Towards a Knowledge-Aware AI
A. Dimou et al. (Eds.)

doi:10.3233/SSW220019

Metadata Based Contextual Summarizer for Technical Conversations in Public Forums

Gyan Ranjan [a], Abinaya Govindan [b] and Amit Verma [c]
[a] *Neuron7.ai, Bangalore, India*
[b] *Neuron7.ai, Bangalore, India*
[c] *Neuron7.ai, California*

Abstract. In recent years, the task of sequence to sequence based neural abstractive summarization has gained a lot of attention. Many novel strategies have been used to improve the saliency, human readability, and consistency of these models, resulting in high-quality summaries. However, because the majority of these pretrained models were trained on news datasets, they contain an inherent bias. One such bias is that most of these generated summaries originate from the start or end of the text, much like a news story might be summarised. Another issue we encountered while using these summarizers in our Technical discussion forums usecase was token recurrence, which resulted in lower ROUGE-precision scores. To overcome these issues, we present a unique approach that includes: a) An additional parameter to the loss function based on ROUGE-precision score that is optimised alongside categorical cross entropy loss. b) An adaptive loss function based on token repetition rate which is optimized along with the final loss so that the model may provide contextual summaries with less token repetition and successfully learn with the least training samples. c) To effectively contextualize this summarizer for technical forum discussion platforms, we added extra metadata indicator tokens to aid the model in learning latent features and dependencies in text segments with relevant metadata information. To avoid overfitting due to data scarcity, we test and verify all models on a hold-out dataset that was not part of the training or validation dataset. This paper discusses the various strategies we used and compares the performance of fine tuned models against baseline summarizers n the test dataset. By end-to-end training our models with these losses, we acquire substantially better ROUGE scores while being the most legible and relevant summary on the Technical forum dataset.

Keywords.
natural language processing, abstractive summarization, sequence-to-sequence models, multiple loss optimization, rouge-based learning, information systems

1. Introduction

With the massive increase in available information due to the growth of the Internet, meaningful data consumption has always been a difficult endeavour. The tremendous growth in the quantity of blogs, articles, research papers, and reports is particularly combustible. Due to the abundance of data, extensive study into various summarising techniques - both abstractive and extractive - has been conducted. According to Radef et al. [3], a summary is ”a text that is constructed from one or more texts, contains crucial information in the original text(s), and is no longer than half of the original text(s) and frequently, substantially less than that.” Having such contextual brief summaries can help people grasp material better, which can lead to more efficient text processing and comprehension. A summary might be extractive or abstractive in nature. The summary is simply a reduced version of the text by picking important sentences, segments, and paragraphs from the original text block. Abstractive summarising techniques rely on the semantic knowledge of the text to provide semantically coherent, factually and logically valid summaries.

At Neuron7.ai we’re building a Service Intelligence platform that, when presented with information about defective hardware, suggests actions and offers actionable insights to the end user. It is critical for us to be able to deliver as many relevant insights as possible during this process of enabling clients to draw meaningful insights from device malfunction logs and repair notes. To accomplish so, we employ technical data sources found in every organization’s ecosystem, such as product manuals, knowledge articles, and internal technical forums. Our application allows users to interact with the system and ask questions about any technical problem they are having, and it then recommends appropriate course correction procedures based on the knowledge gained from the technical documents. The user query often has multiple dimensions, including information on device types, recurrence history, efforts done to resolve the issue to date, and any additional parameters that they believe might help us deliver a better solution. When these parameters are treated as a whole, they can cause highly particular issues, which can add to the model’s confusion rather than help it better its outcomes. As a result, we require a sophisticated and domain-specific summariser capable of making sense of such technical data and condensing it into a format that can be consumed by downstream natural language processes.

We begin by establishing a baseline performance using current state-of-the-art summarizers. From the experiments using these out-of-the-box models, we faced the following issues -

- The input query’s technical terminology and words frequently lacked grammatical structure and were frequently found in the generated summaries in inappropriate circumstances.
- We noticed a lot of token duplication and repetition in the generated summary, which is a frequent problem with many natural language generation tasks.
- Because of the structure of the dataset on which these models were trained, the majority of the produced summaries came from the beginning or conclusion of the text. This presumption is false in our technical discussion forums, as participants frequently express their problems in no particular sequence.

To solve the above issues, we designed and developed a system which

- **Pre-processing stage** - Because the summarizer’s main purpose is to feed our Technical forum Search with summarised user queries (Heterogeneous QA framework in Govindan et al. [2]), the first stage of the pre-processor included custom modules to clean technical phrases that don’t add value to the context of the question.

- **Hierarchical metadata extractor** - The metadata extractor, which is used to extract hierarchical metadata using an unsupervised named entity recognizer as proposed by Govindan et al, [20], is the second portion of the pre-processor.
- **Modeling stage** - To help the framework perform well in the Technical forum domain, we included two new loss parameters: the ROUGE precision-based loss function and the token repetition rate-based loss function, which are learnt in addition to the category cross entropy loss to provide short and targeted summaries.
- **Post-processing stage** - Because we utilise the produced summaries in the Technical Search forum ([2]), we execute a few post-processing processes to assist us give the best search results and suggestions to users.

This paper investigates and compares the proposed approach with various summarising approaches and evaluates performance criteria such as ROUGE-precision, recall, and F1 to assess the quality of the generated summaries. We also use other metrics to assess the length and quality of summaries, such as token average lengths and lengths at different percentiles, to choose the optimum model.

2. Related work

Many methods for extracting summaries from text have been proposed, according to Moratanch et al. [4], using a range of weighting mechanisms such as content, word, and sentence level features in various graph based, fuzzy logic, and Latent semantic analysis based techniques. Recent transformer-based advancements, like as BERT, have led to strategies like leveraging BERT embeddings in combination with clustering algorithms to create extractive summaries, as proposed by Biswas et al. [14].

Sequence to sequence based models have been successfully deployed in applications including neural machine translation, headline generation, and text summarization in recent years. The majority of these models may provide abstractive summaries using just appropriate neural network models like recurrent neural networks RNNS, long short term memory LSTMs (Sepp et al. [6]), or gated recurrent units GRUs. These models are based on the encoder-decoder architecture (Sutskever et al. [10]), in which the encoder accepts an input sequence of words and the decoder emits an output sequence of tokens that make up the resulting summary. These networks are often trained on a large corpus of training text-summary pairs, learning to generate summaries of incoming text as a result.

Another state-of-the-art model is a hybrid pointer-generator network that combines pointing and coverage metrics to replicate words from the source text. These enhancements result in accurate information replication while the generator continues to generate fresh words. Due to the coverage metric [9], these summaries had a lower word repetition rate. Other techniques, such as [7], employ attentional transformers to construct summaries that capture sentence-to-word hierarchy to achieve best-in-class performance in abstractive text summarization.

Basic seq2seq models may be enhanced further by incorporating an attention mechanism (Bahdanau et al. [1]), resulting in attention-based encoder decoder models being a standard architecture for all sequence to sequence tasks, particularly text summarization (Lin et al. [11]). The attention mechanism seeks to ”attend” to the most significant words in the sequence, resulting in the most meaningful summaries based on the order of most

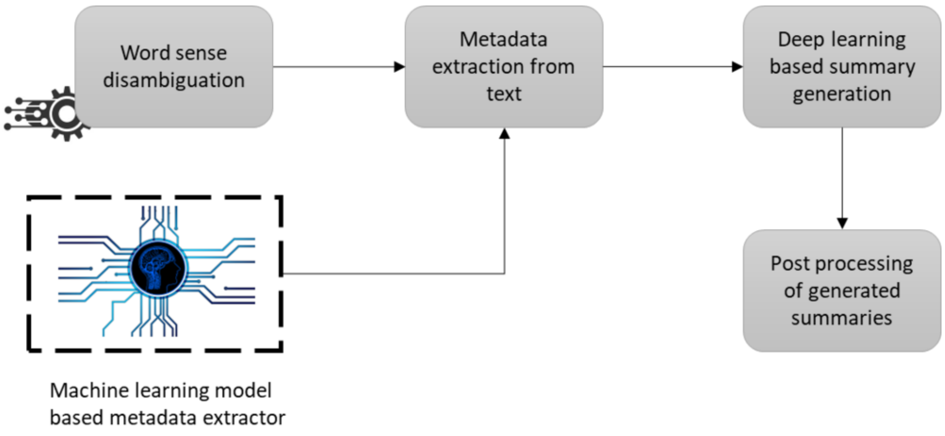

Word sense disambiguation
Metadata extraction from text
Deep learning based summary generation
Post processing of generated summaries
Machine learning model based metadata extractor

Figure 1. Detailed overview of proposed workflow

important words in the text. This is accomplished by learning and feeding a context vector that quantifies the value of each token in the sequence to the decoder when constructing the summaries. You et al. [13] presented a transformer-based encoder-decoder architecture with an encoder-integrated focus-attention mechanism and a separate saliency-selection network that regulates the information flow from encoder to decoder. To predict generalised and resilient summaries, Kouris et al. [8] uses a coping and coverage strategy in encoder-decoder based models, as well as reinforcement learning.

As previously said, the bulk of these models are trained using publicly available datasets such as news datasets, which have a clear grammatical framework and a consistent structure across several news sources. These models, on the other hand, ignore the technical character of inquiries for technical forums, as well as the repetitive structure of the resulting summary. As a result, this research focuses on using current methodologies for the technical forum domain by integrating technical metadata during the data preparation stage and loss functions that can tackle token repetition difficulties during the modelling stage. The suggested framework's core component is a sequence-to-sequence deep learning neural network, which is helped by a metadata-based technique, as detailed below.

3. Our Work

The overall framework is illustrated in 1. The input to the framework consists of single document text, along with a few metadata information, while the output is a human-readable summary. The various sections that comprised the framework are :

- **Pre-processor** - The pre-processor module performs a few custom steps to handle the technical nature of the dataset - in terms of cleaning unwanted tokens to improve quality of input text
- **Hierarchical metadata extractor** - The hierarchical metadata extractor has been trained to extract metadata information at two levels and add it to metadata input

text in order enable the model to learn metadata-based latent features in the data more effectively.

- **Modeler** - We included an extra loss function based on the token repetition rate, which was optimized in addition to the category cross entropy loss, to increase the quality of the output summaries. The model was able to create summaries with a lower token repetition rate attributed to this function. We also introduced an extra loss function derived from the Rouge-precision score was introduced and optimised. Since this loss is predicated on precision, extraneous tokens that appear in summaries along with the correct tokens, are penalised, allowing the model to create more focused summaries.
- **Postprocesser** - Since we use the generated summary as a user query in downstream Technical Forum Search [2], we execute a few post-processing processes such as reinforcing segment which has metadata, adding PII information to guarantee that the created summaries have all the required data to deliver the best query results and suggestions to the end user.

3.1. Data Pre-processor

We created a comprehensive pre-processor with numerous modules, as below:

3.1.1. Technical data cleaner

The data source for our usecase came from Slack-like technical forums, necessitating the deployment of the following bespoke data cleansing modules:

- To extract and eliminate URLs that led to resolutions attempted by the user or other associated information, we employed regular expression-based processors.
- Image processors that detect and remove screenshots of user errors from the query text
- Using the Python library spacy [12], we used out-of-the-box named entity recognizers to extract named entity types like PERSON. Because they don't provide any useful information to the user inquiry, these named entities were eliminated from the input text.
- Motivated by the work of Kouris et al. [8], we use a word sense disambiguator to provide additional meaning to ambiguous statements. For words that are too specialised to the issue, the disambiguator provides further information to the user question. *SLA*, *VDI* and *VNC* are examples of acronyms used in the technical sector that have no relevance to the model until more information is provided. We extract acronyms using word parsers and thoroughly evaluate the most essential ones with business stakeholders. We established a word taxonomy and dictionaries, which are then fed into the preprocessor to provide extra context for the words.

3.2. Hierarchical metadata extractor

The technical nature of the data is the key reason why typical summarizers fail to perform adequately in our scenario. We intended to create a system that could recognise metadata related information in the data to assist the model in making sense of such noisy data. One common observation in this data is The intent is that when a section of text, such as a phrase, has information related to metadata associated with it, that sentence frequently includes useful information that should be kept in the summary. For example, if a user has a problem with their *VDI*, the query might include information such as the time when it stopped operating, the methods taken to resolve the problem, and so on.However, if one were to find the most important pieces of this query, the sentence that includes *VDI* relevant information should be the first place to search. We picked a few entity types that would make the most sense for the domain we want to target. However, technical forum data frequently contained two levels of metadata - for example, a a *mobile* can be the first level, and additional particular information such as *android* or *ios*, if available in the text, can assist provide more depth to the suggested resolution. As a result, the named entity recognizer needed to save hierarchical information and be able to deliver metadata at both levels.

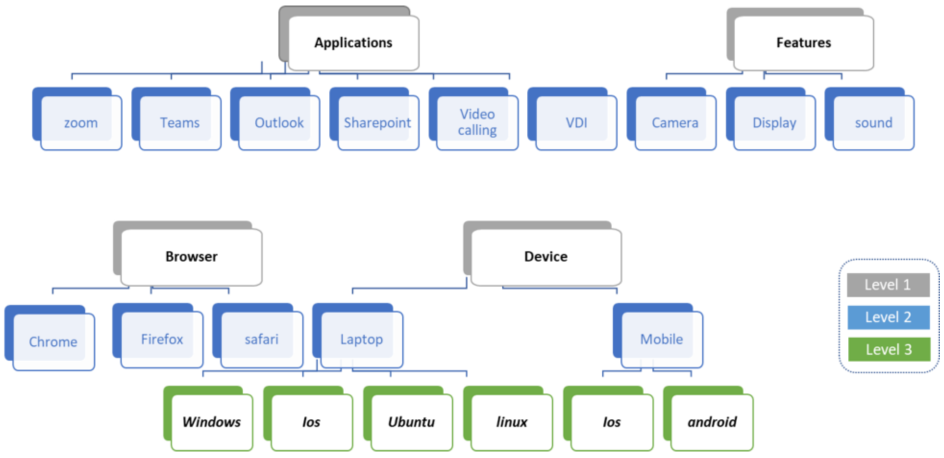


Figure 2. Sample hierarchical structure of metadata extracted

3.2.1. *Heuristic metadata extractor*

As seen in 3, the final metadata extractor is a combination of heuristic models and semi-supervised named entity extractors. Heuristic engines can handle metadata kinds that are rather straightforward. These metadata categories included things like *browser* and others that had extremely restricted values and all of these exhaustive values were already pre-defined and made accessible, either as a result of domain intelligence obtained before or as a result of business definition.

3.2.2. *Semi-supervised named entity extractor*

Semi-supervised named entity extractors were used to extract metadata types with more complex values and various variants and values. We intended to use a semi-supervised data formulation method to annotate and train the entity recognizers, as proposed in Govindan et al, [20]. We employed a two-level classification to manage the data hierarchy: one model predicted the Level-1 of metadata type, and this entity label, along with the text, was used to train the second level of entity recognizer. In 4, the whole NER training mechanism is described. There are two main modules in the system.

- The Level 1 entity recognizer is the initial component of the trainer, and it uses transformer-based named entity recognizers to identify the first level kind of information. We won't go into depth about this model because it's already been covered in Govindan et al, [20]
- The second component of the trainer consists of a Level 2 named entity recognizer that was trained using the Level 1 labels, as well as text that is used to predict the Level 2 entity kinds. For each of the level 1 types, additional tokens that are not part of the current vocabulary are defined, and these tokens are appended before the relevant entity token in the text and supplied as input to the model.
- Negative examples containing only Level 1 types, such as *My mobile does not work*, are also used to train the model so that it can recognise when the second level of metadata is missing in the text.

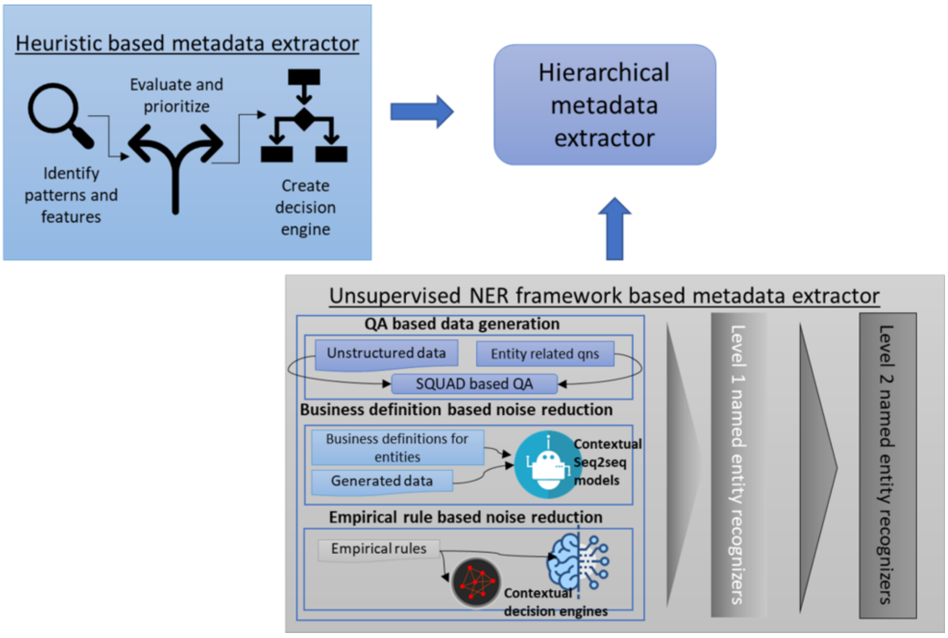

Heuristic based metadata extractor
Evaluate and prioritize
Identify patterns and features
Create decision engine
Hierarchical metadata extractor
Unsupervised NER framework based metadata extractor
QA based data generation
Unstructured data
Entity related qns
SQUAD based QA
Business definition based noise reduction
Business definitions for entities
Generated data
Contextual Seq2seq models
Empirical rule based noise reduction
Empirical rules
Contextual decision engines
Level 1 named entity recognizers
Level 2 named entity recognizers

Figure 3. Hierarchical metadata extractor

We identify the highest level of entity type contained in each phrase once our models have been fine-tuned to recognise Level 1 and 2 entity types. We return Level 1 type if a phrase contains Level 1 entity type. Level 2 entity type is returned if the text has granular Level 2 type. This entity type indicator token was previously defined and added to the transformer's tokenizer's vocabulary before being inserted before each phrase in the text. As mentioned below, this content is used as input text, and the summary is used as a target for the decoder to learn.

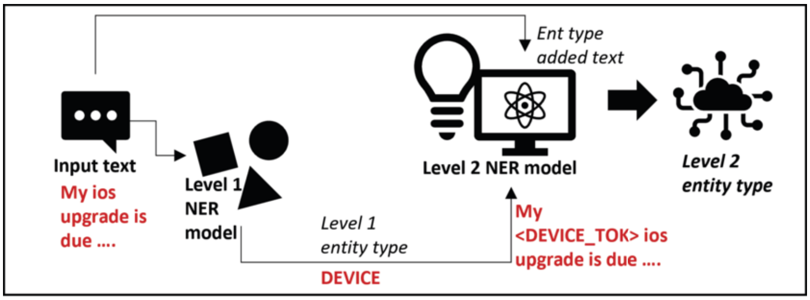


Figure 4. Hierarchical metadata extractor

3.3. Modeler

A deep sequence to sequence model based on Transformer encoders and decoders is used to generate the summary. This model, in more formal terms, predicts the output sequence of summary tokens as defined as :

$$Y` = (y_1`, y_2`, y_3`, .., y_m`)$$

given an input sequence of tokens

$$X = (x_1, x_2, ..x_n)$$

We have built our proposed approach on Transformer based architecture such as DistilBert and T5 since transformer-based models (proposed by Vaswani et al. [17] , Zhang et al. [18] and Song et al. [19]) have been highly popular for numerous natural language processing problems. Refer to 5 to see the suggested model architecture. The modeller goes through several stages:

3.3.1. Input layer

We fragment the input into phrases or logical blocks to feed it to the transformer encoder-decoder (specific to use cases). We next determine if each of the segments has metadata-related information, and if so, we keep the metadata name as well as the granularity level (Level 1 or Level 2). We extract the matching token identification from the metadata classes and append the token to the beginning of each segment. Label encoded values make up the metadata classes. The following structure is now supplied as input to the model:

$$\text{Input} : X = (x_1, x_2, x_3 .... x_n)$$

$$\text{Metadata extracted} : M = (m_1, m_{11}, .., m_n)$$

where m_1 signifies the first Level 1 metadata and

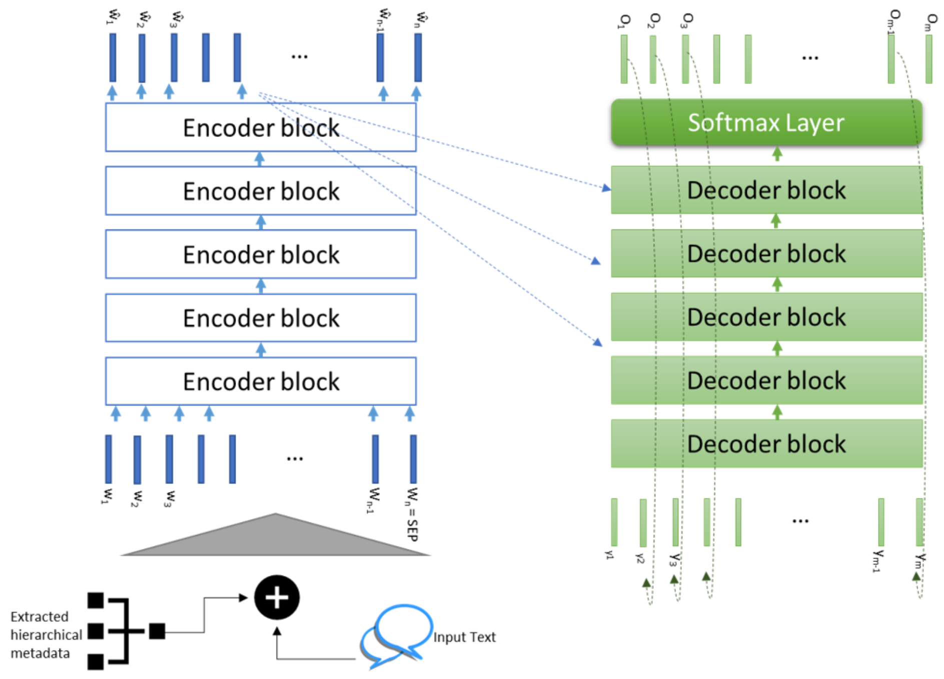


Figure 5. Proposed model architecture

m_{11} corresponds to 1st sub entity in 1st entity class.

$$\text{Final metadata infused input : } X(M) = (m_1 x_1, m_n x_n)$$

3.3.2. Pretrained transformer model

The various variants of the model that we tried are :

DistilBert based trainer. We fine-tuned our summarizer based on DistilBert since we wanted the model to be computationally cheap.

Distilled T5 based trainer. The difficulty of deploying very deep neural network models is especially pertinent for edge devices with limited memory and processing power. To address this issue, Hinton et al. [21] created a model compression strategy to transfer information from a large model into training a smaller model with no substantial performance loss. We used teacher-student training, in which the learner strives to imitate the instructor model and leverage the information to reach similar, if not superior, accuracy. On a t5-base (Raffel et al. ([22])) pretrained model, we attempted knowledge distillation, and the distiled t5-base model was then utilised to fine tune our summarizer.

T5-based trainer. The third variant was based on t5-small proposed by Raffel et al. [22]. We fine-tuned the model's weights using the pre-trained weights from t5-small.

The weights of these models were fine-tuned for our dataset for all of the aforementioned model variants.

3.3.3. Loss functions

In the proposed system, we defined the following losses :

Cross entropy loss. One of the loss functions we use is the classic cross entropy loss which is defined as

$$Loss_{CE}(y,\hat{y}) = -\sum_{i=1}^{\mathrm{N}_c} y_i \log(\hat{y}_i) \tag{1}$$

where $y \in \mathbb{R}^{|V|}$ is a one-hot label vector and N_c is the number of classes.

ROUGE based loss. This innovative loss function was developed to decrease the amount of wrong tokens and token repetition in the generated summary, which is a prevalent problem in natural language production. The loss function is created as follows:

$$\begin{aligned} Precision &= TP/(TP+FP) \\ Prec\hat{i}sion &= min(Precision, 100) \\ Loss_{PRE} &= 1/Prec\hat{i}sion \end{aligned} \tag{2}$$

Repetition rate based loss. The repetition rate based loss has been designed to identify and promote the number of unique tokens in the resulting summary. This is calculated using the total number of tokens in the summary and the number of unique tokens in the summary.

$$\begin{aligned} Loss_{Rep} &= n(\hat{A})/n(A) \\ \text{where} \hat{A} &= set(A) \text{ and} \\ \text{A is number of tokens in generated summary} \end{aligned} \tag{3}$$

During the training phase, all of these losses are trained and optimised as individual functions. We didn't aggregate these individual losses to produce an ensembled or single loss since we didn't want any of the loss values to dominate the final loss. Individual training guarantees that each function contributes evenly to the overall loss function.

3.4. Postprocesor of generated summaries

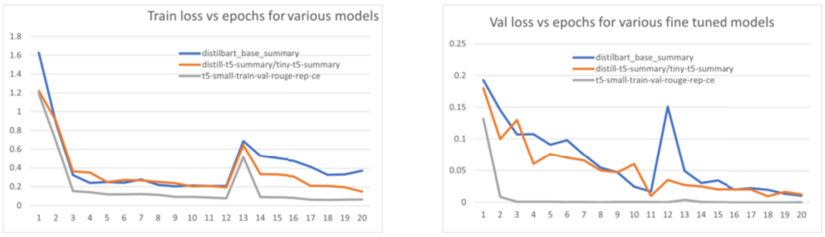
3.4.1. PII extractor

We have observed that Technical forum user queries often include information personal to the user such as device type, username, location, and team names. We extract all of these personal identification identifiers using PII parsers and add these parameters as a JSON which is then passed to the downstream Technical Forum Search platform for better user recommendations.

4. Performance evaluation and Inference

4.1. Parameter settings

We employed 256 and 512 word embeddings and hidden states (of encoder and decoder) in all of our tests since we fine-tuned utilising BERT/t5 based models. The weights of the transformer encoder and decoder were also learnt throughout the training. For stochastic optimization, Adam (Kingma et al. [15]) was employed with the hyperparameters $beta1$= 0.9 and $beta2$=0.999. We utilised a 5e-5 learning rate and a mini batch of 4. Due to the fact that ROUGE-precision loss employs the inverse of ROUGE precision score, very low precision scores result in an extremely high value, hence leading to poor learning by the model. To circumvent this, we set a threshold defined by heuristics for the ROUGE loss. All of the models were trained for a total of 20 epochs. We executed a train-test split of the training dataset, and at the end of each epoch, we evaluated the ROUGE score on the validation set. The input text had a maximum sequence length of 512 characters, and the target summary length was 32, with any longer summaries being truncated. This length was chosen after considering a number of parameters, including the length that allows for the optimal performance in indexed searches and downstream question answering tasks. By comparing the final training and validation losses of multiple fine-tuned models, the influence of ROUGE and token repetition rate based losses is first investigated. In 6, the training and validation loss for models fine-tuned based on distilbert, distiled t5-base, and recommended framework based on t5-small are presented.



(a) Comparison of train loss (b) Comparison of validation loss

Figure 6. Comparison of train and validation loss during fine tuning of various models

4.2. Evaluation metrics

In our test experiments, we validated the baseline models (Wolf et al. [5] huggingface's distilbert-based summarizer) performance on our dataset. To compare the models' performance, the following variables were used:

(A) **Evaluation at runtime** In 7, we use the average run time of summary generation as a first assessment criterion. This is an important statistic for our use case since real-world practical systems must answer in near-real-time, because the user expects a definitive response from the system as soon as he enters his query.

(B) **Succinctness of summary assessment** We evaluate token lengths at various percentiles to determine the model that can provide summaries that are the most comparable to the predicted summaries to assess the quality of summaries in terms of redundant information and additional tokens mentioned.

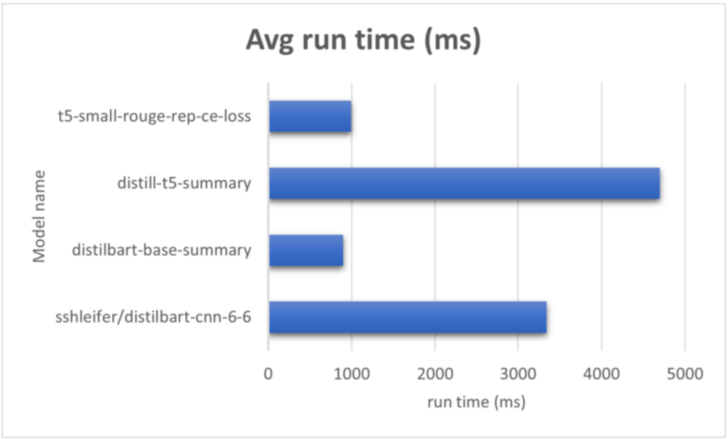
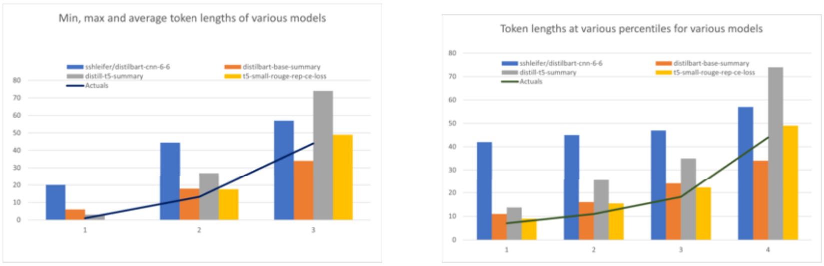


Figure 7. Comparison of runtime for different models



(a) min, max and avg lengths

(b) lengths at percentiles

Figure 8. Comparison of generated summary lengths at various percentiles for different models

(C) **Lexical similarity evaluation** ROUGE scores, or Recall-Oriented Understudy for Gisting Evaluation, were first introduced by Chin et al. [16] and have since become standard metrics for evaluating various text summarization algorithms. To test the quality of summarization, they quantify the number of overlapping units (i.e., n-grams, word sequences, and word pairs) between machine-generated and golden-standard (human-written) summaries. ROUGE-1 (unigram), ROUGE-2 (bigram), and ROUGE-L are the most commonly utilised ROUGE measures for single-document abstractive summarization (longest common subsequence). The pyrouge package, which gives accuracy, recall, and F-score for these metrics, is used to evaluate several models in this article. On our hold-out dataset, we use ROUGE-L scores to evaluate the models' overall lexical performance. As seen in 1, the system proposed surpasses baseline and other finetuned models by a significant margin. We compare our findings to those of other fine-tuned models to demonstrate that fine-tuning any out-of-the-box model may not always produce the best results, especially if training data is scarce.

Model name	**Rouge1**			**Rouge2**			**RougeL**		
	Precision	**recall**	**fmeasure**	**Precision**	**recall**	**fmeasure**	**Precision**	**recall**	**fmeasure**
sshleifer/distilbart-cnn-6-6	0.2573	**0.9125**	0.3714	0.1966	**0.7116**	0.2832	0.2265	**0.8175**	0.3277
distilbert	0.1844	0.2488	0.1975	0.0867	0.1203	0.0953	0.1625	0.2263	0.1766
distil-t5	0.4369	0.7891	0.5211	0.3454	0.6171	0.4086	0.4057	0.7250	0.4808
t5-small-rouge-rep-ce-loss	0.6313	**0.7801**	**0.6742**	0.5388	0.6233	**0.5576**	0.6009	0.7262	**0.6359**

5. Future Work

Our current work focuses on designing a system that provides the most appropriate input format and exploiting token-related scorings to improve the transformer-based model's performance. While our method beats baseline models on test data, we want the summarised query to be constructed as a hybrid of extractive and abstract methods, with the model being able to paraphrase and infer information over a large number of words. This needs the training data to be curated in such a mixed-format. So we would want to apply an active-learning framework, as provided by Govindan et al. [23] in which the proposed model may generate abstractive summaries after sufficient training and the user can edit the summary to create a semi-extractive summary, which can then be used to train hybrid summarizers. We also want to apply loss functions based on decoder-encoder attentions to improve learning of important features while summarizing.

References

[1] Bahdanau, D., Cho, K. and Bengio, Y., 2022. Neural Machine Translation by Jointly Learning to Align and Translate. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1409.0473¿ [Accessed 9 May 2022].

[2] Govindan, A., Ranjan, G. and Verma, A., 2021. Question Answering Module Leveraging Heterogeneous Datasets. [online] http://dx.doi.org/10.5121/ijnlc.2021.10601.

[3] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown, "Introduction to the special issue on summarization," Computational linguistics 2002. 28, 4 (2002), 399–408.

[4] Moratanch, N. and Gopalan, Chitrakala, "A survey on extractive text summarization",2018 1-6.10.1109/ICCCSP.2017.7944061.

[5] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T., Gugger, S., Drame, M., Lhoest, Q. and Rush, A., 2022. HuggingFace's Transformers: State-of-the-art Natural Language Processing. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1910.03771

[6] Sepp Hochreiter and J¨urgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780

[7] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, C¸ a glar Gulc¸ehre, and Bing Xiang. 2016. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. CoNLL 2016 (2016), 280.

[8] Kouris, Panagiotis and Alexandridis, Georgios and Stafylopatis, Andreas, "Abstractive Text Summarization: Enhancing Sequence-to-Sequence Models Using Word Sense Disambiguation and Semantic Content Generalization", Computational Linguistics, 2021.

[9] See, Abigail, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368. https://doi.org/10.18653/v1/P17-1099

[10] Sutskever, I., O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. Advances in NIPS.

[11] Lin, Hui and Vincent Ng. 2019. Abstractive summarization: A survey of the state of the art. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 9815–9822. https://doi.org/10.1609/aaai.v33i01.33019815

[12] Honnibal M, Montani I. spaCy Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[13] You, Yongjian, Weijia Jia, Tianyi Liu, and Wenmian Yang. 2019. Improving abstractive document summarization with salient information modeling. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2132–2141. https://doi.org/10.18653/v1/P19-1205

[14] Biswas, P. and Iakubovich, A., 2022. Extractive Summarization of Call Transcripts. [online] arXiv.org. Available at: https://arxiv.org/abs/2103.10599¿

[15] Kingma, D. and Ba, J., 2022. Adam: A Method for Stochastic Optimization. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1412.6980¿ [Accessed 12 May 2022].

[16] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. Text Summarization Branches Out (2004).

[17] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008.

[18] Zhang, Haoyu, Jianjun Xu, and Ji Wang. 2019. Pretraining-based natural language generation for text summarization. arXiv preprint arXiv:1902.09243. https://doi.org/10.18653/v1/K19-1074

[19] Xu, Song, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1355–1362. https://doi.org/10.18653/v1/2020.acl-main.125

[20] Govindan, A., Ranjan, G. and Verma, A., 2021. Unsupervised Named Entity Recognition for Hi-Tech Domain [online] http://dx.doi.org/10.5121/csit.2021.111917.

[21] Hinton, G., Vinyals, O. and Dean, J., 2022. Distilling the Knowledge in a Neural Network. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1503.02531¿ [Accessed 15 May 2022].

[22] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P., 2022. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1910.10683¿ [Accessed 15 May 2022].

[23] Govindan, A., Ranjan, G. and Karthik, A., 2019. Continuous learning mechanism of NLU-ML models boosted by human feedback [online] http://dx.doi.org/10.1109/ICCIDS.2019.8862102.