Text Summarizing and English to Hindi translation

Anirudh Ambati
CSE
IIIT Sri City
Email: anirudh.a17@iiits.in

Katta Venkata Harsha
CSE
IIIT Sri City
Email: venkataharsha.k17@iiits.in

Ankur Gupta
CSE
IIIT Sri City
Email: ankur.g17@iiits.in

***Abstract*—Our project is an idea to summarise texts and show that to people in their native languages. Summarise a news report and give a short news bullet to the users in their language. Here, particularly we are providing summary in Hindi to an English content. We have tried a two stage method here for that. one, is to summarise an English text and second, is to translate it to Hindi**

1. Introduction

Most content in English or any other language is not readable/ understandable to those from other languages. A native Hindi speaker without knowledge of English will not be able to read news from English sources. He will miss out on a lot of important news and information because of this. So, our tool will help them to read quick summary of a news article or a good book.

To achieve this we have used a two stage model. Stage 1 is text summarizing. This is a pretty standard model English text to English summary. The second stage is English-Hindi translation. This stage will convert English summary to in Hindi. In between these two stages, a python based grammar checker is used to improve the translation correctness.

2. State of the art/Background

Text summarization methods can be classified into extractive and abstactive summarization.There are two different groups of text summarization. Inducative and Informative. Indicative summarization gives the main idea of the text to the user. The length of this summarization is around 5 percent of the given text. The informative summarization system gives brief information of the main text. The length of the informative summary is around 20 percent of the given text. [1]

2.1. Abstract Summarisation

Abstractive Summarization is a method for novel phrasing describing the content of the text which requires heavy machinery from natural language processing, including grammars and lexicons for parsing and generation.There is a lot of research going in this area and we have referred many papers on this.Most of the abstractive summarization methods follows the below described process. [2]

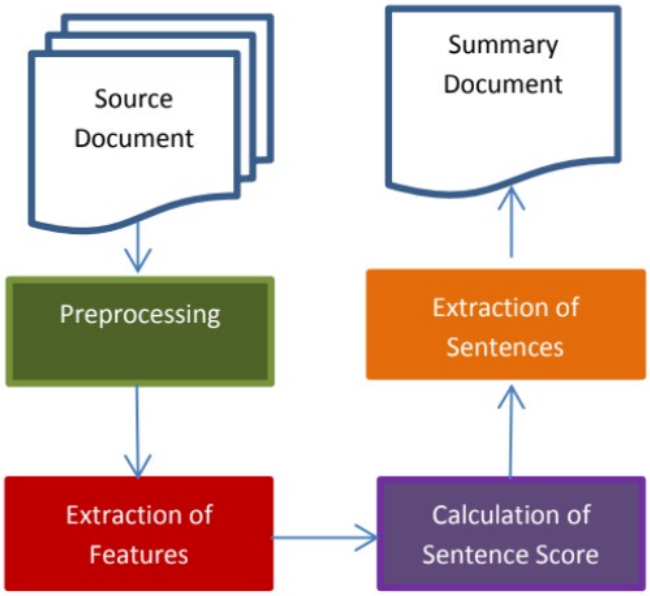


Figure 1. Flowchart of Abstractive summarization

2.2. Extractive Summarisation

Extractive Summarization is a method for determining salient text units (typically sentences) by looking at the text unit's lexical and statistical relevance or by matching phrasal patterns. Extractive Summaries are formulated by extracting key text segments such as sentences or passages from the text, based on statistical analysis of individual or mixed surface level features. [3]

2.3. SUMMARIZATION METHODS

2.3.1. Query Based and Generic Summarization. In query based text summarization the scoring of the sentences of a given document is based on the frequency counts of words or phrases. Higher scores are given to the sentences

containing the query phrases rather than the ones with single query words. The sentences with highest scores are then extracted for the output summary together with their structural context. Portions of text may be extracted from different sections or subsections. The resulting summary is the union of such extracts. In the sentence extraction algorithm, whenever a sentence is selected for the inclusion in the summary, some of the headings in that context are also selected.
Algorithm:
1: Rank all the sentences according to their score.
2: Add the main title of the document to the summary.
3: Add the first level-1 heading to the summary.
4: While (summary size limit not exceeded)
5: Add the next highest scored sentence.
6: Add the structural context of the sentence: (if any and not already included in the summary)
7: Add the highest level heading above the extracted text (call this heading h).
8: Add the heading before h in the same level.
9: Add the heading after h in the same level.
10: Repeat steps 7, 8 and 9 for the next highest level headings.
11: End while [4]

2.3.2. Bayesian Classifier. For each sentence s we compute the probability it will be included in a summary S given the k features fj;j = 1...k, which can be expressed using Bayes' rule as follows: [5]

$$P(s \in S \mid f1, f2, \ldots, fk) = \frac{P(f1,f2,\ldots,fk \mid s \epsilon S)\ P(s \epsilon S)}{P(f1,f2,\ldots,fk)}$$

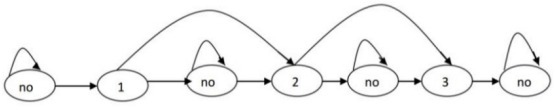
Assuming statistical independence of the features:

$$P(s \in S \mid f1, f2, \ldots, fk) = \frac{\prod_{j=1}^{k} P(f_j \mid s \in S)\, P(s \in S)}{\prod_{j=1}^{k} P(fj)}$$

Figure 2. Formula used to calculate

2.3.3. Hidden Markov Model. The main idea is using a sequential model to account for local dependencies between sentences. In Hidden Markov Model, three features were used:
1)position of the sentence in the document.
2)number of terms in the sentence.
3)likeliness of the sentence terms given the document terms. [6]

Figure 2: Hidden Markov Model to Extract Three Summary Sentences from a Document.



2.3.4. Fuzzy Logic Based Text Summarization. The system involve in following steps:
[7] 1)Perused the source archive into the system.
2)In pre-processing step, the system extracts the individual sentences of the original documents. After, separate the input document into individual words. Next, remove stop words. The last step for pre-processing is word stemming.
3)Each sentence is related in vector of eight features that mentioned in Section II, whose values are obtained from the content of the sentence
4)The features are calculated to obtain the sentence score base on fuzzy logic method.
5)A set of highest score sentences are extracted as document summary based on the compression rate. [8]

2.4. Bidirectional Encoder Representations from Transformers(BERT)

Text summarisation with pretrained encoders.Represents the latest incarnation of pretrained language models which have recently advanced.They introduced a document level encoder based on BERT which is able to express the semantics of a document and obtain representations for its sentences.Their extractive model is built on top of this encoder by stacking several inter-sentence Transformer layers. For abstractive summarization,they proposed a new fine-tuning schedule which adopts different optimizers for the encoder and the decoder as a means of alleviating the mismatch between the two (the former is pretrained while the latter is not). They also demonstrate that a two-staged fine-tuning approach can further boost the quality of the generated summaries. [9]

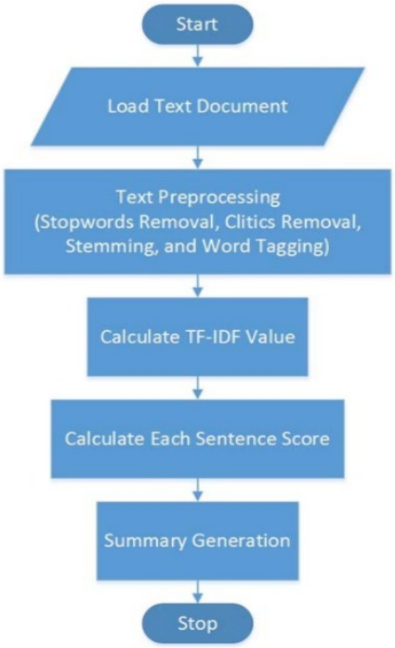
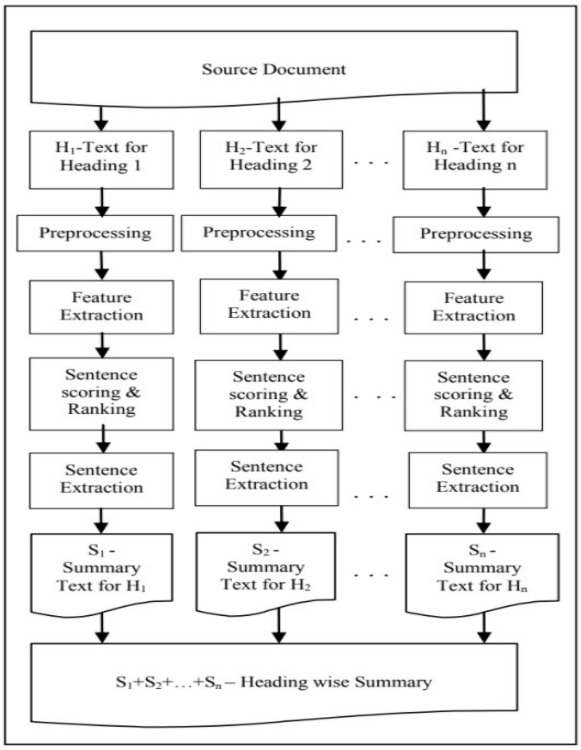


Figure 3 Flowchart of Automatic Summarization

2.5. Automatic Text Summarization by Local Scoring and Ranking for Improving Coherence

It summarizes the given input document using local scoring and local ranking that is it provides heading wise summary. Headings of a document give contextual information and permit visual scanning of the document to find the search contents. This approach applies the same features to all document sentences. But it ranks the sentences heading wise and selects top n sentences from each heading where n depends upon compression ratio. The final heading wise summary produced by this approach is a collection of summary of individual headings. Since the heading wise summary contains the equal proportion of sentences from each heading, it reduces the coherent gap of the summary text. Also it improves the overall meaning and understanding of the summary. [10]



3. Proposed System

3.1. Text Summarisation

The proposed system uses a bidirectional LSTM based summarisation model for english to english text summarisation. Word2vec algorithm skipgram is used in the encoder input. This is achieved by training a neural network to predict context words given a current word. after training the hidden layer is used as the embedding layer. embedding size was kept at 128. A trained skipgram was used in this situation.

For the decoder input and output, one hot encoding of the summary words was used. vocabulary size is 30,000. One hot encoding was used for being able to add a attention layer too.

We use a bidirectional encoder lstm with dropout=0.2, tanh activation. The Decoder is an unidirectional lstm with size = 128, dropout = 0.2 and a softmax activation.

In addition to an LSTM network, Attention is used in this model, so as to focus on the important aspects of the model. An attention layer is in between the encoder and the decoder over the source sequence's hidden states. As skipgram embedding and the one-hot vector sizes are not the same, PCA is used over embedding to allow multiplication with one-hot vectors to get attention weights and vectors in a same dimensions. final prediction of output word in decoder sequence is done by the attention layer. It helps allow the decoder individual encoder state information.

3.2. Spelling and grammar check

The output of the summarisation text is prone with spelling and grammatical errors. When this text is sent to translation, the output of that would be prone to mistakes as the errors of the first model are amplified. Hence, a simple yet powerful spelling and grammar checker python library 'language-check' is used. This will help the summary provided to be grammatically sound and easier for the translation model.

3.3. English to Hindi Text translation

```
[KEY: > input, = target, < output]

> he is painting a picture .
= वह एक चित्र बना रहा है .
< वह एक चित्र बना रहा है .

> why not try that delicious wine ?
= क्यों उस स्वादिष्ट शराब की कोशिश नहीं की?
< क्यों उस स्वादिष्ट शराब की कोशिश नहीं की?

> she not not a poet but a novelist .
= वह एक कवि नहीं बल्कि एक उपन्यासकार हैं।
< वह एक कवि नहीं बल्कि एक उपन्यासकार हैं।

> you re all alone .
= तुम भी पतली हो।
< तुम भी पतली हो।
```

translate from English to Hindi. is made possible by the simple but powerful idea of the sequence to sequence network [11], in which two recurrent neural networks

work together to transform one sequence to another. An encoder network condenses an input sequence into a vector, and a decoder network unfolds that vector into a new sequence. To improve upon this model we'll use an attention mechanism [12], which lets the decoder learn to focus over a specific range of the input sequence.

3.3.1. Data for Translation. In query based text summarization the scoring of the sentences The data for this project is a set of many thousands of English to Hindi translation pairs.

```
I am cold.  मुझे ठंड लग रही है।
```

We are using a subset of IIT Bombay English-Hindi Corpus and took 2867 sentence pairs.
Counted words:
English - 2520
Hindi - 3094

The full process for preparing the data is:
1) Read text file and split into lines, split lines into pairs.
2) Normalize text, filter by length and content.
3) Make word lists from sentences in pairs.

3.3.2. The Seq2Seq Model. A Recurrent Neural Network, or RNN, is a network that operates on a sequence and uses its own output as input for subsequent steps.

A Sequence to Sequence network [11], or seq2seq network, or Encoder Decoder network [13], is a model consisting of two RNNs called the encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder reads that vector to produce an output sequence.

Unlike sequence prediction with a single RNN, where every input corresponds to an output, the seq2seq model frees us from sequence length and order, which makes it ideal for translation between two languages.

Consider the sentence
"मैं काली बिल्ली नहीं हूं" → "I am not the black cat"

Most of the words in the input sentence have a direct translation in the output sentence, but are in slightly different orders, e.g
"काली बिल्ली" and "black cat".

Because of the "am not" construction there is also one more word in the input sentence. It would be difficult to produce a correct translation directly from the sequence of input words.

With a seq2seq model the encoder creates a single vector which, in the ideal case, encodes the "meaning" of the input sequence into a single vector — a single point in some N dimensional space of sentences.

3.3.3. The Encoder. The encoder of a seq2seq network is a RNN that outputs some value for every word from the input sentence. For every input word the encoder outputs a vector and a hidden state, and uses the hidden state for the next input word.

3.3.4. The Decoder-Simple Decoder. The decoder is another RNN that takes the encoder output vector(s) and outputs a sequence of words to create the translation. In the simplest seq2seq decoder we use only last output of the encoder. This last output is sometimes called the context vector as it encodes context from the entire sequence. This context vector is used as the initial hidden state of the decoder. At every step of decoding, the decoder is given an input token and hidden state. The initial input token is the start-of-string SOS token, and the first hidden state is the context vector (the encoder's last hidden state).

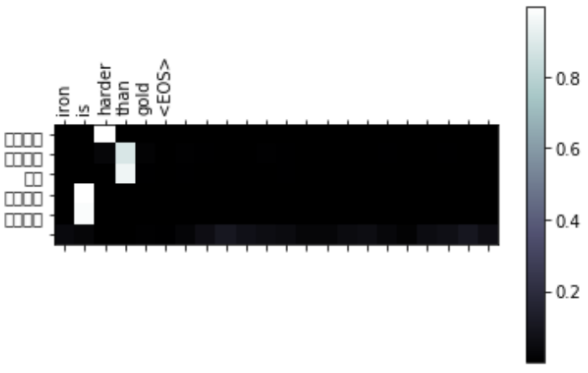
3.3.5. The Decoder-Attention Decoder. If only the context vector is passed between the encoder and decoder, that single vector carries the burden of encoding the entire sentence.

Attention allows the decoder network to "focus" on a different part of the encoder's outputs for every step of the decoder's own outputs. First we calculate a set of attention weights. These will be multiplied by the encoder output vectors to create a weighted combination. The result should contain information about that specific part of the input sequence, and thus help the decoder choose the right output words.

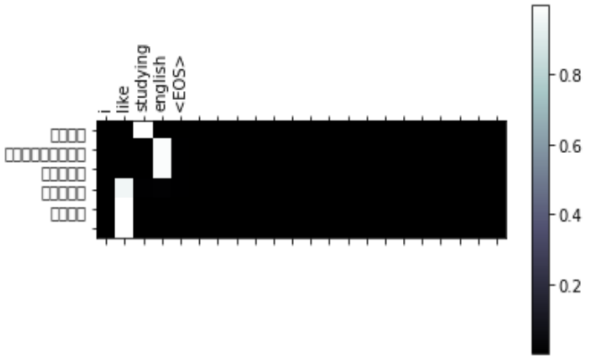
Calculating the attention weights is done with another feed-forward layer attn, using the decoder's input and hidden state as inputs. Because there are sentences of all sizes in the training data, to actually create and train this layer we have to choose a maximum sentence length (input length, for encoder outputs) that it can apply to. Sentences of the maximum length will use all the attention weights, while shorter sentences will only use the first few.

3.3.6. Evaluation. Evaluation is mostly the same as training, but there are no targets so we simply feed the decoder's predictions back to itself for each step. Every time it predicts a word we add it to the output string, and if it predicts the EOS token we stop there. Decoder's attention visualizations are displayed below.

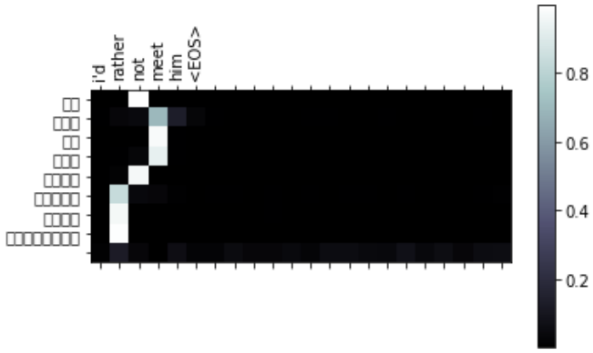
```
input = iron is harder than gold
output = लोहा सोने से सख्त होता
```



input = i like studying english
output = मुझे अंग्रेज़ी पढ़ना अच्छा लगता



input = i'd rather not meet him
output = हो सके तो मैं उससे मिलना नहीं चाहूँगी।



4. Results

Some Example translations

```
input = i'd rather not meet him
output = हो सके तो मैं मिलना था
input = this is big
output = यह बहुत बड़ा है।
input = i like studying english
output = मुझे अंग्रेज़ी में अंग्रेज़ी पढ़ने का मन है।
input = iron is harder than gold
output = लोहा सोने से सख्त होता है।
input = i am happy
output = मैं खुश में खुश हूँ।
```

Some results shown below with their BLEU scores at the last and their corresponding Cumulative 1-gram, 2-gram, 3-gram, 4-gram scores. Many were having perfect BLEU Score of 1.0,

```
English:  i have to put the baby to bed
Actual:  मुझे बच्चे को सुलाना है।
Predicted:  मुझे बच्चे को सुलाना है।
Cumulative 1-gram: 1.000000
Cumulative 2-gram: 1.000000
Cumulative 3-gram: 1.000000
Cumulative 4-gram: 1.000000
1.0
```

```
English:  how many sandwiches are there left
Actual:  कितने सैंडविच बचे हैं
Predicted:  कितने सैंडविच बचे हैं
Cumulative 1-gram: 1.000000
Cumulative 2-gram: 1.000000
Cumulative 3-gram: 1.000000
Cumulative 4-gram: 1.000000
1.0
```

```
English:  whose book is this
Actual:  यह किसकी किताब है
Predicted:  यह किसकी किताब है
Cumulative 1-gram: 1.000000
Cumulative 2-gram: 1.000000
Cumulative 3-gram: 1.000000
Cumulative 4-gram: 1.000000
1.0
```

but some also had less than 1.0 BLEU Score

```
English:  i have been writing letters all morning
Actual:  मैं पूरी सुबह से चिट्ठियाँ लिख रहा हूँ।
Predicted:  मैं पूरी सुबह से चिट्ठियाँ लिख रही हूँ। हूँ।
Cumulative 1-gram: 0.777778
Cumulative 2-gram: 0.697217
Cumulative 3-gram: 0.655270
Cumulative 4-gram: 0.610474
0.6104735835807844
```

5. Conclusion & Future Work

In this work, different approaches of summarization and English to Hindi Translation. We got satisfactory results in both the subparts of the project but there is always scope of improvement as we saw that there were some instances where the results were not perfect.

Most importantly, we demonstrated that a simple, straightforward and a relatively unoptimized approach for English Text Summarization and Translation to Hindi Language, so further work will likely lead to even better translation results. These results suggest that an optimized version of our approach will likely do well in real world applications.

References

[1] S. Babar and P. D. Patil, "Improving performance of text summarization," 2014.

[2] O. Tas and F. Kiyani, "A survey automatic text summarization," 2017.

[3] V. Gupta and G. Lehal, "A survey of text summarization extractive techniques," vol. 2, no. 3, 2010.

[4] H. Christian, M. P. Agus, and D. Suhartono3, "Single document automatic text summarization using term frequency-inverse document frequency (tf-idf)."

[5] J. Kupiec, J. Pedersen, and F.Chen, "A trainable document summarizer," pp. 68–73, 1995.

[6] J. Conroy and D. O'leary, "Text summarization via hidden markov models and pivoted qr matrix decomposition," 2001.

[7] R. Witte and S. Bergler, "Fuzzy coreference resolution for summarization," 2003.

[8] L. Zadeh, "Fuzzy sets. information control," vol. 8, p. 338–353, 1965.

[9] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," 2019.

[10] D. R. Balasundaram and P.Krishnaveni, "Automatic text summarization by local scoring and ranking for improving coherence," 2017.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016.

[13] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.