Abstractive Hindi Text Summarization: A Challenge in a Low-Resource Setting

Daisy Monika Lal and **Paul Rayson**
Lancaster University,
Lancaster, United Kingdom
{d.m.lal, p.rayson}@lancaster.ac.uk

Krishna Pratap Singh and **Uma Shanker Tiwary**
IIIT Allahabad, Prayagraj, India
{kpsingh, ust}@iiita.ac.in

Abstract

The Internet has led to a surge in text data in Indian languages; hence, text summarization tools have become essential for information retrieval. Due to a lack of data resources, prevailing summarizing systems in Indian languages have been primarily dependent on and derived from English text summarization approaches. Despite Hindi being the most widely spoken language in India, progress in Hindi summarization is being delayed due to the lack of proper labeled datasets. In this preliminary work we address two major challenges in abstractive Hindi text summarization: creating Hindi language summaries and assessing the efficacy of the produced summaries. Since transfer learning (TL) has shown to be effective in low-resource settings, in order to assess the effectiveness of TL-based approach for summarizing Hindi text, we perform a comparative analysis using three encoder-decoder models: attention-based (BASE), multi-level (MED), and TL-based model (RETRAIN). In relation to the second challenge, we introduce the ICE-H evaluation metric based on the ICE metric for assessing English language summaries. The Rouge and ICE-H metrics are used for evaluating the BASE, MED, and RETRAIN models. According to the Rouge results, the RETRAIN model produces slightly better abstracts than the BASE and MED models for 20k and 100k training samples. The ICE-H metric, on the other hand, produces inconclusive results, which may be attributed to the limitations of existing Hindi NLP resources, such as word embeddings and POS taggers.

1 Introduction

Automatic text summarization is the process of condensing lengthy text into a concise version that captures and relays important information. The amount of text data created has increased tremendously in the era of the internet, and while statistical approaches were used in earlier summarization methods, deep learning models have emerged as a viable and promising solution. There are two main approaches in text summarization: extractive (ETS) and abstractive (ATS). To create a summary, ETS combines already-written sentences without any alterations while ATS involves text generation. Essentially, the ATS writes its own sentences, similar to how humans summarize text, and is hence preferred over ETS (Mehta, 2016; Gupta and Gupta, 2019). ATS, as it involves text generation, requires massive volumes of data to efficiently train a summarizer. Training models for any NLP task is extremely challenging in the low-resource situation, when textual input is either limited, unannotated, inaccessible, or lacking linguistic resources such as lemmatizers, grammar and spelling checkers, part-of-speech taggers, trained word embeddings, etc. With the increase in text data in Indian languages due to advent of the Internet, text summarization tools for efficiently searching and retrieving information have become imperative. Due to a paucity of data resources, prevailing ATS systems in Indian languages have been primarily dependent on and derived from English text summarization approaches.

Hindi is, along with English, one of the 22 official languages of India. There are about 615 million native Hindi speakers, making it the third most spoken language in the world. A majority of Indians use Hindi as their main language. The morphologically rich indigenous language is a reservoir of ancient wisdom, composed of folklores, songs, poems, chronicles, etc. (Gopalakrishnan, 2009). Though Hindi is the top-most language used in India, there is a lack of a proper summarization system for Hindi text. The prolonged advancements in abstractive Hindi summarization are owed to the absence of labeled Hindi summarization datasets. Even the existing annotated corpora are not sufficiently large enough to train a Sequence-to-Sequence

(Seq2Seq) (Sutskever et al., 2014a,b) model from scratch. Seq2Seq models are loaded with massive weights to be optimized during training. When trained on very few data samples, these models fail to optimize the weights and prohibit any learning, whereas, on small to medium-sized corpora, the models often tend to over-fit. Apparently, in the scenario when data is scarce for independently training a deep neural network, transfer learning or TL-based approaches have proved to succeed (Gehrmann et al., 2019; Fecht et al., 2019; Zolotareva et al., 2020; Chen and Shuai, 2021; Alomari et al., 2022).

The primary contributions of this work include:

- **Multi-level shared weight encoding strategy:** Based on the model introduced by (Lal et al., 2022b), which uses multiple levels of encoding of the input sequence for capturing the underlying context more accurately, we experiment with a similar approach on a low-resource Hindi corpus. The intent was to investigate the effects of rereading the input text in the low-resource setting. We train an autonomous model on Hindi corpora and study its effectiveness for training data randomly sampled at different sizes (5k, 20k, and 100k).
- **Transfer learning strategy:** The TL-based approach uses pre-trained weights from a model trained on English Gigaword corpus. This approach fine-tunes on the new Hindi corpus randomly sampled at different sizes (5k, 20k, and 100k).
- **Evaluation strategy**: We introduce ICE-H (Information Coverage Estimate for Hindi Abstracts), a variant of the ICE Metric (Lal et al., 2022a) for scoring the machine-generated abstractive Hindi summaries. We pinpoint the challenges encountered in devising the metric and propose feasible solutions.

2 Related Work

2.1 Structure-based Approach

The structure-based practice involves the congregation of salient sentences into a preset structure (Lee et al., 2005; Saranyamol and Sindhu, 2014). (Embar et al., 2013) employed a combination approach involving part-of-speech tagging, stemming, named-entities identification, information retrieval, and abstraction routines for Kannada text summarization. (Kallimani et al., 2016) implement a template-based summarization strategy for administering content-aware information-intensive abstracts. The primary step of key content classification is realized using a sentence-scoring mechanism, followed by, root-word identification to facilitate template-based sentence generation.

2.2 Semantic-based Approach

The semantic-based practice involves developing a semantic blueprint of the input document to be summarized. These semantic structures are delivered to a Natural Language Generation (NLG) model for generating the output summary. Semantic graphs, multimodal-semantic strategies, and information-item-based techniques are typical semantic-based practices for generating abstractive summaries (Kabeer and Idicula, 2014). (Sinha and Jha, 2020) highlight the intricacies associated with Sanskrit prose and propose that semantic representations, such as rich semantic graphs are indispensable for Sanskrit text summarization due to the vivid morphological form of the language. Similar to (Sinha and Jha, 2020), (Yeasmin et al., 2017) also recommend semantic-graphs over ontology-based or rule-based strategies. They also suggest that the construction of domain ontologies and Bengali Wordnet are essential prerequisites for semantic-based Bengali text summarization.

2.3 Supervised Approach

(Nambiar et al., 2021b,a) bring to light the unique complexities of the Malayalam language involving lack of predicate agreements between subject and verbs w.r.t person, gender, and number; and extensive use of linking verbs or copulas in sentences. The traditional seq2seq attention model is trained on data constructed by translating freely available BBC news corpora into Malayalam. (Karmakar et al., 2021) employ two frameworks featuring Stacked-Lstm encoders, namely, Time-distributed Stacked-Lstm and Attention-based Stacked-Lstm models. Specifically for training the Hindi and Marathi ATS frameworks, data was manually curated from several news websites and online platforms. (Bharath et al., 2022) proposed a seq2seq architecture trained on 2000 news articles from various Telugu News websites. The model also harnesses the benefits of pre-trained FastText word embeddings (Young and Rusli, 2019). (Talukder et al., 2019) train a seq2seq pipeline consisting of a

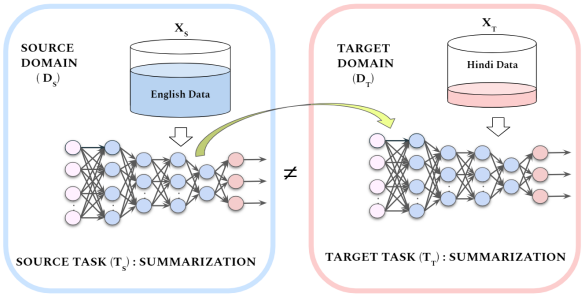

Figure 1: TL-based Hindi ATS Model. Here, X_S denotes the source feature space, X_T denotes the target feature space, D_S, and D_T denote the source and target domains, respectively, T_S and T_T denote the source and target tasks, respectively. The model is a scenario of TL where the tasks are the same ($T_S = T_T$), but feature spaces are different ($X_S \neq X_T$).

bidirectional-LSTM encoder and attentive decoder model for Bengali text summarization. The data consists of Bengali texts assembled from social media posts, news articles, etc.

2.3.1 Unsupervised Approach

(Chowdhury et al., 2021) construct an unsupervised Bengali ATS framework backed up by a pre-trained Bengali text language generation model. The model entails the creation of sentence-level directed graphs where words and corresponding part-of-speech tags form the vertices with directed edges between adjacent words.

3 Proposed Model

3.1 TL-based Abstractive Hindi Text Summarization

The key idea is first to train a high-resource language pair (the parent model), then transfer some of the learned parameters to the low-resource pair (the child model), as shown in Figure 1. The proposed model comprises two phases: (a) Pre-training Phase: In this phase, the model is trained on parent data, i.e., English summarization corpus. (b) Adaption Phase: In this phase, the parent model is transferred and retrained on the Hindi summarization corpus, using fine-tuning strategy. Given a pre-trained model S with parameter θ_S and layers L_S, for the target model, with parameter θ_T and layers L_T, we use θ_S to learn the adapted model parameters. The new parameters are adapted using a mapping function, $\mathcal{F}$, that maps the parameters are $\mathcal{F}(\theta_S) = \theta_T$. For every layer $l_{si} \in L_S$, where l_{si} is the i^{th} source model layer, fine-tuning requires updating at least one layer of the source model. The parameter of primary concern in this process is the learning rate lr, which can be set differently or the same for every layer during re-training. If lr_s is the learning rate for the source model, and lr_i^t is the learning rate for the i^{th} target model layer, then:

$$lr_i^t > 0, \exists i \in [1, L_T] \quad (1)$$

where $\exists i$ implies that, there exists a layer i in L_T, such that, its $lr > 0$. The model framework is defined as:

Feature Space: The source feature space of the proposed model ($\mathcal{X}_S$) consists of all the English documents to be summarized. The target feature space ($\mathcal{X}_T$) consists of all the Hindi documents to be summarized. Here, $\mathcal{X}_S \in \mathbb{R}^n$, consists of all n-dimensional English word vectors w.r.t the source domain ($\mathcal{D}_S$), and $\mathcal{X}_T \in \mathbb{R}^m$, consists of all m-dimensional Hindi word vectors w.r.t the target domain ($\mathcal{D}_T$). The source and target feature spaces are different from each other ($\mathcal{X}_S \neq \mathcal{X}_T$).

Domain: The source domain (English documents), $\mathcal{D}_S = \{\mathcal{X}_S, \mathcal{P}(X_S)\}$, and the target domain (Hindi documents), $\mathcal{D}_T = \{\mathcal{X}_T, \mathcal{P}(X_T)\}$, are different as the feature spaces are different ($\mathcal{X}_S \neq \mathcal{X}_T$).

Label Space: The source label space, $\mathcal{Y}_S$, corresponding to the source feature space ($\mathcal{X}_S$), is defined as the set of all n-dimensional English word vectors corresponding to the summary ($\mathcal{Y}_S \in \mathbb{R}^n$), and the target label space, $\mathcal{Y}_T$, corresponding to the target feature space ($\mathcal{X}_T$), is defined as the set of all m-dimensional English word vectors corresponding to the summary ($\mathcal{Y}_T \in \mathbb{R}^m$). Here, the source and target label spaces are also different, i.e., $\mathcal{Y}_S \neq \mathcal{Y}_T$.

Task: The source task, $\mathcal{T}_S = \{\mathcal{Y}_S, \mathcal{F}(\cdot)\} = \{\mathcal{Y}_S, \mathcal{P}(Y_S|X_S)\}$, is the task of English ATS. The target task, $\mathcal{T}_T = \{\mathcal{Y}_T, \mathcal{F}(\cdot)\} = \{\mathcal{Y}_T, \mathcal{P}(Y_T|X_T)\}$, is the task of Hindi ATS. For the proposed model, the source and target tasks are the same, i.e., $\mathcal{T}_S = \mathcal{T}_T$.

3.2 ICE-H: Information Coverage Estimate for Hindi Abstracts

To evaluate the efficacy of the models, we propose an evaluation metric that is a slight variant of

Table 1: The Rouge (Recall (R), Precision (P), and F1) Scores of the Base, MED, and RETRAIN (RT) model on the HTSS Test Set for 20k training samples. The HTSS Test Set is obtained by randomly sampling 100 article-summary pairs.

	BASE			MED			RT		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
R1	3.48	2.07	2.60	4.11	2.18	2.85	5.12	2.46	3.32
R2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RL	3.48	2.07	2.60	4.11	2.18	2.85	5.12	2.46	3.32

Table 2: The Rouge (Recall (R), Precision (P), and F1) Scores of the Base, MED, and RETRAIN (RT) model on the HTSS Test Set for 100k training samples. The HTSS Test Set is obtained by randomly sampling 100 article-summary pairs.

	BASE			MED			RT		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
R1	19.9	16.2	17.6	25.7	22.6	23.5	29.1	23.7	25.5
R2	4.60	3.70	4.10	8.40	7.90	7.90	8.70	7.60	7.90
RL	19.3	15.8	17.0	24.3	21.5	22.3	27.4	22.0	23.8

the ICE (Information Coverage Estimate) metric introduced by (Lal et al., 2022a). Like ICE, the source text information (keywords and length), and synonymy of terms (cosine similarity between the source text and summary), are key prerequisites for estimating the information covered in the machine-generated Hindi abstracts using ICE-H.

For a set of Hindi language document-summary pair, $(\mathcal{H}_D, \mathcal{H}_S)$, where the summary length is less than the document length (in terms of word count), that is, $|\mathcal{H}_S| << |\mathcal{H}_D|$, the objective of ICE-H metric, is to find a set of document and summary keywords, $\mathcal{K}_D$ and $\mathcal{K}_S$, such that, $\mathcal{K}_D$ in $\mathcal{H}_D$ and $\mathcal{K}_S \in \mathcal{H}_S$, where $|\mathcal{K}_D| \subseteq |\mathcal{H}_D|$ and $|\mathcal{K}_S| \subseteq |\mathcal{H}_S|$. The intent is to capture the information retained in the abstractive summary by computing the cosine similarity between the overlapping keywords. Ideally, the keywords contained in the document must also be included in its summary to provide good information coverage, that is, $\mathcal{K}_D \cong \mathcal{K}_S$. Each keyword, $k_j^{\mathcal{D}}$ sampled from a summary $\mathcal{H}_S$, is weighed against each keyword, $k_i^{\mathcal{S}}$, sampled from its corresponding document $\mathcal{H}_D$, using a similarity score:

$$\forall_j(\forall_i\ sim(k_j^{\mathcal{S}}, k_i^{\mathcal{D}})) \tag{2}$$

The similarity score estimates how similar two words are in terms of their interpretation or meaning. ICE-H aims to quantify the information coverage delivered by an automatically generated abstractive Hindi summary using the reckoned keyword similarity estimates, $sim(k_j^{\mathcal{S}}, k_i^{\mathcal{D}})$. Here, each summary keyword's coverage score, $Cov_j^{\mathcal{H}}$, is the maximum of its similarity score with each document keyword:

$$Cov_j^{\mathcal{H}} = \forall_j \max(\forall_i\ sim(k_j^{\mathcal{S}}, k_i^{\mathcal{D}}) \tag{3}$$

The information coverage is the summation of the coverage rendered by each sampled summary keyword divided by the number of sampled document keyword terms:

$$ICE - H = \frac{\sum_j Cov_j^{\mathcal{H}}}{|\mathcal{K}_{\mathcal{D}}|} \tag{4}$$

NN-VB-JJ-CD Sampling: The ICE metric (Lal et al., 2022a) examined the efficacy of five sampling strategies (NN, NN-VB, NN-JJ, NN-VB-JJ, NN-VB-JJ-CD), and the experiments showed that NN-VB-JJ-CD sampling outperformed all the other strategies. Therefore, we use the NN-VB-JJ-CD sampling to filter out the keyword sets, $\mathcal{K}_D$ and $\mathcal{K}_S$. The NN-VB-JJ-CD sampling implies that all the words with the NN or VB or JJ or CD POS tag are extracted to form the sampled keyword set, where $\mathcal{K}_D \in \mathcal{H}_D$ and $\mathcal{K}_S \in \mathcal{H}_S$.

4 Experimental Setup

4.1 Datasets

Gigaword corpora: It encompasses news articles and corresponding headlines assembled from

seven different sources. The news articles are up to 31.4 tokens long, and the headlines are up to 8.3 tokens long. Gigaword contains 3.8M training, 189k development, and 1951 test instances.

Hindi Text Short Summarization Corpus (HTSS): We trained the model on 5k, 20k, and 100k sampled articles from the HTSS corpora. HTSS contains articles and their headlines collected from Hindi News Websites. The articles are up to 100 tokens long, and the summaries are 12 tokens long.

4.2 System Configurations

The models were administered using the deep learning library Keras with the following system configurations: 16.4GB Intel Xeon CPU, 17.1GB Tesla P100 Nvidia GPU, and 220GB disk space.

4.3 Model Parameters

Base: The BASE model is a simple attention-based ATS model trained on HTSS corpora. The source and target vocabularies are 35k and 6k most frequent words, respectively. The 100-dimensional word embeddings are randomly initialized using a truncated normal distribution with mean and standard deviation set to 0.0 and 0.01, respectively, and are trained from scratch. All summaries are clipped to a length of 16 words. The model is trained with a batch size of 128 using the RMSProp Optimizer[3]. The learning rate $lr = 0.001$ and $epsilon = 1e - 0.7$. We use a dropout with a rate of 0.4. We train the model for 50 epochs with early stopping criteria[4][5].

MED: MED is a multi-level encoding framework that reads the input sequence twice. The source and target vocabularies are 35k and 6k most frequent words, respectively. Bidirectional-GRU with hidden state dimensionality of 300 is used as the reread encoder. Unidirectional-GRU decoder having a hidden state size of 300 and beam search size of 3 with Bahdanau attention (Bahdanau et al., 2014) is used for generating the summary. All summaries are clipped to a length of 16 words. The model is trained with a batch size of 128 using the RMSProp Optimizer[3], The learning rate $lr = 0.001$ and $epsilon = 1e - 0.7$. We use a dropout with a rate of 0.4. We train the model from scratch for 50 epochs with early stopping criteria[4][5].

Table 3: The ICE-H Scores of the BASE, MED, and RETRAIN model on the HTSS Test Set randomly sampled 100 input-summary pairs. The scores correspond to model summaries generated from training on 100k samples.

Model	ICE-Score
Base	0.2699
Med	0.2047
Retrain	0.1918

RETRAIN: The RETRAIN model comprises two training stages: 1) Primarily, the source model is trained on English Gigaword corpora to form the source model. 2) The source model is then fine-tuned on the target task, i.e., Hindi summarization corpora (HTSS). The model configurations are preserved according to the MED architecture mentioned above. The learning rate $lr = 0.0001$ and $epsilon = 1e - 0.8$. We use a dropout with a rate of 0.2. We retrain the model for 50 epochs with early stopping criteria[4][5].

4.4 Word Embeddings

The ICE-H score is calculated using a similarity measure (cosine similarity) that estimates the extent to which two words are related. The cosine similarity used embedding representations of words or sentences to gauge the similarity score, thus necessitating the transformation of words into real-valued vector representations. The transformations in a vector space deem words with high semantic closeness to be close to each other. We employ the iNLTK (Arora, 2020) open-source library for acquiring the Hindi pre-trained word embeddings. The iNLTK is a powerhouse of various NLP resources for 13 Indian languages, including pre-trained language models, word and sentence embeddings, etc.

4.5 POS Tagger

The sampling strategy for the ICE-H metric requires words to be marked with an appropriate POS tag. The IndoWordNet (Bhattacharyya, 2010) is an integrated framework consisting of wordnet of various Indian languages. *pyiwn*[6] is an user-friendly framework with NLTK WordNet interface

[3] $https://keras.io/api/optimizers/rmsprop/$

[4] $https://keras.io/api/callbacks/early_stopping/$

[5] $https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/$

[6] https://pypi.org/project/pyiwn/

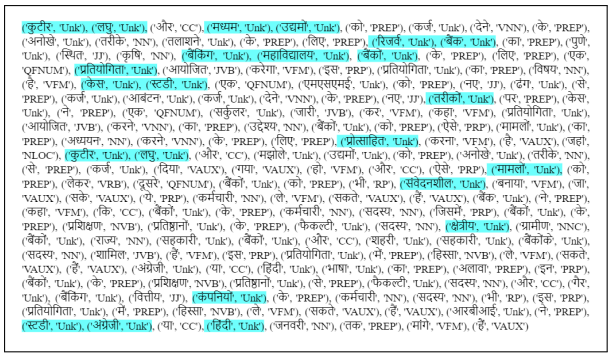

Figure 2: Examples of POS tagged sequence with <Unk> mark.

in Python (Panjwani et al., 2018). We use the IndoWordNet for attaining the Hindi POS tags.

5 Results

We examine the BASE, MED, and RETRAIN model performances against 5k, 20k, and 100k training samples. For 5k sample training, the models yield incompetent summaries and the Rouge scores are computed to be 0.0 for all three aspects, Rouge-1, Rouge-2, and Rouge-L. The Rouge scores are observed to show an incline from 5k training samples to 20k training samples (Table 1). The rise is not significant, and the model summaries still yield unsatisfactory results with grammatically incorrect sentences, repetitive phrases, and factual inconsistencies. Despite the absurdity, it was observed that the models can generate expressions and retain some facts. The Rouge scores increase significantly for 100k training samples (Table 2), generating remarkably well-formed summaries. For 100k sample training, the model summaries are observed to be grammatically well-formed, retaining multiple facts and no repetitive information (Figure 5). The Rouge Scores (R1, R2, and RL) on 20k and 100k HTSS training samples for all the models, are stated in Tables 1 and 2, respectively.

BASE: For 20k training samples, the model produces an R1 F1 Score of 2.60, R2 F1 Score of 0.0, and RL F1 Score of 2.60. The results signify a very narrow overlap of words between the original summary and BASE-generated summary, i.e., around 2.6% 1-gram overlap, 0% 2-gram overlap, and 2.6% LCS overlap. For 100k training samples, the model produces an R1 F1 Score of 17.6, R2 F1 Score of 4.1, and RL F1 Score of 17.0. The results signify a considerable overlap of words between the original summary and BASE-generated summary, i.e., around 17.6% 1-gram overlap, 4% 2-gram overlap, and 17% LCS overlap.

MED: For 20k training samples, the model produces an R1 F1 Score of 2.85, R2 F1 Score of 0.0, and RL F1 Score of 2.85. The results signify a very narrow overlap of words between the original and MED-generated summaries, i.e., around 2.8% 1-gram overlap, 0% 2-gram overlap, and 2.8% LCS overlap. For 100k training samples, the model produces an R1 F1 Score of 23.5, R2 F1 Score of 7.9, and RL F1 Score of 22.3. The results signify a significant overlap of words between the original and MED-generated summaries, i.e., around 23% 1-gram overlap, 8% 2-gram overlap, and 21% LCS overlap.

RETRAIN: For 20k training samples, the RETRAIN model produces an R1 F1 Score of 3.32, R2 F1 Score of 0.0, and RL F1 Score of 3.32. The results signify a very narrow overlap of words between the original summary and RETRAIN-generated summary, i.e., around 3.3% 1-gram overlap, 0% 2-gram overlap, and 3.3% LCS overlap. The RETRAIN model observes around 27% rise in R1 F1 and RL F1 scores compared to the BASE and MED models. For 100k training samples, the model produces an R1 F1 Score of 25.5, R2 F1 Score of 7.9, and RL F1 Score of 23.8. The results signify a significant overlap of words between the original summary and RETRAIN-generated summary, i.e., around 25% 1-gram overlap, 8% 2-gram overlap, and 23% LCS overlap. The model observes around 44% rise in R1 F1 and RL F2 scores compared to the BASE and MED models.

5.1 ICE-H Scores

The ICE-H scores on 100k HTSS training samples for the BASE, MED, and RETRAIN models have been stated in Table 3. The BASE model generates a score of 2.699, exhibiting an extremely low level of informativeness in the summaries. The ICE-H scores are observed to decline for the MED and RETRAIN models, where MED achieves a score of 0.2047 and RETRAIN a score of 0.1918. The low scores attained by the models are attributed to the various shortcomings encountered when devising the metric, which is discussed in Section 6.

Figure 3: Examples of generated summaries on the HTSS corpus trained on 5k training samples. **Review:** source news article, **Original summary:** actual headline, **Base:** output of the BASE model, **MED:** output of the MED model, and **RT:** output of the RETRAIN model.

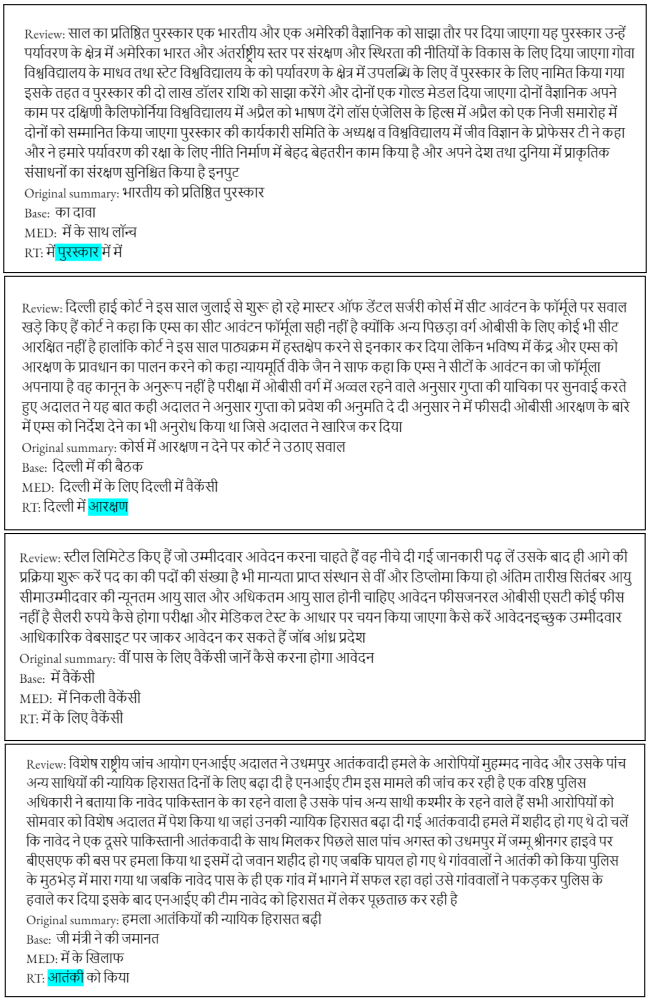

Figure 4: Examples of generated summaries on the HTSS corpus trained on 20k training samples. **Review:** source news article, **Original summary:** actual headline, **Base:** output of the BASE model, **MED:** output of the MED model, and **RT:** output of the RETRAIN model.

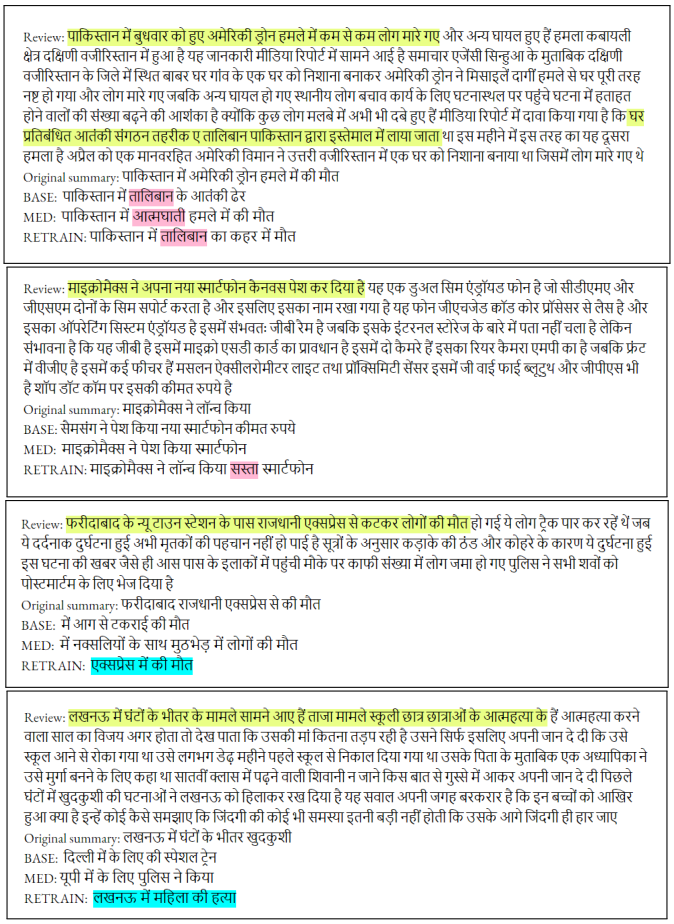

Figure 5: Examples of generated summaries on the HTSS corpus trained on 100k training samples. **Review:** source news article, **Original summary:** actual headline, **Base:** output of the BASE model, **MED:** output of the MED model, and **RT:** output of the RETRAIN model.

5.2 Computational Costs

On the HTSS dataset sampled at 100k, the BASE, MED, and RETRAIN model's training time per epoch is about 5.5 hours, 5.3 hours, and 2 hours, respectively. We trained each model for 50 epochs with early stopping on a 17.1GB Tesla P100 Nvidia GPU. The RETRAIN model is the fastest to train and generates the best Rouge scores. As shown in Figure 4, the BASE model takes the most amount of time and converges after 40 epochs, and the MED model takes almost the same time and converges after 37 epochs. Still, the RETRAIN model takes only 17 epochs for converging and trains approximately three times faster than the BASE and MED models.

Table 4: Training speed of the BASE, MED, and RETRAIN model for 100k sample training. The training speed is calculated as the elapsed time (hours) per epoch, tested on a 17.1GB Tesla P100 Nvidia GPU card.

	Epochs	**Training Time (hrs)**	**Total Time (hrs)**
BASE	40	0.137 hrs/epoch	5.480
MED	37	0.142 hrs/epoch	5.254
RETRAIN	17	0.111 hrs/epoch	1.887

6 Challenges

The low performances relayed by the three models may be a result of one of the following:

1. **Size of the dataset for model training:** Since the experiments are carried out to investigate the model's efficacy in the low-resource scenario, it is certainly challenging to attain high model performance. As established by the results (Figures 3 and 4), training an ATS model from scratch, with 5k and 20k training samples, is likely to deliver substandard results. However, as can be seen from the development from 5k to 100k (Figures 3 through 5), increased training data promises the creation of fluent and informative sentences.

2. Comparatively longer input articles in Target Domain: The pre-trained feature extractor (source model) is learned on the Gigaword dataset that has input articles of up to 31.4 tokens long on average. All deep learning frameworks demand a fixed input length value to be specified for training, as the networks process only fixed-length inputs. Since the source model is completely transferred for fine-tuning on the target model, the hyperparameter values are preserved. However, the target dataset contains input articles of up to 100 tokens long on average. Longer sequences cause the models to generate a summary by considering only the first 3 to 4 sentences.

3. Lack of efficient POS Taggers: Due to a lack of efficient POS Taggers, keywords could not be efficiently sampled to find the coverage score. Many words were POS tagged as "unk" or "keyword error" was thrown in case of words that were not lemmatized, i.e., words having inflectional endings and not the BASE form, as shown in Figure 2. The existence of phonetic translations of English words in the target Hindi dataset also makes it harder to acquire a POS mark.

4. Lack of adequately trained Hindi word embeddings: The iNLTK library used for acquiring the Hindi word embeddings was inadequate in providing word vectors for multiple Hindi words in the dataset. Since the dataset contains multiple word inflexions, phonetically translated English words, etc., multiple keywords could not be considered when computing the ICE-H estimate. This loss of key information led to significant inaccuracies in the ICE-H scores. The scores could not be rendered useful for gauging the information covered in the model-generated summaries.

7 Conclusion

The primary aim of this preliminary assessment is to conduct experiments to test the effectiveness of three models in the low-resource setting, regular attention-based encoder-decoder (BASE), multi-level encoder-decoder (MED), and TL-based fine-tuned model (RETRAIN), for performing abstractive Hindi text summarization. To test how the size of the training sample effects the output summary, we create randomly sampled instances of sizes 5k, 20k, and 100k from a Hindi corpora containing news-headline pairs, for fine-tuning the three models. At 5k, all the models are unable to extract meaningful information from the text and instead produce word repetition sequences that neither make sense nor convey any significant information. As the sample size is increased to 20k, all models start to extract significant words from the text, but the sentences are still poorly constructed. The RETRAIN model, however, is able to predict important words from text but the headline is still incompetent. At 100k, every model begins crafting coherent headlines that convey significant information, where, the RETRAIN strategy produced headlines that were more well-focused and informative. For assessing the quality of the Hindi abstracts generated for these random samples, we also propose an evaluation metric (ICE-H), the framework of which is borrowed from the ICE metric for evaluating English language summaries. We evaluate the respective model's performance using ICE-H and report our findings and limitations.

References

Ayham Alomari, Norisma Idris, Aznul Qalid Md Sabri, and Izzat Alsmadi. 2022. Deep reinforcement and transfer learning for abstractive text summarization: A review. *Computer Speech & Language*, 71:101276.

Gaurav Arora. 2020. inltk: Natural language toolkit for indic languages. *arXiv preprint arXiv:2009.12534*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

B Mohan Bharath, B Aravindh Gowtham, and M Akhil. 2022. Neural abstractive text summarizer for telugu language. In *Soft Computing and Signal Processing*, pages 61–70. Springer.

Pushpak Bhattacharyya. 2010. IndoWordNet. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*.

Yi-Syuan Chen and Hong-Han Shuai. 2021. Meta-transfer learning for low-resource abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12692–12700.

Radia Rayan Chowdhury, Mir Tafseer Nayeem, Tahsin Tasnim Mim, Md Saifur Rahman Chowdhury, and Taufiqul Jannat. 2021. Unsupervised abstractive summarization of bengali text documents. *arXiv preprint arXiv:2102.04490*.

Varsha R Embar, Surabhi R Deshpande, AK Vaishnavi, Vishakha Jain, and Jagadish S Kallimani. 2013.

saramsha-a kannada abstractive summarizer. In *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 540–544. IEEE.

Pascal Fecht, Sebastian Blank, and Hans-Peter Zorn. 2019. Sequential transfer learning in nlp for german text summarization. In *SwissText*.

Sebastian Gehrmann, Zachary Ziegler, and Alexander M Rush. 2019. Generating abstractive summaries with finetuned language models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 516–522.

Sudha Gopalakrishnan. 2009. Manuscripts and indian knowledge systems: the past contextualising the future. In *3rd International UNUESCO Conference*.

Som Gupta and Sanjai Kumar Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65.

Rajina Kabeer and Sumam Mary Idicula. 2014. Text summarization for malayalam documents — an experience. In *2014 International Conference on Data Science Engineering (ICDSE)*, pages 145–150.

Jagadish S Kallimani, KG Srinivasa, and B Eswara Reddy. 2016. Statistical and analytical study of guided abstractive text summarization. *Current Science*, pages 69–72.

Rishabh Karmakar, Ketki Nirantar, Prathamesh Kurunkar, Pooja Hiremath, and Deptii Chaudhari. 2021. Indian regional language abstractive text summarization using attention-based lstm neural network. In *2021 International Conference on Intelligent Technologies (CONIT)*, pages 1–8. IEEE.

Daisy Monika Lal, Krishna Pratap Singh, and Uma Shanker Tiwary. 2022a. Ice: Information coverage estimate for automatic evaluation abstractive summaries. *Expert Systems with Applications*, 189:116064.

Daisy Monika Lal, Krishna Pratap Singh, and Uma Shanker Tiwary. 2022b. Multi-level shared-weight encoding for abstractive sentence summarization. *Neural Computing and Applications*, 34(4):2965–2981.

Chang-Shing Lee, Zhi-Wei Jian, and Lin-Kai Huang. 2005. A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(5):859–880.

Parth Mehta. 2016. From extractive to abstractive summarization: A journey. In *ACL (Student Research Workshop)*, pages 100–106. Springer.

Sindhya K Nambiar, Sumam Mary Idicula, et al. 2021a. Attention based abstractive summarization of malayalam document. *Procedia Computer Science*, 189:250–257.

Sindhya K Nambiar, S David Peter, and Sumam Mary Idicula. 2021b. Abstractive summarization of malayalam document using sequence to sequence model. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 347–352. IEEE.

Ritesh Panjwani, Diptesh Kanojia, and Pushpak Bhattacharyya. 2018. pyiwn: A python based api to access indian language wordnets. In *Proceedings of the 9th Global Wordnet Conference*, pages 378–383.

CS Saranyamol and L Sindhu. 2014. A survey on automatic text summarization. *International Journal of Computer Science and Information Technologies*, 5(6):7889–7893.

Shagun Sinha and Girish Nath Jha. 2020. Abstractive text summarization for sanskrit prose: A study of methods and approaches. In *Proceedings of the WILDRE5–5th Workshop on Indian Language Data: Resources and Evaluation*, pages 60–65.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014a. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014b. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Md Ashraful Islam Talukder, Sheikh Abujar, Abu Kaisar Mohammad Masum, Fahad Faisal, and Syed Akhter Hossain. 2019. Bengali abstractive text summarization using sequence to sequence rnns. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE.

Sabina Yeasmin, Priyanka Basak Tumpa, Adiba Mahjabin Nitu, Md Palash Uddin, Emran Ali, and Masud Ibn Afjal. 2017. Study of abstractive text summarization techniques. *American Journal of Engineering Research*, 6(8):253–260.

Julio Christian Young and Andre Rusli. 2019. Review and visualization of facebook's fasttext pretrained word vector model. In *2019 International Conference on Engineering, Science, and Industrial Applications (ICESI)*, pages 1–6. IEEE.

Ekaterina Zolotareva, Tsegaye Misikir Tashu, and Tomás Horváth. 2020. Abstractive text summarization using transfer learning. In *ITAT*, pages 75–80.