Sentence-Level Extractive Text Summarisation on WikiHow Data

NTU-AI6127: Project Final Report (Group 1)

Cheng Wen Xin, Kelly
kell0034@e.ntu.edu.sg

Goh Zhi Yi, Glendon
ggoh022@e.ntu.edu.sg

Leong Kah Yuan, Andrew
aleong015@e.ntu.edu.sg

Sapuan, Abdul Rasyid
s210025@e.ntu.edu.sg

Zhang Ziyi
ziyi004@e.ntu.edu.sg

Abstract

Text Summarisation is the process of summarising a document into a short and concise version while preserving contextual information. It consists of two main approaches: Extractive and Abstractive. This projects aims to perform Extractive Text Summarisation at the sentence-level on WikiHow articles for quicker consumption while keeping vital information.

We formulate this Text Summarisation task into a multi-label classification problem at the sentence-level. Summary sentences which appear in the reference summary are labelled as 1 while the rest are labelled as 0. Two pre-trained sentence transformer models from Huggingface: "all-MiniLM-L6-v2"[1] and "all-distilroberta-v1"[2], are used to tokenize and generate the sentence embeddings for each paragraph to capture semantic information.

Our proposed model - the **Pre-Trained eXtractive Summarizer (PTX-Sum)** - takes the sentence embeddings as input and output the probability of each sentence token being in the summary. The top 4 highest probability tokens and its associated sentence are selected to form the final summary output. TextRank is used as a baseline model for comparison with our proposed model.

All models are evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) which measures the quality of the system summary against a reference summary.

Our experiments suggest that pairing PTX-Sum with pre-trained models trained over multiple datasets provide higher ROUGE scores than most of the text summarising models that we tested against.

1 Introduction

Text Summarisation is the process of summarising a document of sentences while preserving contextual information. It consists of two main approaches: 1) Extractive - model selects a subset of words or sentences from the document to summarise it. 2) Abstractive - model interprets information from the document and generate its own sentences to summarise it.

In the vast ocean of knowledge on the internet, text summaries are important as they allow readers to comprehend the gist of the article before deciding to spend more time reading the full article. This project is motivated from this and aims to perform Extractive Text Summarisation at the sentence-level on WikiHow articles for quicker consumption while keeping vital information. We will conduct an investigation into how well existing Text Summarization methods perform on WikiHow articles and attempt to improve them by augmenting the said models. It would be interesting to be able to summarise the multi-steps how to articles into a few key sentences. As the WikiHow articles are written and edited by different authors, the writing styles varies which makes this dataset challenging.

Existing works on Extractive Summarisation includes MatchSum, MiniLM, Roberta. These models are encoder or transformer or TextRank.

We approach this Text Summarisation task by formulating it into a multi-label classification problem at the sentence-level. Summary sentences which appear in the reference summary are labelled as 1 while the rest are labelled as 0. Pre-trained sentence transformer models from Huggingface - "all-MiniLM-L6-v2"[1] and "all-distilroberta-v1"[2] - were used to tokenize sentences and generate their semantic embeddings. These sentence embeddings are input into our transformer encoder which has a final classification layer. Sigmoid activation function is applied to the output to get the probability of each sentence token being in the summary. The top 4 highest probability tokens and its associated sentence are selected to form the final summary output.

TextRank model is used as a baseline for comparison against our proposed model.

The models are evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE)[3] which measures the quality of the system summary against a reference summary. ROUGE-1, ROUGE-2 and ROUGE-L score is used.

The outline of this paper is as follows: Section II discusses the related works. Section III describes our proposed approach to Extractive Text Summarisation. Section IV presents our experimental results. Section V evaluates our results against other approaches. Section VI draws conclusion and future directives.

2 Related Works

2.1 WikiHow Dataset

In 2018, the WikiHow dataset was completed and hosted as a public dataset as a solution to the lack of publicly-available large-scale quality text summarization datasets [4]. Prior to this development, the only public datasets were DUC, Gigaword, New York Times, CNN/Daily Mail and Newsroom. These datasets were constructed from professionally-written articles and as such are more structured content-wise than informal text such as those from blogs and product reviews. This formed the base motivation for developing the WikiHow dataset as its source is an online compendium of articles written by mostly ordinary (non-professional) people.

To construct the WikiHow dataset, 142,783 unique articles were extracted from the WikiHow website using a Scrapy bot. A written summary prefaces each WikiHow article, which usually describes a multi-step task in multiple paragraphs. Key sentences from the paragraphs were manually matched against these summaries and labelled 1. Other sentences were labelled 0. From the effort above, a total of 230,843 articles and respective summary pairs were generated.

2.2 MatchSum

The paper "Extractive Summarization as Text Matching"[5] proposes the MatchSum model that attributes its high performance to semantic matching-based summarization. Instead of directly modeling the relationships between sentences using encoder-decoder frameworks, the MatchSum model employs a modified Siamese-BERT model to learn semantically meaningful embeddings of both dataset document and extracted summaries.

The idea is that the gold summary should be semantically similar to - ie. matches - the document. Hence, when scoring the generated summaries using the ROUGE[3] metric, the summary with the highest score should also match the gold summary. This semantic modelling approach proved more successful than sentence-level modelling.

2.3 HIBERT

The paper "HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization"[6] proposes the HIBERT model that replaces the previously used RNN with transformer and use it as the basic building block in the hierarchical document encoder. Since summarizing a document would require two different levels of vector representation, HIBERT uses two transformers as its encoder: the first one to encode the word-level tokens, then the second one to encode the sentences.

As for the pre-training, unlike most common methods which pre-train the model by predicting masked missing words in a sentence, HIBERT predicts the masked missing sentence in a document since summarization is a document-level task. As a result, one more transformer will act as decoder to perform denoising task and predict the complete document. When pre-training is done, the decoder transformer is removed, and the two encoder transformers are repurposed to do sentence label predictions for generating the summaries.

2.4 MiniLM-L6-v2

The all-MiniLM-L6-v2 model is a sentence transformer based on the BERT architecture and outputs 384-dimensional sentence embeddings. The sentence transformer uses a pretrained nreimers/MiniLM-L6-H384-uncased model[7] and was fine-tuned on a dataset that contains 1B sentence pairs using a contrastive learning objective, where given a sentence pair, the model should predict which sentence was actually paired with it. The data used for fine tuning came from a variety of datasets such as Reddit comments (2015-2018), SQuAD2.0, Wikihow and many more.

2.5 DistilRoberta-v1

The all-distilroberta-v1 is another sentence transformer that is based on distilroberta-base[8] model which is a distilled version of the roberta-base transformer and outputs 768-dimensional sentence embeddings. This model was also fined tuned on a similar, 1B sentence pair dataset that was used by all-MiniLM-L6-v2 and also using the same contrastive learning objective.

3 Approach

Extractive text summarization is a multi-label classification problem. Each sample in the WikiHow dataset is a sequence of sentences forming a complete article. Accompanying the sample are truth labels in the form of multi-hot binary vectors, where each element in the vector is either 0 or 1, where 1 means the associated sentence appears in the real summary.

3.1 Baseline

TextRank [9] can summarise text at the sentence-level and it is used as our baseline model. It ranks sentences by how representative they are of the document.

Word embeddings are created for each sentences, and from those word embeddings, word vectors and sentence vectors are retrieved from each sample sentence. The similarity between each sentence are calculated and stored in a similarity matrix of cosine similarities using:

$$\text{Similarity}\,(S_i, S_j) = \frac{S_i \cdot S_j}{||S_i||||S_j||} \tag{1}$$

Where S_i and S_j represents two sentences vectors. Each sentence vector is represented by arrays of word vectors created by:

$$WordVec(i) = \begin{cases} \frac{\sum(W)}{N+\alpha}, & \text{if } N \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Where W represents the array of all word embeddings of a particular sentence i, N representing the length of the sentence, and α representing a small constant variable.

Afterwards, the similarity matrix is converted into a weighted graph where each vertex represents a sentence and the edge between two vertices represents the similarity between the corresponding sentences. The PageRank algorithm [10] is applied to the graph:

$$WS(V_i) = (1-d) + d * \sum_{V_j \in \mathrm{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in \mathrm{Out}_{(V_j)}}^{w_{jk}}} WS(V_j) \tag{3}$$

Where V_i and V_j represents two vertices and w_{ij} represents the weight between them. d is a damping factor ranged [0, 1].

The top 4 sentences with the highest PageRank score are selected and appended into a summary of the article, separated by commas. However, if the number of sentences within an article is less than 4, no summarization is done, and instead, all sentences are chosen as the summary of the article.

3.2 Proposed Approach

We trained a model with output dimensions the same length as the input sequence and put it through a final sigmoid function to get the probability of each token.

We also decided to use 2 different pre-trained sentence transformer models from huggingface, "all-MiniLM-L6-v2"[1] and "all-distilroberta-v1"[2] to tokenize and get the sentence embeddings for each paragraph. The MiniLM-L6 is based on the larger MiniLM-L12-H384-uncased[11] distilled BERT based model, while all-distilroberta-v1 is a distilled version of the RoBERTa-base model[12]. Both pre-trained models has been modified by Hugging Face to be sentence transformers, this allows it output sentence embeddings that captures the semantic information of the paragraph.

Our model - the **Pre-Trained eXtractive Summarizer (PTX-Sum)** - uses these sentence embeddings as input to a transformer encoder in which its output is passed through a final classification layer to get the probability of each sentence token being in the summary. We then take the top 4 highest probability tokens and its associated sentence to form the final summary output.

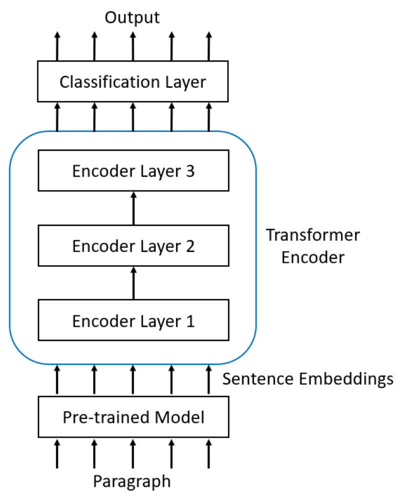

Figure 1: Model Architecture

4 Experiments

4.1 Data

In this project we use a dataset based on the WikiHow dataset [13]. The WikiHow dataset contains over 200,000 articles and has 2 different types:

1. Describe single-method tasks in different steps
2. Describe multiple steps of different methods for a task.

Each article consists of multiple paragraphs and each paragraph starts with a sentence summary.

Since for this project we are performing Extractive Summarization, the modified dataset we used is taken from TransformerSum website [14]. They have modified the original WikiHow dataset to be suitable for Extractive Summarization.

The Extractive WikiHow dataset contains the same articles but it also includes a list of binary classes that indicates which sentences from the original paragraph is used as the summary.

Some further pre-processing was done on the dataset as the one from TransformerSum has a sentence as a list of tokens and a paragraph containing a list of sentences. For certain approaches, the dataset was pre-processed to combine each token into a string to represent the sentence and each paragraph being a list of these pre-processed strings.

The dataset also came in multiple files split into 93.5% training, 3.5% validation and 3.5% test sets. We have combined these into a single json file for easier manipulation.

4.2 Evaluation method

The performance of PTX-Sum was evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [3] which measures the quality of the system (computer-generated) summary against a reference summary (ideal summary created by humans). The score is derived by counting the number of overlapping tokens between the two pieces of text.

Metrics ROUGE-1, ROUGE-2 and ROUGE-L will be used for evaluation. A higher ROUGE score indicates stronger similarity between candidate and references.

1. ROUGE-N: Overlap of n-gram between the system and reference summaries.

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{ ReferenceSummaries }} \sum_{gram_n \in S} \left(\text{count}_{\text{match }} \left(\text{gram}_n\right)\right)}{\sum_{S \in \text{ ReferenceSummaries }} \sum_{\text{gram }_n \in S} \left(\text{count } \left(\text{gram}_n\right)\right)} \tag{4}$$

   where n stands for the length of the n-gram and $\text{count}_{\text{match}} \left(\text{gram}_n\right)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

   (a) ROUGE-1: Overlap of unigram (each word) between the system and reference summaries.

   (b) ROUGE-2: Overlap of bigrams between the system and reference summaries.

2. ROUGE-L: Measure longest matching sequence of words using Longest Common Subsequence (LCS) between two sentences. LCS-based F-measure is used to estimate the similarity between reference summary sentence X of length m and candidate summary sentence Y of length n.

$$R_{lcs} = \frac{LCS(X, Y)}{m} \tag{5}$$

$$P_{lcs} = \frac{LCS(X, Y)}{n} \tag{6}$$

$$F_{lcs} = \frac{\left(1+\beta^2\right) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{7}$$

$$\text{ROUGE-L} = \frac{\sum_{s_i \in S_1} \max s_{s_j \in S_2} LCS\left(s_i, s_j\right) + \sum_{s_j \in S_2} \max_{s_i \in S_1} LCS\left(s_i, s_j\right)}{\sum_{s_i \in S_1} \text{length}\left(s_i\right) + \sum_{s_j \in S_2} \text{length}\left(s_j\right)} \tag{8}$$

   Where $LCS(X, Y)$ is the length of a longest common subsequence of X and Y. ROUGE-L does not require consecutive matches but in-sequence matches that reflect sentence level word order as n-grams. It does not require predefined n-gram length as it automatically includes longest in-sequence common n-grams.

4.3 Experimental details

In this experiment, we used 2 different pre-trained sentence transformer models and kept its settings at its default configuration, while the transformer encoder we used has the following parameters:

- Pre-trained Model: all-MiniLM-L6-v2 / all-distilroberta-v1
- Input Dimension: 384 / 768
- Hidden Layer Dimension: 256
- Number of Encoder Layers: 3
- Number of Attention Heads: 4

Due to the size of the WikiHow dataset, we have trimmed it to only contain paragraphs with a maximum of 80 sentences. This still contains 150,000 training pairs which is 90% of the original dataset. We then trained the model for 10 epochs, in each epoch, a random sample of 20,000 training pair samples are selected from the trimmed dataset.

In training, we used the Adam optimizer with a learning rate of 1e-4, together with the Binary Cross Entropy loss function. The training time, depending on the pre-trained model used, takes approximately 3 to 4 hour. The PTX-Sum model using MiniLM-L6 contains 10,255,568 parameters while the PTX-Sum model using all-distilroberta-v1 contains 24,028,496 parameters.

4.4 Results

The table below contains the Rouge scores PTX-Sum model permutations achieved compared to other models[15].

Model	Rouge-1	Rouge-2	Rouge-L
Baseline TextRank	22.36	4.57	21.25
PTX-Sum using all-MiniLM-L6-v2	38.60	21.90	36.11
PTX-Sum using all-distilroberta-v1	37.60	21.01	35.17
distilbert-base-uncased-ext-sum	30.69	8.65	19.13
distilroberta-base-ext-sum	31.07	8.96	19.34
bert-base-uncased-ext-sum	30.68	8.67	19.16
roberta-base-ext-sum	31.26	9.09	19.47
MatchSum	31.74	8.96	29.48
HIBERT	36.18	20.71	33.43

Based on the results above, the PTX-Sum scores were higher than other models' across all Rouge scores. Our guess is that the pre-trained models were able to capture the contextual information in the paragraph because it was trained on a large datasets across a variety of corpuses. This provided a good base sentence embedding for us to use as input to our transformer encoder model.

5 Analysis

Currently PTX-Sum works by getting sentence embeddings from pre-trained sentence transformers as an input. It can exploit the available pre-trained sentence transformers that are trained on various large and diverse datasets which capture contextual information well. For the context of WikiHow articles, our proposed model performed better than the baseline model and other state-of-the-art models. PTX-Sum using all-MiniLM-L6-v2 performed the best followed by PTX-Sum using all-distilroberta-v1.

However, as each pre-trained model has its own limitations based on the sequence-length (number of sentences) of the paragraph, certain documents that contain very long paragraphs may face issues in the encoding of the sentences if they cannot be trimmed down.

PTX-Sum is also reliant on pre-trained models to generate the sentence embeddings. These pre-trained models are trained on very large datasets. In the scenario where there are no suitable pre-trained models to use, in which case we will have poor sentence embeddings which will then result in poor performance or we would have to train a sentence transformer from scratch which will be less efficient and more time consuming.

6 Conclusion

In conclusion, our proposed approach can exploit the available pre-trained sentence transformer that are trained on large and diverse datasets which capture contextual information well. For the context of WikiHow articles, our proposed model performed better than the baseline model and other state-of-the-art models. PTX-Sum using all-MiniLM-L6-v2 performed the best followed by PTX-Sum using all-distilroberta-v1. However, in scenarios where no suitable pre-trained sentence transformers is available, our model performance will suffer. This is because we will have poorer sentence embedding and need to train a sentence transformer from scratch.

References

[1] Hugging Face. *Sentence Transformer: all-MiniLM-L6-v2*. URL: `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2#all-minilm-l6-v2`.

[2] Hugging Face. *Sentence Transformer: all-distilroberta-v1*. URL: `https://huggingface.co/sentence-transformers/all-distilroberta-v1`.

[3] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: `https://aclanthology.org/W04-1013`.

[4] Mahnaz Koupaee and William Yang Wang. "WikiHow: A Large Scale Text Summarization Dataset". In: *CoRR* abs/1810.09305 (2018). arXiv: `1810.09305`. URL: `http://arxiv.org/abs/1810.09305`.

[5] Ming Zhong et al. "Extractive Summarization as Text Matching". In: *CoRR* abs/2004.08795 (2020). arXiv: `2004.08795`. URL: `https://arxiv.org/abs/2004.08795`.

[6] Xingxing Zhang, Furu Wei, and Ming Zhou. "HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5059–5069. DOI: `10.18653/v1/P19-1499`. URL: `https://aclanthology.org/P19-1499`.

[7] Hugging Face. *nreimers/MiniLM-L6-H384-uncased*. URL: `https://huggingface.co/nreimers/MiniLM-L6-H384-uncased`.

[8] Hugging Face. *distilroberta-base*. URL: `https://huggingface.co/distilroberta-base`.

[9] R. Mihalcea and P. Tarau. "TextRank: Bringing Order into Texts". In: *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelonaand Spain, July 2004.

[10] Sergey Brin and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Computer Networks* 30 (1998), pp. 107–117. URL: `http://www-db.stanford.edu/~backrub/google.html`.

[11] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: `2002.10957 [cs.CL]`.

[12] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (2019).

[13] mahnazkoupaee. *WikiHow-Dataset*. URL: `https://github.com/mahnazkoupaee/WikiHow-Dataset`.

[14] Hayden Housen. *Extractive WikiHow-Dataset*. URL: `https://transformersum.readthedocs.io/en/latest/extractive/datasets.html`.

[15] Hayden Housen. *Rouge Score Baselines*. URL: `https://transformersum.readthedocs.io/en/latest/extractive/models-results.html#wikihow-rouge-scores`.

A Member Contribution

The members contributed to the corresponding areas:

1. Abdul - Report
2. Andrew - Data processing and PTX-SUM

3. Glendon - TextRank model
4. Kelly - Report
5. Ziyi - MatchSUM and HIBERT model

B Source codes for the recent papers' models

The source codes and implementation instructions for the MatchSum and HIBERT can be found in the following links:

1. MatchSum: https://github.com/maszhongming/MatchSum
2. HIBERT: https://xingxingzhang.github.io/hibert.html