallow neural network and generate word embeddings for the same.

REFERENCES

[1] N. Moratanch and S. Chitrakala, "A survey on extractive text summarization," in *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, 2017, pp. 1–6, doi: 10.1109/ICCCSP.2017.7944061.

[2] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic text summarization: A comprehensive survey," *Expert Systems with Applications*, vol. 165, p. 113679, Mar. 2021, doi: 10.1016/j.eswa.2020.113679.

[3] D. Jurafsky and James Martin, *Question answering and summarization*, 2nd ed. Pearson Education India, 2013.

[4] E. Lloret and M. Palomar, "Text summarisation in progress: a literature review," *Artificial Intelligence Review*, vol. 37, no. 1, pp. 1–41, Jan. 2012, doi: 10.1007/s10462-011-9216-z.

[5] C. Khatri, G. Singh, and N. Parikh, "Abstractive and extractive text summarization using document context vector and recurrent neural networks," *Computation and Language*, Jul. 2018.

[6] N. Andhale and L. A. Bewoor, "An overview of text summarization techniques," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, 2016, pp. 1–7, doi: 10.1109/ICCUBEA.2016.7860024.

[7] J. D'Silva and U. Sharma, "Development of a Konkani language dataset for automatic text summarization and its challenges," *International Journal of Engineering Research and Technology*, vol. 12, no. 10, pp. 1813–1817, 2019.

[8] K. Kamat, "Konkani Heritage Album: The origins of the Konkani language," Kamat's Potpourri. [Online]. Available: http://www.kamat.com/kalranga/konkani/konkani.htm (Accessed Feb 15, 2021).

[9] Office of the Registrar General & Census Commissioner India, "Distribution of population by Scheduled and other Languages India, States and Union Territories-2011," *Data on Language and Mother Tongue*, Ministry of Home Affairs, Government of India. 2011. [Online]. Available: https://censusindia.gov.in/2011Census/Language_MTs.html (Accessed: 15-Nov-2020).

[10] J. D'silva and U. Sharma, "Automatic text summarization of indian languages: a multilingual problem," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 11, pp. 3026–3037, 2019.

[11] O. Gross, A. Doucet, and H. Toivonen, "Language-independent multi-document text summarization with document-specific word associations," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 853–860, doi: 10.1145/2851613.2851647.

[12] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2016, pp. 427–431.

[14] Wikimedia, "List of Wikipedias," Wikimedia Meta-Wiki, 2001. [Online]. Available: https://meta.wikimedia.org/wiki/List_of_Wikipedias#1_000+_articles (Accessed Feb 15, 2021).

[15] K. B. Mani, "Text summarization using deep learning and ridge regression," Unpublished, Dec. 2016, ArXiv abs/1612.08333.

[16] A. Sinha, A. Yadav, and A. Gahlot, "Extractive text summarization using neural networks," arXiv preprint, Feb. 2018, arXiv:1802.10137.

[17] C. Mastronardo, "Integrating deep contextualized word embeddings into text summarization systems," Thesis, Universita di Bologna, 2019.

[18] D. Suleiman and A. A. Awajan, "Deep learning based extractive text summarization: approaches, datasets and evaluation measures," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019, pp. 204–210, doi: 10.1109/SNAMS.2019.8931813.

[19] S. S. Lwin and K. T. Nwet, "Extractive Myanmar news summarization using centroid based word embedding," in *2019 International Conference on Advanced Information Technologies (ICAIT)*, 2019, pp. 200–205, doi: 10.1109/AITC.2019.8921386.

[20] M. Nitsche and S. Halbritter, "A comparison of neural document classification models," Unpublished, 2019.

[21] N. Pittaras and V. Karkaletsis, "A study of semantic augmentation of word embeddings for extractive summarization," in *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources associated with RANLP 2019*, 2019, pp. 63–72, doi: 10.26615/978-954-452-058-8_009.

[22] S. Lamsiyah, A. El Mahdaouy, S. O. El Alaoui, and B. Espinasse, "A supervised method for extractive single document summarization based on sentence embeddings and neural networks," in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2019)*, 2020, pp. 75–88.

[23] A. Dudchenko and G. Kopanitsa, "Comparison of word embeddings for extraction from medical records," *International Journal of Environmental Research and Public Health*, vol. 16, no. 22, p. 4360, Nov. 2019, doi: 10.3390/ijerph16224360.

[24] N. Alghamdi and F. Assiri, "A comparison of fasttext implementations using Arabic text classification," in *Intelligent Systems and Applications*, Y. Bi, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2020.

[25] M. Afzal, F. Alam, K. M. Malik, and G. M. Malik, "Clinical context–aware biomedical text summarization using deep neural network: model development and validation," *Journal of Medical Internet Research*, vol. 22, no. 10, p. e19810, Oct. 2020, doi: 10.2196/19810.

[26] D. Suleiman and A. Awajan, "Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–29, Aug. 2020, doi: 10.1155/2020/9365340.

[27] T. T. Hailu, J. Yu, and T. G. Fantaye, "A framework for word embedding based automatic text summarization and evaluation," *Information*, vol. 11, no. 2, p. 78, Jan. 2020, doi: 10.3390/info11020078.

[28] S. Li and B. Gong, "Word embedding and text classification based on deep learning methods," *MATEC Web of Conferences*, vol. 336, p. 06022, Feb. 2021, doi: 10.1051/matecconf/202133606022.

[29] F. Chollet, *Deep learning with Python*. New York: Manning Publications, 2017.
[30] J. D'Silva and U. Sharma, "Unsupervised automatic text summarization of Konkani texts using K-means with Elbow Method," *International Journal of Engineering Research and Technology*, vol. 13, no. 9, p. 2380, Sep. 2020, doi: 10.37624/IJERT/13.9.2020.2380-2384.
[31] E. B. Baum, "On the capabilities of multilayer perceptrons," *Journal of Complexity*, vol. 4, no. 3, pp. 193–215, Sep. 1988, doi: 10.1016/0885-064X(88)90020-9.
[32] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," in *Published as a conference paper at ICLR 2015*, 2015, doi: 1412.6980.
[33] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," 2021, pp. 203–224.
[34] "L2 norm," in *Encyclopedia of Biometrics*, Boston, MA: Springer US, 2009, pp. 883–883.
[35] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004, pp. 74–81.

BIOGRAPHIES OF AUTHORS

Jovi D'Silva is presently a research scholar in the Department of Computer Science and Engineering, School of Engineering, Assam Don Bosco University, Guwahati, India. He has obtained BCA and MCA degrees from Bangalore University, India and M.Tech. in Computer Science and Engineering from Christ University, India. His research area is Natural Language Processing. He can be reached through email jovidsilva@gmail.com.

Uzzal Sharma has obtained his MCA from IGNOU and completed PhD from Gauhati University. He has over 16 years of experience in the field of academics and industry. His research areas include Speech Signal Processing, Software Engineering, Cyber Security and Data Engineering. He has produced two PhD scholar and currently guiding 4 research scholars. He has also guided many M.Tech, B.Tech and MCA students in various domains. He has published more than 25 research journals and conference proceedings. He also has 11 book chapters to his credit in edited book. He has also published 5 books as the sole author. Currently, he is an Assistant Professor- Stage 2 at Assam Don Bosco University, Guwahati, India. He can be reached through email uzzal.sharma@dbuniversity.ac.in.