# Textual entailment as an evaluation metric for abstractive text summarization

Swagat Shubham Bhuyan, Saranga Kingkor Mahanta, Partha Pakray, Benoit Favre

**Noname manuscript No.**
(will be inserted by the editor)

# Textual Entailment as an Evaluation Metric for Abstractive Text Summarization

**Swagat Shubham Bhuyan · Saranga Kingkor Mahanta · Partha Pakray · Benoit Favre**

the date of receipt and acceptance should be inserted later

**Abstract** Automated text summarization systems require to be heedful of the reader and the communication goals since it may be the determining component of whether the original textual content is actually worth reading in full. The summary can also assist enhance document indexing for information retrieval, and it is generally much less biased than a human-written summary. A crucial part while building intelligent systems is evaluating them. Consequently, the choice of evaluation metric(s) is of utmost importance. Current standard evaluation metrics like BLEU and ROUGE, although fairly effective for evaluation of extractive text summarization systems, become futile when it comes to comparing semantic information between two texts, i.e in abstractive summarization. We propose textual entailment as a potential metric to evaluate abstractive summaries. The results show the contribution of text entailment as a strong automated evaluation model for such summaries. The textual entailment scores between the text and generated summaries, and between the reference and predicted summaries were calculated, and an overall summarizer score was generated to give a fair idea of how efficient the generated summaries are. We put forward some novel methods that use the entailment scores and the final summarizer scores for a reasonable evaluation of the same across various scenarios. A Final Entailment Metric Score (FEMS) was generated to get an insightful idea in order to compare both the generated summaries.

**Keywords** Automatic Evaluation · Abstractive Text Summarization · Text Entailment · Natural Language Processing · Deep Learning

✉ Swagat Shubham Bhuyan[1]*
swagat_ug@cse.nits.ac.in

Saranga Kingkor Mahanta[1]*
saranga.mahanta@ip-paris.fr

Partha Pakray[2]**
partha@cse.nits.ac.in

Benoit Favre[3]**
benoit.favre@lis-lab.fr

[1] Computer Science and Engineering department, National Institute of Technology Silchar, Assam, India

[2] Electronics and Communication Engineering department, National Institute of Technology Silchar, Assam, India

[3] Laboratoire d'Informatique et Systèmes / CNRS, Aix-Marseille Université, Marseille, France

* joint first author
** directional leads

## 1 Introduction

Abstractive text summarization is a keystone Natural Language Processing task that is still in its adolescence when it comes to big data perspectives, keeping grammar and other such literary rules in mind. At the same time, many such implementations exist that try to tackle this NLP problem with the best efficiency possible today. However, a true metric to judge such summaries is still dependent on a thorough human-dominant review. Many evaluation metrics such as ROUGE and BLEU give a fair idea of where the predicted summary stands in contrast to the original document, but, without human verification, it is hard to avoid exceptions and incorrect assessments. Automatic evaluation of machine-generated abstractive text summaries is hence still a very interesting problem to address in the field of NLP. The idea of constructing a neural network just to serve the purpose of evaluating the summaries generated by another such NLP model is revolutionary and can be considered as the foundation of a textual entailment-based evaluation metric, thanks

to many works [4] [21] that have worked on the Recognizing Textual Entailment (RTE) problem in context with abstractive summarization before. The purpose of this work is to devise a method that has the potential to reduce the dependence on manual labour when it comes to evaluating summaries. This has been carried out using a Textual Entailment neural network to efficiently identify if the generated summary is an entailment of the original text and/or the reference summary. We also devised some expressions that use these entailment scores to provide a fair pretense towards its evaluation and correctness. For training the base abstractive summarizer, the InShorts News Corpora has been used and will be explained in further sections. Many approaches for the text entailment model have been proposed, but the paper has proceeded with an LSTM based neural network text entailment approach, where most of the feature extractions are carried out in the hidden layers itself without explicitly summoning these paradigms. An appreciable set of evaluation metrics are taken to review the abstractive summaries, which complement well with the text entailment evaluation metric to give a more insightful assessment of the final generated summary. It was found that the transformer model performed much better than the base seq2seq model in summary generation. This was tested thoroughly using the set of evaluation metrics aided with the strong text entailment automated evaluator, with both its summary-based entailment scores and mutual entailment scores as well. The mutual entailment score, although experimental, performed surprisingly well in the seq2seq model generated summaries. Finally, an accumulated score was calculated to determine the final efficiency of the test summaries generated which had a strong emphasis on the entailment decisions.

The organization of the paper is as follows: Section 2.1 consists of the Literature Review and all related works. Section 3 gives a brief description of the datasets used. Section 2.2 peruses through the theory of all the related topics and sub topics that are covered throughout this research paper. Objectives, some entailment approaches, similarity metrics, etc. are covered in this section. Section 4 elaborates on the system description and the experimental setup. The results are elucidated in Section 5. Finally, Section 6 concludes the paper with a conclusive review and future scopes.

## 2 Background

### 2.1 Literature Review

Earlier works mostly used rule-based approaches. Luhn used word frequencies to find the most indicative sentences for summarization [20]. Words that appeared frequently and in close proximity to one another suggested significant sentences. Thresholds were established so that the most common and least frequent terms were disregarded. [7] also employed cue words, title and header words, and structural indications such as sentence position in addition to word frequencies. Significant sentences or paragraphs appear extremely early and very late in the section or document, according to this research. Eventually, Machine learning algorithms came into play. On chosen extracts, Kupiec et al. [15] used a Naive-Bayes classifier-based supervised technique. Sentence length cut-off, fixed-length, paragraph, thematic word, and uppercase words were among the parameters used to train the classifier. In [29], a centroid-based summarising technique for multi-document summarization was presented. Clusters were formed by grouping together similar texts and sentences where each cluster could be a different sub-topic. The cluster centroid was a pseudo document that represented the cluster. Sentences similar to the centroids would be included in the summary. [3] suggested a domain-sensitive content model that used a Hidden Markov Model with domain topics as states and generated sentences that were relevant to that topic. The sentences were created using an n-gram model that learned both content selection and information order. TextRank was proposed in [22], a graph-based system in which each sentence was regarded as a node in the graph, based on Google's PageRank algorithm. The edges of the graph were matched to the similarity of sentences. The text was converted into a weighted graph, which was then subjected to a ranking system (such as HITS, POS, or PageRank). For the summary, the graph nodes with the highest scores were chosen.

Neural networks were applied for abstractive summarization in [30]. This method combined a neural language model with an attention-based input encoder. Three different encoders were used in the experiments: bag-of-words, convolutional (TDNN), and attention-based; the latest, being the most successful. Only the first sentence was used in the studies, therefore the process was limited to headline generation from the first sentence only. An attentional encoder-decoder RNN model was used by Nallapati et al. for achieving an abstractive summarization [23]. Uncommon or OOV (Out Of Vocabulary) words were addressed with a pointer-generator model. At the word and sentence levels, the attention mechanism was hierarchical. Since prior datasets were confined to single-sentence summaries, a new dataset based on CNN/DailyMail [11] news articles was introduced, with summaries averaging 53 words and 3.72 sentences. This study became the foundation for the base model for abstractive summaries of large corpora in the field. Liu et al. attempted to generate Wikipedia articles in [18]. They chose the most significant content tokens in the extractive step, and for the abstractive phase, they developed a scalable decoder-only transformer architecture that integrated input and output sequences into a consistent singular sequence. The final model was comprised of five layers, with memory-compressed and local attention both present and

the network altering between these two mechanisms. Fan et al. improved performance by feeding knowledge graph representations of the text to a seq2seq model [9]. In [9], BERTSUM, a BERT modification for summarising, was proposed. Multiple sentences were encoded as a unified input sequence by the mentioned model. To separate the sentences, interval segment embeddings were implemented. Different summarization layers such as basic classifier, RNN, and inter-sentence transformer were tested for fine-tuning and extracting document-level features.

Plenty of studies have used textual entailment while generating text summaries to primarily boost the performance of the systems. Lloret et al. used textual entailment recognition along with text summarization and achieved a performance improvement of 6.78% [19]. The work compared the performance of the combined approach with the DUC 2002 baseline model and their own word-frequency-based model. Gupta et al. [10] proposed a method that considered text summarization as an optimization problem and used a graph-based algorithm, Weighted Minimum Vertex Cover (WMVC) to solve it. They also used textual entailment to measure sentence connectivity and construct the graph on which WMVC operated. In [35] text entailment was used for text segmentation and summarization. Saini et al. used the textual entailment score between sentences in a summary and a figure's caption as one of the sentence scoring features for automatic figure summarization in biomedical scientific articles [31]. Textual entailment was used to propose a robust evaluation method for Machine Translation outputs in [25] and [24]. Pasunuru et al. used a reinforcement learning approach for abstractive summarization[27]. They exploited entailment as one of the two of their reward functions.

Falke et al. [8] evaluated summaries generated by abstractive SOTA models by crowd-sourcing. They showed whether textual entailment predictions could be used to detect errors and reduce them by re-ranking alternative generated summaries. A weakly-supervised model based on observations from the errors made by SOTA summarization models was proposed in [14] for evaluating factual consistency between the source documents and corresponding predicted summaries. Zhang et al. [37] proposed BERTSCORE, an efficient metric for text generation evaluation that computes token similarity using contextual embeddings. It was claimed to correlate better with human judgements. Another learned evaluation metric based on BERT, BLEURT was proposed in [33].

## 2.2 Related Theory

### *2.2.1 Standard Evaluation Metrics*

Standard evaluation metrics like ROUGE and BLEU despite being excellent evaluation metrics for many NLP tasks including extractive summarization have their limitations when it comes to evaluating semantic meaning or similarities between sentences.

1. **BLEU** : The Bilingual Evaluation Understudy (BLEU) score [26], originally introduced to evaluate machine translation systems is one of the most conventional metrics used for abstractive text summarization evaluation as well. It is a language-independent method that correlates highly with human evaluation. The requirement to perform a reasonable evaluation, however, is a number of reference sentences with which the machine-generated output can be compared. BLEU compares the n-gram of the machine-generated output to the n-grams of the reference outputs to count the number of matches. A modified precision formula is used to get a fair estimate of the matches. The final BLEU score of a generated output sentence is basically a weighted combination of the modified precision formula scores on different n-grams, as shown in Equation 1.

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log p_n) \tag{1}$$

where, $N$ is the number of grams (usually the unigram, bigram, trigram and 4-gram are used), $w_n$ is the weight of the $n_{th}$ modified precision score $p_n$. $BP$ denotes Brevity Penalty which is an adjustment factor that penalizes output sentences by using exponential decay on generated sentences that are shorter than the provided reference sentences.

The major limitation of BLEU lies in the fact that it cannot estimate the semantic relevance of a sentence. Different words in different orders of their placement may reflect the same expected meaning and to capture that, various reference sentences containing different words in different orders that express the same meaning need to be provided.

2. **ROUGE** Recall-Oriented Understudy for Gisting Evaluation [17] is a set of metrics consisting of ROUGE-N, ROUGE-L, ROUGE-S, etc. In ROUGE-N the n-grams of sentences are used. It measures the number of matching n-grams between the model-generated summary and a reference summary. The ROUGE recall is the count of the number of matching n-grams found in both the reference summary and model generated summary divided by the total number of n-grams in the reference summary. The ROUGE precision is the same but it is divided by the total number of n-grams in the model generated summary. The ROUGE F-score is simply the harmonic mean of the ROUGE precision and ROUGE recall. Thus, it relies on the model capturing as many matching words as possible (recall) without generating a lot of irrelevant words (precision). The ROUGE-L measures the

longest common subsequence (LCS) between the model-generated output summary and the reference summary. The longest sequence of tokens that is shared between both is counted. Like BLEU, however, even ROUGE has no scope of evaluating any kind of semantic meaning.

*2.2.2 Standard Text Entailment approaches*

In the most basic sense, a premise P is said to entail a hypothesis H if H is true in every circumstance of possible world in which P is true. In the complexity chart of NLP, textual entailment can be considered in the highest layer of discourse and coreference. The most crucial objective of text entailment can basically be considered to be text similarity, but a more refined version of the same. This means that on entailment detection, the pair of corpus is not only similar to each other, but also, the hypothesis is a valid representation of the premise, meaning abstraction is achieved in summary generation.

1. **Bag of Words based approach** A very simplistic approach to see the common words between premise and hypothesis. This approach falls short due to lack of complexity. It lacks detection of negation and sentiment, as a negating word is simply an added word, hence, the bag of words score will remain high even if the hypothesis is a contradiction of the premise in many cases. This is why a more refined text entailment approach was sought after very popularly by the NLP community to tackle these hurdles.
2. **Dependency tree based approach** In a dependency tree based approach, the corpus is tagged into syntactic elements using a dependency parser. It passes the various tags for each word and puts them in a tree format for analysis.
3. **Predicate based approach** The premise and hypothesis pair is transformed into a predicate-argument tuple. This makes it easier for analysis based on predicate solution which is a very popular logic problem solving field, and hence, text entailment detection will be made easier in this form. Theory proving algorithms are then used on the predicate-argument pair and hence, text entailment is checked.
4. **Probabilistic approach** According to this approach, a premise $p$ is said to entail a hypothesis $h$ (denoted as p $\rightarrow$ h) if $p$ escalates the likelihood of $h$ being true, according to Equation 2.

$$P(Tr_h = 1|p) > P(Tr_h = 1) \tag{2}$$

This means that the conditional probability of the hypothesis given the premise should be more than the unconditional probability of the hypothesis.

There exists a paradox in this approach on application of Bayes' Theorem and further resolution, it shows that premise entails hypothesis only if the hypothesis entails premise. This is clearly false, as the entailment should not depend on the converse. A concept of mutual entailment is introduced later in this paper, which dabbles around this idea, but that is different as it only ensures the strength of the entailment rather than implying cases other than p $\rightarrow$ h && h $\rightarrow$ p are not entailments.

5. **Deep Semantic based approach** This approach is based on the workings of UNL (Universal Networking Language) tool. UNL is a tool for representing text in terms of semantic relations between entities. It consists of:
   - Universal Words: Unambiguous concepts in the sentence.
   - Attributes: How a concept is used in a sentence.
   - Relations: Denote semantic dependency between constituents
6. **Machine Learning approach** This is perhaps the most sought after approach when tackling the RTE problem. Many researchers in the field of NLP apply deep learning methodologies to detect entailments, hence marking the foundation of various text entailment applications such as Question Answering Systems, Machine Translation, and as in the case of this paper, Abstractive Summarizing and Automatic Evaluation. Many deep learning concepts such as word embeddings, recurrent neural networks, LSTMs, etc. will be discussed briefly in the following sections.

## 3 Datasets

For the base abstractive summarization model, Inshorts News Data (obtained from Kaggle) was used for training the model. It consisted of 55,014 rows of Text-Headline pairs. The Headlines were considered as summaries of the corresponding text. The text body consisted of around 60 words while the headlines were of around 15-20 words. This dataset is relatively very small in terms of language tasks. However, it was chosen due to our limitations in resources, and because the base model was created from scratch and the training had to take place without freezing any weights. Furthermore, the attention layer [2] runs with quadratic time complexity, thus increasing the training time greatly.

The Stanford Natural Language Inference (SNLI) [5] corpus is a collection of 570,000 human-written English sentence pairs manually labeled as entailment, contradiction, or neutral. It was used to train our Textual Entailment model since this dataset is balanced and most conveniently contains the desired annotations. On preprocessing, the gold labels were examined carefully, and cleaned for further usage by the text entailment model, which will be discussed in further sections. The SNLI corpus comprises of 3 arranged datasets:

train, validation and test. The training dataset comprises of 549,361 training pairs with gold labels. The validation set has 9842 sets of pairs and gold labels and the test set contains 9824 records.

Generated Abstractive Summaries is a collection of 100 test data generated from the base summarizer model and the BART model are passed through the text entailment model and a score is generated depicting how good the summary generated is.

## 4 System Description

The entire methodology along with more relevant theoretical concepts have been elucidated in the following sections.

### 4.1 Abstractive Summarization base model

#### *4.1.1 Pre-processing*

Neural networks can be fed only numerical data. The text-headline pairs, both being text data had to pass through various pre-processing stages before being fed into the model for training.

Currency symbols and word contractions were replaced by the text abbreviations and expansions respectively. For instance, "$" to "USD", didn't" to "did not", "you'll" to "you will", and so on. All the characters were converted to lowercase and stopwords were removed using the NLTK English stopwords list. The word counts distribution of the cleaned texts and headlines can be seen in Figure1.

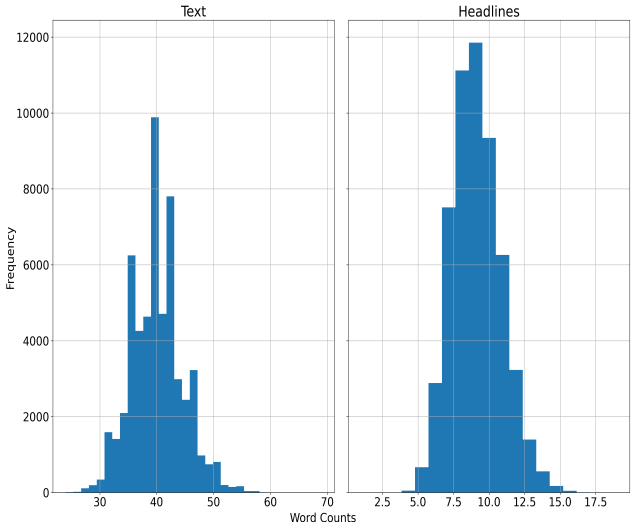


**Fig. 1** Histogram of Word Counts in the Inshorts News Dataset

With reference to the histograms, a maximum word length of 57 for texts and 15 for headlines were chosen. These numbers can further be justified because 99.9% of the text bodies had word counts of less than 57 and 99.93% of the summaries had word counts of less than 15. After adding the SOS (Start of Sentence) and EOS (End of Sentence) tokens to each summary example, the dataset was then split into train and test splits in the ratio of 9:1. Consequently, the train split consisted of 49512 rows and the test split had 5502 rows of cleaned text-summary pairs. The data was then tokenized using the keras.preprocessing.text.Tokenizer class [6]. The text data finally got converted to a sequence of numbers after the tokenization. It was followed by padding using the keras.preprocessing.sequence.pad_sequences class. The padding was done by appending zeros at the end of each sequence to make all the examples have a similar length, in our case 57 for texts and 15 for summaries.

#### *4.1.2 LSTMs*

Vanilla Recurrent Neural Networks (RNNs) pose a problem of vanishing and exploding gradients. The LSTM [12] provides a solution to these problems through the modification of the RNN. LSTMs perform notably well with data that is sequential in nature, basically where prediction outputs depend on previous inputs as well and can handle long-term dependencies and contextual weights. It introduces a new configuration to the typical vanilla architecture of RNNs (Recurrent Neural Networks) i.e.- the memory cell consisting of basically 4 constituents: input gate, forget gate, output gate, and a neuron that simply links to the same neuron itself. The LSTM cell deals with the gradient vanishing problem by withholding the error information that can be disseminated through the mechanism of back-propagation in between the layers and time. Inside the LSTM, the cell state $(C_t)$ is well connected to three gates, which are basically the input gate $(i_t)$, output gate $(o_t)$, and forget gate $(f_t)$ separately. $\bar{C}_t$ demonstrates the cell state at any timestamp, t, $W_x$ and $b_x$ denotes the weights and bias terms for the respective gates $x$, respectively. $x$ can either be a $(i)$ input gate, a $(f)$ forget gate, an $(o)$ output gate, or even a $(c)$ cell state vector. We consider $h_{t-1}$ to be the output of the former LSTM cell at a given timestamp $t-1$ and $x_t$ can denote the input at timestamp $t$. Equations 3, 4, 5, 6, 7, 8 clearly portray the functions discussed above.

$$f_t = \sigma(b_f + W_f \cdot [h_{t-1}, x_t]) \tag{3}$$
$$i_t = \sigma(b_i + W_i \cdot [h_{t-1}, x_t]) \tag{4}$$
$$\bar{C}_t = \tanh(b_C + W_C \cdot [h_{t-1}], x_t) \tag{5}$$
$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \tag{6}$$
$$h_t = o_t * \tanh(C_t) \tag{7}$$
$$o_t = \sigma(b_0 + W_0 | h_{t-1}, x_t |) \tag{8}$$

#### 4.1.3 Attention Mechanism

The attention mechanism basically improves the performance of the model to a great extent by imparting the model the ability to weigh attention. In other words, it endorses the model to know which are the relevant parts of the inputs. Bahdanau et al. [2] first introduced and implemented this concept in a Machine Translation task. Similar to their work, the attention layer has been used in the decoder part of our base abstractive text summarizer. The context vector $c_i$ that generates the most probable word on a particular time step is dependable on a sequence of annotations $(h_1, ..., h_{T_x})$ to which the encoder maps the input sequence ($T_x$ is the total time steps in the sequence). $c_i$ is calculated as a weighted sum of the annotations $h_i$ as shown in Equation 9.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{9}$$

where, $\alpha$ is the attention weight, the amount of attention that should be paid to $h_j$. The attention weights are calculated using Equation 10.

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{j=1}^{T_x} e_{ij}} \tag{10}$$

where, $e_{ij}$ can be calculated using a small neural network (usually having one hidden layer) that learns the weights during gradient descent while backpropagating to formulate a relevant function that attempts to capture the alignment between input at $j$ and output at $i$. Consequently. the inputs to this neural network would be $s_{i-1}$, the hidden state from the previous time step before emitting $y_i$ (output at $i$), and $h_j$, the $j$th annotation of the input sequence. Equation 10 is basically a softmax function used on $e_{ij}$ in order to make the total sum of the attention weights equal to one. $\alpha$ essentially exhibits the importance of $h_j$ with respect to the previous hidden state $s_{i-1}$ in order to decide the next state $s_i$ and generating a probable target word $y_i$. Henceforth, the decoder is able to decide which parts of the input sentence it should pay attention to while generating a word at a particular time step. In this work, global attention is used, i.e all the hidden states of the encoder LSTM network is used while calculating the context vector.

#### 4.1.4 Experimental Setup

For our base abstractive text summarizer, an embedding dimension of 110 was used for the text inputs. The encoder consisted of an embedding layer and three stacked LSTM layers with a latent dimension of 200. Dropouts and recurrent dropouts were used to restrict overfitting. The decoder is comprised of an embedding layer, an LSTM layer, an attention layer, and finally a time-distributed dense layer with the same number of nodes as the vocabulary space of the summaries with a softmax activation function. The skeleton of the model is shown in Figure 2. The Adam optimizer [13] and the categorical-cross-entropy loss function were used. Due to limited resources, the training ran for 10 epochs with a batch size of 16. The Tensor Processing Unit available in Google Colaboratory (16 GB RAM, NVIDIA Tesla V100 GPU) was exploited to fasten the training process. It took approximately 5 hours for the training to complete.

For comparison of the results, summaries of the test set were also generated using a pre-trained BART [16] transformer [36] model fine-tuned on the CNN/Daily Mail dataset [11].

An example of summaries generated by both models is shown below:

**Cleaned text:** *saudi stock exchange appointed sarah al ceo investment bank ncb capital first female head first woman chair major government financial institution kingdom chair arab world 39 largest stock exchange time preparing offer shares public*

**reference summary:** *saudi stock exchange appoints its first female head*

**Base model generated summary:** *saudi arabia sign 1st ever female woman*

**Transformer generated summary:** *saudi stock exchange appointed sarah al ceo investment bank ncb capital first female*

### 4.2 Text Entailment model

Text entailment is a crucial metric to check for a given set of premises and their corresponding hypotheses. A typical base model would simply run through the sentence pairs using a simple RNN cell and a final softmax activation function on the concatenated fully connected linear layers which would put the sentence pairs into one of the three classes based on the maximum probability of the class value.

#### 4.2.1 Pre-Processing

Most of the pre-processing are usually done while summarizing itself in the base seq2seq model, but since a more advanced transformer-based summarizer was introduced, the SNLI dataset, as well as the generated summary set for testing, has to undergo pre-processing. Lemmatization was carried out on the corpus, stop words and special symbols were removed. Unnecessary columns were dropped and the gold labels were identified for each sentence sets from the SNLI dataset and were sequenced efficiently.

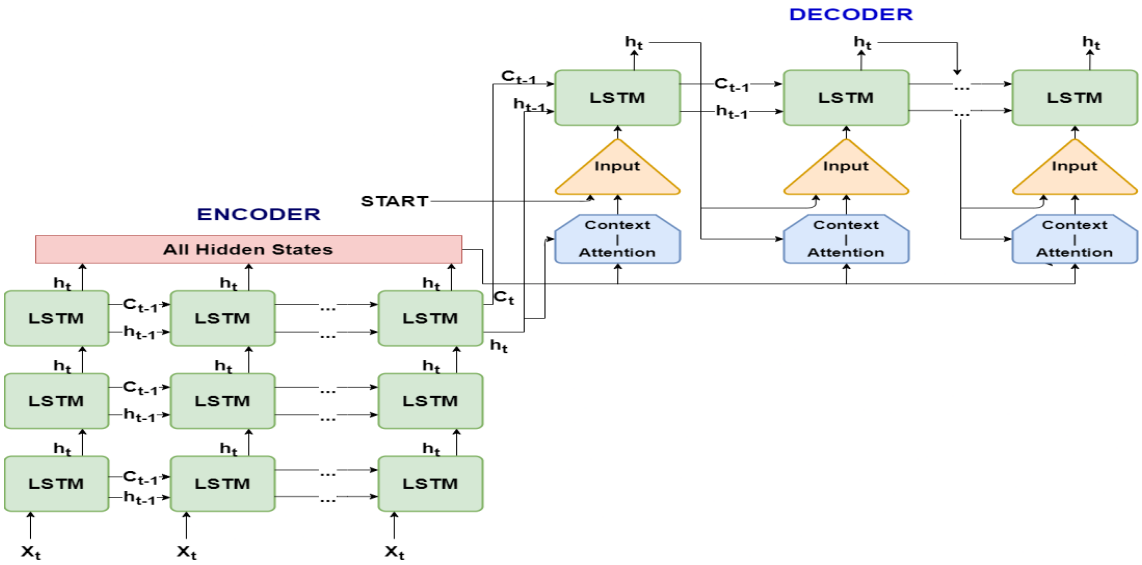


**Fig. 2** Base Abstractive Summarizer Architecture

### *4.2.2 Word Embeddings*

Word Embeddings can be considered to be the mathematical representation of words in a vectorized form, that portrays the semantic meaning of the word. The concept is based on the intuition that words that occur in similar contexts tend to have relatable meanings. The word embeddings used for the textual entailment model, the GloVe [28] (Global Vectors) Embeddings, is a count-based, unsupervised learning model that models the vector representations of words at a global level using co-occurrence data (how frequently two words appear together). It is built on word-word co-occurrence matrix probability ratios, which combines the intuitions of count-based models with the linear structures captured by embedding approaches like word2vec.

As demonstrated in Equation 11, GloVe employs a global log-bilinear (LBL) regression model that leverages a simple weighted least squares algorithm.

$$w_i.w_j = \log P(i|j) \tag{11}$$

### *4.2.3 Experimental Setup*

The pre-trained GloVe embeddings have been provided by Stanford for NLP research. For better results, 300-dimensional GloVe vectors were used in the embedding layer of the textual entailment model.

After the word embeddings were invoked, the sentences and words were stored along with their indexing in a specialized class called vocabulary. The vocabulary would contain all the words used in the cleaned text corpus and the test dataset and will be assigned a hash based on when they are inculcated into the vocabulary class. These hashed words and sentences were then used by the word vectors to generate similarities between the different words by referring to their indexes (since strings cannot be understood by the systems). The vocabulary class hashing and searching were done by exploiting CUDA libraries that used the full potential of the local graphics processing unit for maximum speed.

On the modification of a typical text entailment model, our model, depicted in Figure 3 , was made to use Bidirectional LSTM cells [12][32].

The bi-LSTM model made use of a ReLU layers, which applied the Rectified Linear Unit function [1] in an element-wise manner.

Instead of the more widely used tanh functions, we used ReLU because ReLU offers sparsity in activation, which means that it can produce more zero-valued activations compared to tanh. This sparsity property can be beneficial for certain applications, such as text summarization, where not all words or phrases contribute equally to the final summary. Furthermore, ReLU is computationally more efficient compared to tanh activation. ReLU simply sets negative values to zero, whereas tanh involves more complex calculations. This efficiency can be crucial in scenarios where computational resources are limited, which was our case as well.

The model also contained a dropout layer [34] to restrict overfitting. There were three fully connected concatenation layers being implemented to trickle down the size of the matrices after each passing layer. Finally an output layer was used to take the matrix as input and output 3 values corresponding to the final class scores: entailment, neutral and contradiction

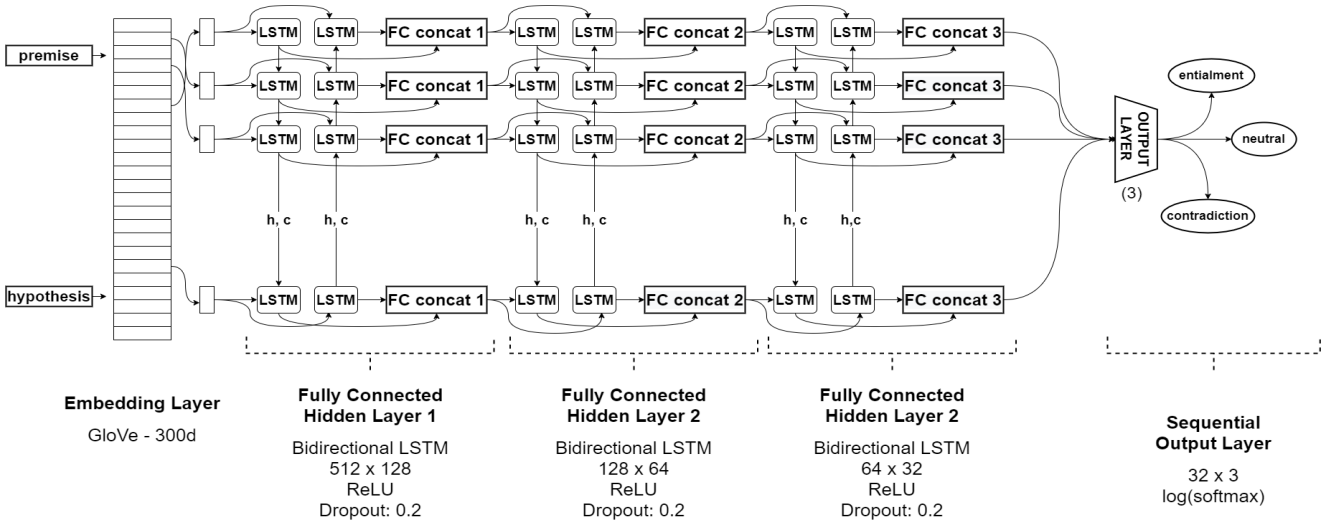


**Fig. 3** Entailment Model Architecture

The defined layers of the model were then put into a Sequential layer, which is basically a layer that combines all the defined layers in a structured sequential manner to ultimately make the pipeline of layers for the textual entailment model. The model architecture is depicted in Figure 3.

The textual entailment model made use of two forward propagation functions, one calling the other on initialization. The first forward function, 'forward_once' initializes the embedding Layer with the GloVe word vectors for the word-indexes in the vocabulary class for the neural network, as it is the first layer.

$$premise := forward\_once(premise, (h0, c0))$$

$$hypothesis := forward\_once(hypothesis, (h0, c0))$$

$$forward \rightarrow Sequential\_layer(combined\_output)$$

Given as input the token indexes in the concatenation of premise and hypothesis, the embeddings obtained after the first layer, initialized with Glove embeddings, are processed by the three bidirectional LSTM.

#### *4.2.4 Training*

With the testing and training datasets loaded, vocabulary classes spawned and the textual entailment model defined, the hyperparameters were fine-tuned before training for optimal results. Batches of 32 sentence pairs were loaded into the model, GloVe-embedded words using an embedding size of 300 and the padded sequences were propagated into the sequential layer. The training ran for 15 epochs in a local system comprising of a GTX 1050 GPU with 4GB VRAM, 16GB of primary RAM, and an Intel i7 - $7^{th}$ generation processor. The Cross-Entropy criterion was used as the loss function.

## 5 Results

The seq2seq base abstractive summarizer and the Bart-based summarizer have been used to generate summaries for 100 text samples from the test data. The generated sentences have been compared with the original summaries using the standard evaluation metrics. The average scores are enlisted in Table 1. Additionally, the pre-trained 'stsb-roberta-large' sentence transformer [36] has been used to measure the BERTScore score between the original summary and predicted summary. A higher semantic similarity score for the Bart-based model was achieved, as expected since it was a pre-trained transformer-based model optimized and fine-tuned on a much larger dataset for a longer period of time.

A typical entailment model having vanilla RNN layers was noted to have a training loss of 0.8754 and a training accuracy of 60.14%. It was also noted to have a validation loss of 0.8718 resulting in a validation accuracy of 61.04%. It had a similar test statistic record of 60% accuracy and 0.87 loss. After training the entailment model having bi-LSTM layers for 15 epochs, the training loss was reduced to 0.7113 and a training accuracy of 70.53%, recording a 10.39% increase in accuracy from the typical model was achieved. It also yielded a validation loss of 0.6588 pointing to a validation accuracy of 73.20%, recording a decent 12.16% increase in accuracy from the base model. On the test set too, a testing loss of 0.6575 was recorded along with a test accuracy of 73.21%, which is big step up from the base model test accuracy of around 60%. A higher validation accuracy can be attributed to the smaller validation domain and a low variance among training and validation sets, meaning a more robust model when it comes to validation and testing. This is also attributed to the existence of dropouts (20% dropout was being implemented) so as to prevent overfitting. The en-

| | BLEU | BERTScore | ROUGE-1 F-score | ROUGE-2 F-score | ROUGE-L F-Score |
|---|---|---|---|---|---|
| **Base Model** | 0.754882 | 0.391581 | 0.22535 | 0.071559 | 0.219551 |
| **BART based pre-trained model** | 0.654945 | 0.556738 | 0.267859 | 0.079555 | 0.23525 |

**Table 1** Standard Evaluations Metrics

tailment model was then tested on the dedicated test dataset provided by SNLI itself. The training results are shown in Table 2.

| | Accuracy | Loss |
|---|---|---|
| **Training Set** | 0.7053 | 0.7113 |
| **Validation Set** | 0.7320 | 0.6588 |
| **Test Set** | 0.7321 | 0.6575 |

**Table 2** Model Training Statistics

Now, while using the text entailment model to evaluate the generated summaries both from the base seq2seq model and the transformer model, two scoring metrics were devised from the bi-LSTM based entailment model to get a better understanding of the sentence pairs' entailment scores. 100 rows of test data were generated from the base summarizer and the BART-based summarizer containing the original texts, original (reference) summaries, and predicted summaries. Various results are recorded for both summary-based entailment and mutual entailment.

## 5.1 Summary-based Entailment

The Entailment score of the generated summary was calculated with respect to the text body whose summary was generated. In other words, the entailment between the text and the generated summary was evaluated. The raw softmax() values (raw tensor values from the softmax() function) of each class (entailment, neutral, and contradiction) were extracted before applying argmax() and were then normalized so as to get an idea as to how much entailment, neutrality or contradiction was present in the summarized sentence with respect to the original text (premise) using Equation 12.

$$normal_i = \frac{prediction_i}{max(predictions) - min(predictions)} \quad (12)$$

### *5.1.1 Base summarizer model*

After applying argmax() on the output probabilities of the base seq2seq summarizer, we found that 41 sentence pairs were identified to have $Entailment$, 40 sentence pairs have $Neutral$ and 19 sentence pairs were identified to be $Contradictions$. Therefore, our Entailment-based evaluator could mark the base seq2seq based summarizer to have 41% Entailment, which was decent, but not groundbreaking.

### *5.1.2 BART-based summarizer model*

After applying argmax() on the output probabilities of the BART-based summarizer, we found that 89 sentence pairs were identified to have $Entailment$, 10 sentence pairs have $Neutral$ and only 1 sentence pair was identified to be Contradictions. Therefore, our Entailment-based evaluator could mark the BART-based summarizer to have a 89% Entailment score, which is a notable score for any well-trained summarizer.

### *5.1.3 Summarizer model comparison*

From the summary-based Entailment categorizations observed, we can clearly come to the conclusion that on a premise text to hypothesis summary form of entailment training, the BART summarizer model simply outperforms the base seq2seq summarizer model. A comparison table is shown Table 3.

| | Base seq2seq model | BART model |
|---|---|---|
| **Entailment** | 41 | 89 |
| **Neutral** | 40 | 10 |
| **Contradiction** | 19 | 1 |

**Table 3** Summary-based Entailment Scores

## 5.2 Mutual Entailment

A two-way Entailment score was also calculated between both the reference summary and the predicted summary. This means that a classification score list was generated where the reference summary was taken as premise and the predicted summary was taken as the hypothesis and vice versa.

Four Categories were classified from the two sets of Labels (label of premise and label of hypothesis)

1. **Perfect Entailment**:
   If both the premise → hypothesis and hypothesis → premise were entailing each other, this meant both the

reference summary and the predicted summary were entailing each other, and hence were considered to have full or perfect entailment.

2. **Partial Entailment**:
   If neither one of the summaries was a contradiction of the other, excluding the case of perfect entailment, these sentence pairs were categorized as having Partial Entailment.
3. **Perfect Contradiction**:
   If both the premise and the hypothesis were a contradiction of each other, it was termed as a contradiction.
4. **Mutually Neutral**:
   The rest of the sentence pairs, more predominantly having the hypothesis as a neutral/contradiction of the premise and vice versa (but both weren't neutral or contradiction of each other simultaneously) were categorized to be under the class of Mutually Neutral. This is because, although one may be the contradiction of the other, we cannot fully commit to the categorization of negative entailment, as it is an experimental metric with a very high case of this happening naturally.

On a sentence pair level analysis, two categories were given as output once taking the reference summary as premise and the predicted summary as hypothesis, and in the next scenario the vice versa. Depending on the two scores, one of the four discussed categories were assigned to that particular sentence pair, and depending on this category, a $MutualEntailment$ score was conjured, which is later used on generating the overall Entailment Metric Score complementing with the summary-based entailment category score as well.

### *5.2.1 Base summarizer model*

After evaluation, we found that there were only 7 perfectly entailed sentence pair, where the [reference summary → predicted summary] label and [predicted summary → reference summary] label was Entailment. There were 56 partially entailed sentence pairs, where either the [reference summary → predicted summary] label or the [predicted summary → reference summary] label was Entailment/Neutral, or both cases were neutral to each other. There was only 1 Perfectly Contradictory sentence pair, where the [reference summary → predicted summary] label and the [predicted summary → reference summary] label were both contradictions (pure contradictions).

### *5.2.2 BART-based summarizer model*

Entailment results of 100 test cases generated from the BART abstractive text summarizer were also analyzed. It yielded interesting results. We found that there were 13 perfectly entailed sentence pair, where the [reference summary → predicted summary] label and [predicted summary → reference summary] label was Entailment. There were 66 partiall entailed sentence pairs, where either the [reference summary → predicted summary] label or the [predicted summary → reference summary] label was Entailment/Neutral, or both cases were neutral to each other. There were 0 Perfectly Contradictory sentence pairs, where the [reference summary → predicted summary] label and the [predicted summary → reference summary] label were both contradictions (pure contradictions).

### *5.2.3 Summarizer model comparison*

From the mutual Entailment categorizations observed, we can clearly come to the conclusion that on a reference summary to predicted summary form of entailment training, the BART summarizer model simply outperforms the base seq2seq summarizer model. Although the base model performed well with only 1 perfect contradiction, the BART model simply surpassed the base model with 0 perfect contradictions. A comparison table is shown in Table 4:

| | **base seq2seq model** | **BART model** |
|---|---|---|
| **Perfect Entailment** | 7 | 13 |
| **Partial Entailment** | 56 | 66 |
| **Mutually Neutral** | 36 | 21 |
| **Perfect Contradiction** | 1 | 0 |

**Table 4** Mutual Entailment Scores

Based on the above mutual scores and summary scores, we attain the line graphs that are shown in Figures 4 and 5. We can clearly see that the BART model dips very little compared to the base model in both the metric systems. In case of the mutual entailment scores, the values are figuratively similar, but we can see a large dip in score to as low as -1.00 [The scoring system for both the entailment paradigms have been explained thoroughly in the next section]. Hence, even in this case, the BART model performs better.

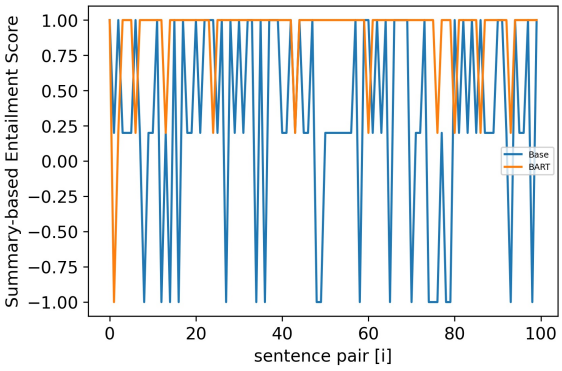


**Fig. 4** Summary-based entailment score

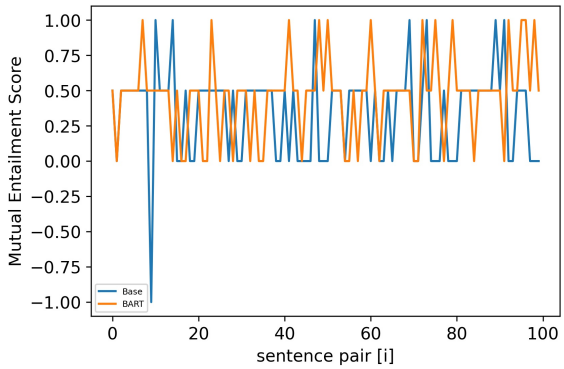


**Fig. 5** Mutual entailment score

### 5.3 Final Entailment Metric Score

On analysis of both the summary-based and mutual entailment paradigms, an accumulated final entailment score was more or less arbitrarily devised from both the above mentioned metric systems. The mutual perspective being a more experimental approach, a weightage of 0.30 was given to the results of this section while the basic summary-based entailment was given a higher priority of 0.70 while calculating the final entailment metric score. The Scoring protocols are mentioned below:

$$FEMS = 0.70(E_S) + 0.30(E_M) \tag{13}$$

Where, $E_S$: Summary-based Entailment, $E_M$: Mutual Entailment.

Here, in the Summary based Entailment, the categorical scoring are:

- Entailment: 1
- Neutral: 0.2
- Contradiction: -1

A partial advantage is given to neutral sentences. This is because Neutral Sentences are simply not an entailment of the given sentence, but that doesn't mean it is a total contradiction of the sentence. Neutrality also signifies that the entailment is not strong enough for the correlation to exist, but is good enough for a user to match the two as somewhat similar sets of lexicons.

Now, in the Mutual Entailment, the categorical scoring are:

- Perfect Entailment: 1
- Partial Entailment: 0.5
- Mutually Neutral: 0
- Perfect Contradiction: -1

Therefore, for a given premise text, a summary will be generated. Then the text, reference summary and predicted summary will be fed into the text entailment model. Here, the Summary based entailment category (SE) will be predicted, and then the Mutual Entailment Category (ME) will be predicted. Then, both these categorical scores will be used to calculate the final entailment metric score (FEMS) 13 to give a vivid idea as to how good the generated summary is.

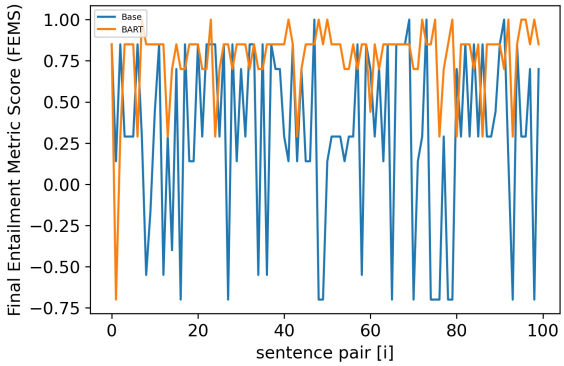


**Fig. 6** Final Entailment Metric Scores

The FEMS vary from as low as -1.00 to as high as +1.00. The FEMS of all the generated summaries were averaged out for both the base model summaries and the transformer generated summaries and were compared to each other. The findings are visualized in Figure 6 and recorded in Table 5.

From Table 5 and Figure 6, we can clearly summarize that the BART transformer model not only has a higher average FEMS of 0.768 compared to the base model's FEMS of 0.312, but is also much more consistent with its good entailment results with a standard deviation of just 0.234 compared to the base model's 0.545 standard deviation value. Thus, from these findings, it can be safely concluded that the BART-based transformer outperforms the base summarizer model.

## 6 Conclusion

Being one of the major NLP tasks presently being tackled by many, automatic evaluation of deep learning-based text Summarization is in its development phases. To "summarize", in our work a base seq2seq abstractive text summarizer model was developed to summarize news articles. A BART transformer-based model was also used, and test sets from both summarizers were used for the analysis of the textual entailment model's performance of evaluation. The model was trained on the standard SNLI dataset provided by Stanford. A summary-based Entailment result was recorded on 100 transformer-generated summaries along with the Mutual Entailment based score between the original and generated sum-

| | FEMS mean | FEMS median | FEMS standard deviation |
|---|---|---|---|
| **Base summarizer model** | 0.312 | 0.29 | 0.545 |
| **BART summarizer model** | 0.768 | 0.85 | 0.234 |

**Table 5** FEMS evaluations

maries. The same was also done for the seq2seq summarizer-generated summaries. We finally came to the conclusion that the pre-trained BART transformer-based summarizer performed objectively better in generating more accurate abstractive summaries, as was expected.

Throughout the experimentation and research on this field, many shortcomings were noticed and a lot of potential for improvements were recorded so as to better tackle this problem in further publications. For starters, to have a better grasp of the reference summary from where to evaluate our model by in case of mutual entailment, the Inshorts dataset of news and summary was used instead of the standard CNN/Daily Mail dataset more prevalent in this sort of NLP tasks. In future endeavors, this can be easily tackled in order to get a better result of the summaries. The entailment model made from scratch too has a lot of room for improvement. It does not make use of an extensive Attention Layer, which has newly become very popular in the NLP field due to its efficiency to handle complexities of higher degrees. Due to shortcomings in our processing power, this attention mechanism was omitted out but is mentioned for future endeavors in this problem. The aim of the experiment, although values the importance of accuracies in training and testing, was not solely built on the premise of exceeding the SOTA models already prevalent in the NLP domain. Thus, although the entailment model performs better than most models present rooting to naive probabilistic approaches or other such approaches (that have been mentioned above, that mostly require hard coding lexical rules and logic), it does not stand out among the baseline Deep Learning-based models mentioned in the SNLI projects, as most of them use high complexity transformers or use attention mechanisms. There is potential in this model as it can inculcate attention mechanisms and also use transfer learning to yield better results, but for now, a 73% accuracy is fairly decent. Moreover, for better testing, the efficiency of the entailment model, a rigorous round of RTE test datasets could be used. RTE is standard when it comes to tackling the text entailment problem, hence for future research and experimenting, RTE can be surely used along with the SNLI corpus. Also, with higher performance GPUs and processors, higher complexities of matrix multiplications can be done in a shorter time, and this will drastically help with preventing the OOV problem (Out Of Vocabulary), where different vocabulary classes were created to cater to different testing paradigms, instead of making one giant vocabulary class. This was done to make the word indexes comparatively smaller numbers, so as to ease up on calculations (Otherwise we found ourselves crashing the CUDA ports entirely, and the entire model had to be retrained).

With that said, the entailment performs close to human judgement. The BART-based summaries were more meaningful than the base summarizer-generated summaries, as expected. The FEMS was found to be much more reliable than other metrics such as ROUGE and BLEU. This is because BLEU and ROUGE do not take into account semantic complexities. Due to the usage of LSTMs, important contexts are remembered by the entailment model and it portrays the relationships between a text and the corresponding summary with better efficiency.

The objective of the work was to basically exhibit the potential of textual entailment to become one of the factors for an efficient evaluation of abstractive text summarization systems. There are tremendous future scopes. Due to limited resources, relatively smaller and simpler architectures were trained on a few epochs in this work. Thus, future works could use more sophisticated models to deliver further promising results and derive more mathematically appropriate expressions to use the entailment results. This may not solely be the best evaluation method for such systems, however, using it in conjunction with other transformer-based metrics like the BERTScore, and other conventional metrics shall certainly lead to a highly efficient method that may become comparable to human evaluations.

**Acknowledgements** The work presented here falls under the research project entitled “Deep Summarization Evaluation”, Sanction Order No: IFC/4130/DST-CNRS/2018-19/IT25 (DST-CNRS). The authors are thankful to the Center for Natural Language Processing and Dept. of Computer Science & Engineering, NIT Silchar, for providing the infrastructure to conduct the experiments related to the work. We also acknowledge the collaboration with Laboratoire d’Informatique et Systèmes (LIS), Aix-Marseille University, CNRS for this work.

## Conflict of interest

The authors declare that they have no conflict of interest.

## References


1. Agarap, A.F.: Deep learning using rectified linear units (relu) (2018)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate (2014)
3. Barzilay, R., Lee, L.: Catching the drift: Probabilistic content models, with applications to generation and summarization (2004)
4. Bhaskar, P., Pakray, P.: Automatic evaluation of summary using textual entailment. In: Proceedings of the Student Research Workshop associated with RANLP 2013, pp. 30–37 (2013)
5. Camburu, O.M., Rocktäschel, T., Lukasiewicz, T., Blunsom, P.: e-snli: Natural language inference with natural language explanations (2018)
6. Chollet, F.: keras. https://github.com/fchollet/keras (2015)
7. Edmundson, H.P.: New methods in automatic extracting. Journal of the ACM (JACM) **16**(2), 264–285 (1969)
8. Falke, T., Ribeiro, L.F., Utama, P.A., Dagan, I., Gurevych, I.: Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 2214–2220 (2019)
9. Fan, A., Gardent, C., Braud, C., Bordes, A.: Using local knowledge graph construction to scale seq2seq models to multi-document inputs (2019)
10. Gupta, A., Kaur, M., Mirkin, S., Singh, A., Goyal, A.: Text summarization through entailment-based minimum vertex cover. In: Proceedings of the Third Joint Conference on Lexical and Computational Semantics (* SEM 2014), pp. 75–80 (2014)
11. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. Advances in neural information processing systems **28**, 1693–1701 (2015)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Kryściński, W., McCann, B., Xiong, C., Socher, R.: Evaluating the factual consistency of abstractive text summarization (2019)
15. Kupiec, J., Pedersen, J., Chen, F.: A trainable document summarizer. In: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 68–73 (1995)
16. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension (2019)
17. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out, pp. 74–81 (2004)
18. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.: Generating wikipedia by summarizing long sequences (2018)
19. Lloret, E., Ferrández, O., Munoz, R., Palomar, M.: A text summarization approach under the influence of textual entailment. In: NLPCS, pp. 22–31 (2008)
20. Luhn, H.P.: The automatic creation of literature abstracts. IBM Journal of research and development **2**(2), 159–165 (1958)
21. Maynez, J., Narayan, S., Bohnet, B., McDonald, R.: On faithfulness and factuality in abstractive summarization (2020)
22. Mihalcea, R.: Graph-based ranking algorithms for sentence extraction, applied to text summarization. In: Proceedings of the ACL interactive poster and demonstration sessions, pp. 170–173 (2004)
23. Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al.: Abstractive text summarization using sequence-to-sequence rnns and beyond (2016)
24. Padó, S., Cer, D., Galley, M., Jurafsky, D., Manning, C.D.: Measuring machine translation quality as semantic equivalence: A metric based on entailment features. Machine Translation **23**(2-3), 181–193 (2009)
25. Padó, S., Galley, M., Jurafsky, D., Manning, C.D.: Robust machine translation evaluation with entailment features. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 297–305 (2009)
26. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pp. 311–318 (2002)
27. Pasunuru, R., Bansal, M.: Multi-reward reinforced summarization with saliency and entailment (2018)
28. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543 (2014)
29. Radev, D.R., Jing, H., Styś, M., Tam, D.: Centroid-based summarization of multiple documents. Information Processing & Management **40**(6), 919–938 (2004)
30. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization (2015)
31. Saini, N., Saha, S., Bhattacharyya, P., Tuteja, H.: Textual entailment–based figure summarization for biomedical articles. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **16**(1s), 1–24 (2020)
32. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing **45**(11), 2673–2681 (1997)
33. Sellam, T., Das, D., Parikh, A.P.: Bleurt: Learning robust metrics for text generation (2020)
34. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
35. Tatar, D., Mihis, A., Lupsa, D., Tamaianu-Morita, E.: Entailment-based linear segmentation in summarization. International Journal of Software Engineering and Knowledge Engineering **19**(08), 1023–1038 (2009)
36. Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45 (2020)
37. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert (2019)