Text Summarization for Policies

Ezana Tesfaye, Liubov Tovbin, Sadia Yousafzai, Swetha Shiva Shankar Reddy
Computer Engineering Department
San Jose State University, San Jose, CA
Email: ezana.tesfaye@sjsu.edu, liubov.tovbin@sjsu.edu, sadia.yousafzai@sjsu.edu,
swetha.shivashnakarreddy@sjsu.edu

***Abstract*—With the rapid growth of the web and mobile services, users frequently come across unilateral contracts such as "Terms of Service" or "User Agreement." Most current mobile and web applications, such as Facebook, Google, and Twitter, require users to agree to "Terms and Conditions" or "Privacy Agreements." However, most of us rarely, if ever, read these conditions before signing. The problem is that policies are usually long and written in language that is hard for laypersons to comprehend. Besides, users agree to them without reading them carefully. Providing users with at least the essence of legally binding contracts helps them understand what users agree to before signing them. In this project, we aim to solve this problem with automatic text summarization. We base our work on the state-of-the-art pre-trained model, PEGASUS. We investigate the possibility to tailor it for a specific task of summarizing the legal policies. Based on our experiments, we conclude that given a small domain-specific dataset, it is better to fine-tune only a small part of the entire architecture, namely the last layer of the encoder and decoder. It prevents the model from overfitting and saves computations. Besides text summarization, we train our model to recognize user preference for the summary length. We use the ROGUE metric as well as partial human evaluation to assess the model performance. To make our text summarization engine accessible, we present it as a web application.**

***Index Terms*—NLP, Transformer, text summarization, PEGASUS, legal documents, fine-tuning**

I. Introduction

The textual data grows every day. There are 2.5 quintillion bytes of data created each day at the current pace, but that pace is only accelerating with the growth of the Internet of Things. Over the last two years alone, 90 percent of the world's data was generated, as reported by Forbes. At the beginning of 2020, the digital universe was estimated to consist of 44 zettabytes of data. Fig. 1 shows the amount and composition of data that is generated per minute in 2020. By 2025, approximately 463 exabytes would be created every 24 hours worldwide. With this volume of data, it has practically now become impossible for humans to read everything.

The majority of the texts, articles, journals, and contracts are online. However, clicking on 'I Agree' on those online contracts often written as terms and conditions in small print can cause massive problems. A recent survey, Fig. 2, conducted by Rainer Böhmef, UC Berkeley, and Stefan Köpsell of Dresden's Technische Universität, showed that only 7 percent of people read the terms and conditions carefully before signing them. One of the reasons identified is the complexity of text in them that many people find difficult to understand. That means we are waiving our contractual rights, and there could be severe

Fig. 1. Data generated per minute in 2020. Source: Adapted from Times Wiki

legal consequences if we do not comply with them. Currently,

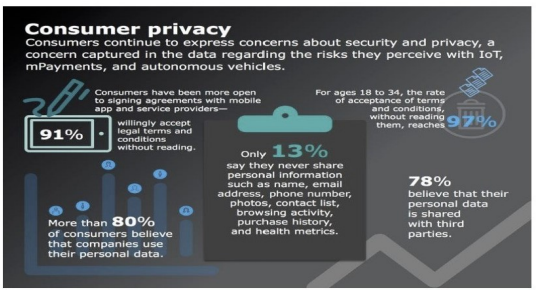

Fig. 2. Consumer Privacy Policy Agreement Statistics.
Source: Adapted from Google

two approaches are in use to tackle this problem. One is to hire a third-party vendor that summarizes the legal contracts document. Unfortunately, this approach is prone to human errors and is time-dependent. Another approach is to leverage artificial intelligence algorithms to generate a brief text from online policy documents while retaining crucial information. The later is known as Text Summarization (TS).

TS is one way to deal with the curse of dimensionality and acquire the knowledge effectively from the vast amount of textual information getting augmented in all domains. TS can be performed on a single text or a corpus of text. TS helps extract the most representative sentences or parts by applying transformations and condensing the text into a summary. The summary contains only the most significant parts of the original text while preserving critical information. TS is a complex task, and one of the most demanding research works these days, as it is tough to achieve summarization close to a human-generated one. [1].

TS can be of two types: extractive and abstractive. In extractive summarization, the important sentences and phrases are identified in a document and are used in the final summary. In abstractive summarization, the sentences in the final summary is not present in the original document but express the main concept of the original document. In recent years, researches on abstractive TS using Artificial Intelligence algorithms have increased.

One of the most popular research works is done by Google Brain, which implements a transformer-based encoder-decoder model. They pre-train the model on large text corpus and fine-tune on several different datasets. They achieved promising results and concluded that the summarization is close to as done by humans. However, the authors did not test the model to make summaries in plain English for legal documents such as "User Policies." Also, the summaries cannot be customized by users as per their needs.

This study aims to develop a web application that can summarize legal policy documents without losing essential information such that the generated summaries can be close to those written by humans. Also, we add a feature that allows users to choose the summary length: long or short.

This research study uses a dataset of contractual policy text data of 446 records [2] which will be increased in future.

The rest of the paper is organized as follows. Section II gives an overview of the current research and presents state-of-the-art techniques for text summarization. Section III describes our system's architecture, datasets, and evaluation methodology. Section IV explains the PEGASUS model's structure and performance. Section V and Section VI present the fine-tuning experiments. Section VIII gives a web application architecture overview, and Section IX concludes our work.

II. Related Works

TS algorithms can be classified into two main categories: abstractive and extractive.

A. Extractive Summarization

The significant part of the present research on TS focuses on extractive summarization [3], [4]-[8]. In extractive summarization, the final text summary may only consist of the words and sentences present in the original text. Currently, the mainstream approach to extractive summarization is to cluster sentences from the original text based on some similarity. Then, to rank each sentence or even each word in each cluster according to its importance in the text. The importance metric may vary from application to application. To rank words and sentences, researchers often use the Term Frequency - Inverse Document Frequency (TF-IDF) score and graph-based algorithms such as TextRank, LexRank etc.

In 2019, Mohd M. et al.[3] use a pre-trained Word2Vec model to find all the similar words for each word in a given sentence. This technique produced a collection of "Big Vectors," representing a sentence and all possible semantically similar words. Then they rank each cluster based on the sentence length, position, and a sum of TF-IDF scores for each word. Similarly, Rouane O. et al. [4] use k-means clustering to group sentences. They use the Apriori algorithm to mine frequent itemsets in each cluster, which improved the extractive summarization compared to mainstream text summarizers like TextRank, ItemSet Based Summarizer, and others. Mutlu B. et al. [5] make a step towards the neural networks approach in extractive summarization and compare it to two types of fuzzy inference systems by focusing on documents through manually picked features such as several sentences, title relevance, various terms, and inverse frequencies. Then they use a 4-layer perceptron with ReLU activation functions to determine the most relevant sentences in a given text. Jindal S. G. et al [6] propose an approach using the Rapid Automatic Keyword Extraction, TF-IDF score along with fuzzy C-means clustering. Also, they use rule-based sentence selection to choose sentences from the clusters. Merchant K. et al. [7] apply latent semantic analysis to capture concepts within a single document. They use two approaches: a single-document untrained and a multi-document trained approach depending on the type of input case (criminal or civil). S. P. Singh. [8] use a restricted Boltzmann machine to generate a shorter version of the original document by emphasizing the importance of relevant sentences. https://www.overleaf.com/project/5f250c6ad9e45f0001f76311 The main advantage of the extractive summarization that is based on the techniques described above is the fact that we do not need to train the model, and consequently, we do not need a large training dataset.

B. Abstractive Summarization

Abstractive summarization systems generate new phrases, possibly rephrasing or using words that were not in the original text. That task is more suitable for deep learning algorithms such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). The most recent approaches to abstractive summarization utilize Generative Adversarial Network (GAN) and a pre-trained language model, Bidirectional Encoder Representations from Transformer (BERT). BERT is Google's neural network-based technique for Natural Language Processing (NLP) pre-training [9]-[14], [16]-[19]. The main drawback of the abstractive summarization approach is that the model needs a large training dataset and also the authors have shown that the encoder-decoder based deep learning models like RNN perform well on shorter texts but fail to perform on longer text inputs and outputs.

C. State-of-the-Art

Text summarization is in high demand. Various Chrome browser plugins and web applications exist that perform automatic TS: *Open Text Summarizer* from splitbrain.org, *Text Summarization* from Dataiku, *Resoomer* and *Legal Leaf*.

Many tech giants such as Facebook, Google, and Salesforce dedicate their effort to the TS task. In 2015 Facebook proposed adding an attention mechanism in the decoder to the neural probabilistic language model from Bengio. Y. et al. [22]. Later, in 2018, the Facebook researchers proposed an interactive TS system that allows the users to choose summarization parameters such as length of a summary, entities of interest, style,

and the portion of the original text they want to summarize [23].

In 2018, the Salesforce researchers improved their earlier developed algorithm [12] by using a pre-trained language model to encourage paraphrasing in the final summaries [25]. In June 2020, the Google research team made the most recent leap in the abstractive TS task [26] by taking pre-training to the extreme. They base their TS model on Transformer seq2seq architecture. Transformer, also developed by Google [27], based solely on the attention mechanism and has proven to be the most successful seq2seq language model. In their recent work, the Google researchers decided to push forward Transformer performance on abstractive TS tasks by pretraining it on a large amount of data. They invented what they called "gap sentence generation." The idea is to take a text, "mask" some sentences, and train the model to predict those sentences. The process is called "self-supervised learning " and allows us to generate a massive, labeled dataset for TS without human intervention. After pretraining, they fine-tuned the model on twelve publicly available datasets for TS. The researchers found that the model required as few as 1000 examples for fine-tuning to perform sufficiently. With more training examples, the model performs so well that the human reader sometimes chooses the automatically generated summaries over human-generated ones.

The research body on machine learning-based TS has grown in recent years. That makes it possible to look back and critically evaluate the existing research framework and resulting progress. Kryściński W. et al. [20] analyze state-of-the-art TS models regarding a research setup: the quality of a dataset and evaluation metric correctness. The authors show that accessible datasets such as CNN/Daily Mail and Newsroom are pretty noisy. That impacts the training results, of course. They also show that the widely accepted ROUGE score metric is non-informative, and only weakly reflects a human judgment. Due to questionable datasets quality and evaluation metrics relevance, the progress of the most recent TS models "only slightly outperform" the baseline model that takes the first three sentences as a summary.

As for TS in the legal domain, not much research exists concerning abstractive summarization of the legal documents.[28] Legal texts are usually longer and have a specific structure and vocabulary, compared to the news articles. Those characteristics are the extra challenges when it comes to an automatic abstractive TS.

In this project, we aim to develop a web browser plugin that can summarize legal policy documents, without losing significant information such that the generated summaries can be compared with human-written summaries.

III. Project Design

A. Project Architecture

The user provides a text input that requires summarising and defines user preferences, such as the entities of interest and length of the final summary. The entities of interest tell the TS model to focus on the specific keywords in the original text. The idea of a user-controlled TS comes from the Facebook AI research publication released in July 2018 [23]. The word embedding module translates each word in the input into a vector representation. The term "embedding" usually refers to the learned vector representation using neural networks. Some of the most popular neural network architectures to learn word embeddings are Word2vec and GloVe [29]. We can choose whether to train those word embedding models on our documents' corpus or use available pre-trained versions [30], [31]. The core module is an abstractive TS system that takes input as preprocessed and vectorized text, along with the user summarizing preferences to produce a clear, concise summary that focuses on user-defined entities.

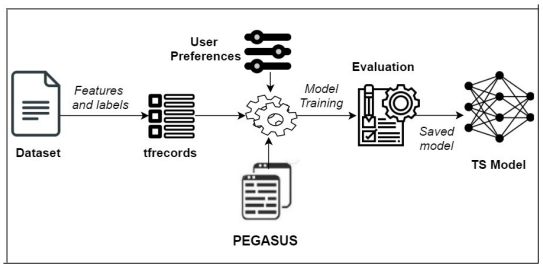

Fig. 3. Overall System Architecture

The goal of this project is to determine the best architecture for the abstractive TS model. We expect the Transformer-based PEGASUS architecture with pretraining method from Google AI research to perform the best. We plan to enhance it with the user-defined parameters following the idea of "controlled summarization" presented by Facebook AI research in 2018 [23].

	AESLC	CNN/Daily Mail	Legal
Train set	14,430	287,113	356
Test set	1,906	11,490	45
Validation set	1,960	13,368	45

Fig. 4. The comparison of three datasets: AESLC, CNN/Daily Mail, and Legal.

B. Datasets

The preliminary requirement of any data science project is to obtain a dataset. In this project, we are working with three datasets: AESLC, CNN/Daily Mail, and Legal dataset. Fig. 4 gives a comparison for all three. The train-validation-test split is 80-10-10%. The AESLC and CNN/Daily Mail datasets are publicly accessible through TensorFlow Datasets API, whereas the Legal dataset is handcrafted by Manor L. et al. [2].

C. Implementation

We implemented the project and performed several evaluations using HPC Cluster and Google Colab. Details are given below:

1) TFRecords: The dataset that contained legal texts and their target summarization was given in .csv format. The input data format that PEGASUS model supported was .tfrecord and .tdfs. We developed script in python to convert the dataset from .csv format to .tfrecords. The dataset was divided into train, test and validation using a ratio of 80:10:10.

2) HPC Cluster: SJSU provides high computing power servers on which students can run their code. We leveraged parallel processing capabilities and executed the code. We executed our program on GPU node (NVIDIA P-100) with memory of 256GB. Pegasus model and its libraries were installed from Github website (https://github.com/google-research/pegasus). All programs are developed using Python3 language in Unix environment.

3) Google Colab: Google Colab or Google Colaboratory is a product from Google which allows user to write and execute Python code using GPUs. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. We executed some experiments on Google Colab and uploaded our Python3 notebooks to compare performance with HPC Cluster.

D. Evaluation Methodology

The evaluation methodology is very particular to the topic of the project. It helps us to understand the performance and accuracy of the model. TS has a wide variety of evaluation metrics for examining the summaries which involve human judgments for various quality metrics such as conciseness, readability, context, grammar, etc. Once the system generates summaries, it is very crucial to evaluate these qualities in the summary. The evaluation criteria are as follows: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) BLEU (bilingual evaluation understudy) Precision (P) Recall (R) F measure (F1 score) ROUGE is one of the most promising and one of the widely used evaluation metrics for TS. It uses statistical approaches that consider the overlapping units such as n-gram co-occurrences, word sequences, redundancy, and word pairs to automatically measure the quality of system generated summaries against a set of human-generated (ideal) summaries. ROUGE comes in five different variants: ROUGE-N, ROUGE-L, ROUGE-S, ROUGE-W, ROUGE-SU which determine specific performance characteristics of a summarization system. In our project, we will be evaluating the performance of our system using the ROUGE scores obtained for the generated summaries. Generally, the ROUGE score measures the precision i.e., the number of words and/or n-grams seen in the human-generated summaries that also appear in the system generated summaries. So, if we have many words and/or n-grams in the human-generated summaries also appearing in the system generated summaries we will observe high ROUGE scores. Therefore, ROUGE can be considered as one of the standard metrics used in evaluating summaries which are based upon the lexical overlap between the gold standard human-generated summaries and the system-generated summaries. In spite of all these advantages, ROUGE does have limitations as follows: The automatic summaries generated by the system need to be compared with the human-generated summaries for quality and reliability. The generated summaries may suffer from readability and lack of linguistic qualities which is not taken into account for the ROUGE score computation. ROUGE is not an effective evaluation criterion to rely on when it comes to terminologies and paraphrasing as it is computed solely relying on the lexical relationships between words and phrases. ROUGE is not one of the best metrics to choose from when it comes to checking redundant information. ROUGE scores purely depend on the length of the generated summary i.e, the longer the system summary comparative to the human-generated summary, the higher will be the ROUGE score which could be misleading.

IV. Baseline Performance

For the baseline approach, we choose the most recent development in abstractive TS that involves pre-training the Transformer-based model on a large, specially designed dataset. The idea belongs to the Google AI research team [26]. The authors published their work in June 2020, where they present the PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization Sequence-to-sequence models.

The novel approach for creating an annotated dataset for supervised abstractive summarization is as follows. Take an original text and delete ("mask") a selected number of sentences from it. The selected sentences represent the desired output. Train the model to predict those sentences. The authors consider three strategies for selecting m the sentences from the original text:

- First m
- Randomly selected
- Based on the ROUGE score between each sentence and the rest of the text

The authors identify the last strategy as an optimal one and consider two options for selecting m sentences based on the ROUGE score:

- Score each sentence individually and choose top m
- Use the greedy algorithm to choose m

After pre-training the model, they experiment with fine-tuning it on the twelve datasets for abstractive TS. All the mentioned datasets are publicly accessible through the TensorFlow library [28]. The difference between the proposed Gap Sentence Generation (GSG) approach and the BERT's Masked Language Model (MLM) [15] is as follows. The MLM masks randomly 15% of the words in a given text, whereas GSG masks the complete, strategically chosen sentences. However, for the sake of comparison, the authors experiment with pre-training their model with both GSG and MLM objectives separately as well as combined. To determine how a pre-training and size of the model improves the summarization results, the authors experimented with three variants: Transformer Base - Transformer without pre-training. PEGASUS Base - the pre-trained Transformer with 223M parameters. PEGASUS Large - the pre-trained Transformer with 568M parameters. In addition, the authors experiment with two datasets pre-training and GSG hyperparameter, the gap-sentence ratio. The approach showed dramatic improvement in the performance

on abstractive TS tasks achieving the "human performance on multiple datasets using human evaluation."

Because of the nature of pre-training with extracted gap-sentences, the PEGASUS model initially produces only extractive summaries. To make PEGASUS produce abstractive summaries, we need to train it further on a dataset that supplies an abstractive summary for a given text. This additional training we call "fine-tuning."

The next section describes the fine-tuning experiments that aim to discover the best approach to fine-tune a model with so many parameters having limited resources: time, computation power, data.

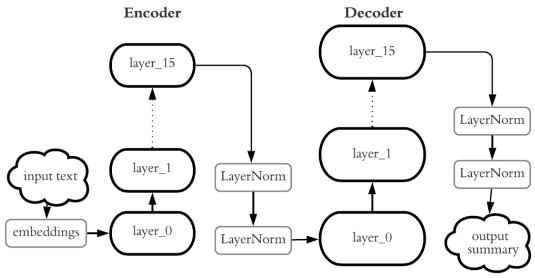

Fig. 5. The details of the PEGASUS architecture. The arrows show the data flow direction.

V. Fine-tuning Experiments

As shown on the Fig.5, the PEGASUS model has 15 identical layers in the encoder and 15 identical layers in the decoder. The encoder and decoder's layers are not the same (see Fig.6). Also, the architecture has the embedding layer that comes first, before the encoder. The available open-sourced PEGASUS model contains a total of 568M parameters. It takes a lot of resources to fine-tune the whole model. In the fine-tuning experiments described below, we investigate whether it is possible to get satisfactory results after fine-tuning only a part of the model.

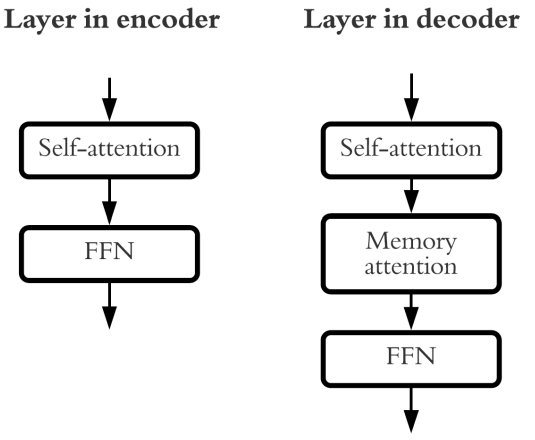

Fig. 6. The inner structure of each one of the 15 layers in the PEGASUS architecture. The layers in the encoder and decoder are different. Each layer in the decoder has an additional component, the Memory attention part.

A. Fine-tuning the Embedding Layer

The embedding layer is the first layer in the model. It takes the tokenized text as input. The purpose of the embedding layer is to learn vector representation for each word in the vocabulary—this approach yields dense word vector representation in contrast to other vectorization techniques.

Certainly, the input vocabulary plays a fundamental role in defining the final word vectors space. In this experiment, we fine-tune only the embedding layer to see how much it affects a whole model's summarization ability.

B. Fine-tuning the Last Layer

Besides the embedding layer, we have an option to fine-tune the encoder's last layer, the decoder's last layer, or both. The choice of the network's last layer for the fine-tuning is common in transfer learning. It is computationally faster to get updates on the weights in the last layer. Also, the last layers in the neural network are the ones to capture specific details in the training data. Those aspects are helpful when the available dataset is relatively small. Since we aim to obtain a model that can produce abstractive summaries for legal policies given a dataset of only 446 records, it is crucial to determine how we can get the most out of training.

C. Results for the Fine-tuning Experiments

Fig. 12 shows an example of input from the AESLC evaluation set and the output summaries for each fine-tuning variation described above, together with the non-fine-tuned model. Also, we include in the comparison the fully fine-tuned model that authors of the PEGASUS made available for public access.

As expected, the non-fine-tuned model chooses whole sentences from the input. However, after fine-tuning, the final summaries are much shorter. Judging by the input and summaries' content, the fine-tuning of the last layer of encoder and decoder yields a satisfactory result producing a concise summary without losing essential information.

Fig. 13 shows how the ROUGE scores vary when we do the fine-tuning experiments on the AESLC dataset. Fine-tuning only the embedding layer gives a significant jump in the ROUGE score. However, we also get a high percentage of empty predictions, where the model fails to produce any summary. Fig. 14 shows how the ROUGE score varies when we do the fine-tuning experiments on the CNN/Dailymail dataset. This dataset is more than ten times larger than the AESLC dataset with much longer inputs. Nevertheless, the results here are about the same as for the AESLC dataset. Judging by the ROUGE scores, the fine-tuning the last layer of the encoder and decoder, yields performance comparable to the performance of fully fine-tuned model with no empty outputs.

D. Human Evaluation using Qualtrics Survey

The ROUGE scores don't account for the terminologies and paraphrasing as it is computed purely based on the relationship between the words and phrases. ROUGE scores mostly depend

on the length of the summary i.e. the longer the length of the summary, the larger is the ROUGE score. Considering the drawbacks of ROUGE scores, we conducted a survey where the users were asked to rate the summaries between 1-5 where Summary 1 were the Targets, Summary 2 were the Predictions. The raters were asked to rate without having explicitly mentioned which summary is machine-generated and human-generated. We performed this experiment on a total of 50 users to rate a set of four summaries for the results obtained by fine-tuning the last layers on the pre-trained legal dataset model by the legal dataset which showed the promising results compared to the other fine-tuned experiments. We found that the raters showed not much preference towards human-generated summaries / targets.

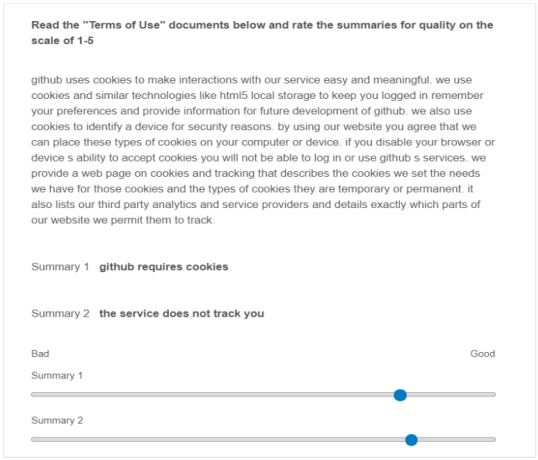

Fig. 7. Qualtrics survey showing an example of a GitHub terms of use with its summary. Summary 1 stands for Targets, Summary 2 stands for the Predictions

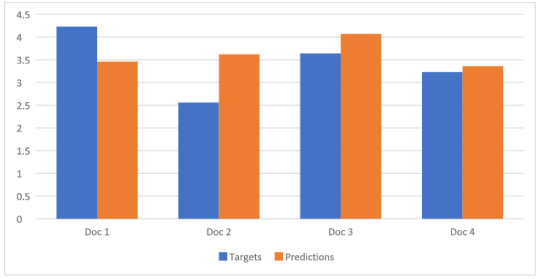

Fig. 8. The results showing the average values from the Survey for a set of four documents and their respective summary ratings.

VI. Fine-tuning on Legal Dataset

Since our legal domain dataset is only 446 records, we perform a series of experiments to find out how beneficial it is to fine-tune the model on such a small dataset.

A. Zero-shot Learning

First, we check how the non-fine-tuned model performs on the Legal dataset. When we present the machine learning model with the unseen domain's data, it is called zero-short learning. That gives us a baseline to compare the model performance after fine-tuning.

B. Fine-tuning on Non-legal Dataset

Next, we leverage the availability of the large dataset to perform legal text summarization. We take the model that has been fully fine-tuned on the CNN/Daily Mail dataset and evaluate it on the Legal dataset.

C. Fine-tuning on Legal Dataset Only

In this experiment, we fine-tune the model on the Legal dataset. The training set here is only 356 points (see Fig. 4). we compare the results after 10 and 20 training epochs. As we concluded in the previous section, the model produces satisfying summaries after only fine-tuning the last layer's encoder and decoder. Thus, we include this option in comparison.

D. Fine-tuning on both, Legal and Non-legal Dataset

The larger and more diverse fine-tuning dataset is beneficial since it prevents the model from overfitting. However, to tailor the model for the policies summarization task, we have to present it with the domain-specific dataset. Thus, we take the model that has been fine-tuned on the CNN/Daily Mail dataset and further fine-tune it on the Legal dataset.

E. Results for the Fine-tuning on Legal Dataset

Fig. 15 shows an input example from the Legal dataset and the the corresponding summaries from the following models:

1) non-fine-tuned model
2) model fine-tuned on the CNN/Daily Mail dataset only
3) model fine-tuned on the Legal dataset only
   - 10 epochs
   - 20 epochs
   - 20 epochs encoder and decoder last layer only
4) model fine-tuned on both, CNN/Daily Mail and Legal datasets
   - 10 epochs
   - 20 epochs

Fig. 16 shows the corresponding ROUGE scores for the above four models with 10 epochs-variation for model 3 and 20 epochs-variation for model 4. Fig. 17 shows the resulting ROUGE-1, ROUGE-2, and ROUGE-L scores for the three variations of model 3, where we fine-tuned on the Legal dataset only.

The non-fine-tuned model produces an extractive summary, as expected, because of the GSG pre-training. After 20 epochs of fine-tuning on the Legal dataset only, the model clearly overfits. Judging by the ROUGE scores and the quality of sampled summaries, the model fine-tuned on both the CNN/Daily Mail and the Legal dataset appears to produce the best result. The resulting summary is concise and does not miss important information. However, after fine-tuning the encoder-decoder last layer on the Legal dataset only, the model generates good enough summaries and does not seem to overfit.

VII. Summarization with Length Preference

A. User Preference Incorporation Technique

We borrowed the idea to incorporate user preferences into the TS model from the Facebook AI research paper [23]. In this work, we focus on user preference for the summary's length: "short" or "long." We define the "short summary" as a maximum of 64 words and "long summary" as a maximum of 256 words.

B. Dataset Creation and Training

To train the model to recognize user preference for a summary length, we need to construct a specific training dataset. Our original idea is to use PEGASUS as a tool to make a dataset containing the short and long summary versions of the same input text. We start by taking the CNN/Daily Mail dataset and do the following:

- In the PEGASUS code, set the hyperparameter *max_output_len* to 64 and fine-tune it on a given training dataset. In that way, we train the model to produce short summaries.
- Run the evaluation to obtain short summaries for each input in the evaluation dataset.
- Repeat the previous steps setting *max_output_len* to 256 to obtain long summaries for the same inputs in the evaluation dataset.
- Prepend each input with a keyword "Long" or "Short," according to the resulting summary and combine in one dataset. Putting the keyword at the beginning of the input is preferable because otherwise, it might get cut off if the input length exceeds the predefined *max_input_len* hyperparameter.
- Split the resulting dataset into training, evaluation, and test sets.
- Fine-tune the PEGASUS on the training set.
- Evaluate on the evaluation set.

Our resulting dataset contains 2000 entries, 1000 for short, and 1000 for long summaries. We use 80% for training and 10% for evaluation and testing, so the evaluation set contains 200 entries, 100 for short, and 100 for long summaries.

C. Results for the Summarization with Length Preferences

Two histograms on Fig. 7 shows two lengths distributions for summaries corresponding to the inputs with the keywords "Short" and "Long," respectively. The mean length for long summaries is 49.68 words. The mean length for short summaries is 35.35 words. We compare those two samples using T-test which yields a $p\text{-}value = 1.5 \times 10^{-10}$. Thus, we are more than 95% confident that the resulting difference in the mean for long and short lengths is not random, but due to our training.

Fig. 18 gives an input example from the Legal test set and two outputs that the model produced for "long" and "short" length preference. Even though the summaries' quality here is low due to insufficient training, the model can differentiate between user preference for length.

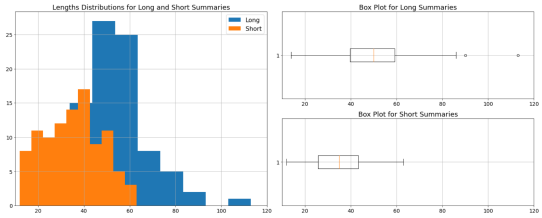

Fig. 9. Difference in lengths distributions for inputs with the keyword "Short" and inputs with the keyword "Long" after fine-tuning PEGASUS for user-defined summary length.

VIII. Web Application for an Automatic Policies Summarization

After we trained the model, the only last step was to make the model visible on the web. We converted the model to a saved model which is converted to a tensoflow.js.

A. Flow

- A pre-trained PEGASUS model is converted to tensorflow.js model.
- User opens the web application.
- Users can get a summary by uploading a .txt file or by copying pasting text.
- A user could could also select short or large summarizing style, short is chosen by default.
- The text is sent through the model and summarized text is given to the user as the output.

B. Components

- React: is a JavaScript library for building user interface.
- Tensorflow.js: A JavaScript library for training and deploying ML models in the browser and on Node.js.

C. Architecture

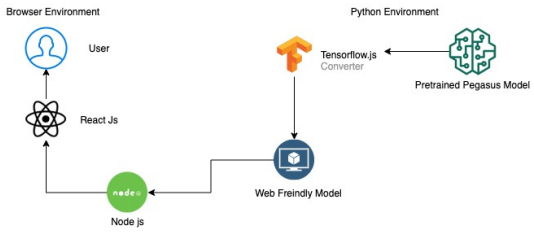

Fig. 10. Web Architecture

D. User Interface

Fig. 9, Fig. 10, and Fig. 11 demonstrate the graphical user interface for the text summarization web application.

IX. Conclusion

In this work, the authors aimed to leverage state-of-the-art automatic text summarization techniques to build a system that allows users to quickly get summaries for the long and hard-to-read legal policies such as Terms Of Use or User Agreements.

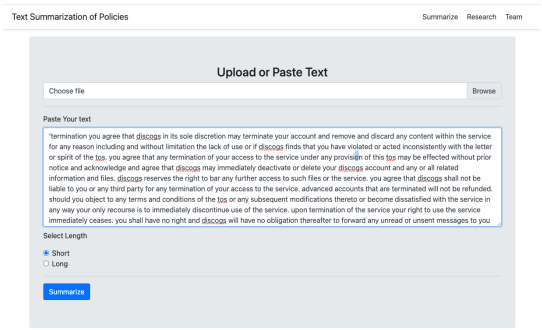

Fig. 11. Main Panel

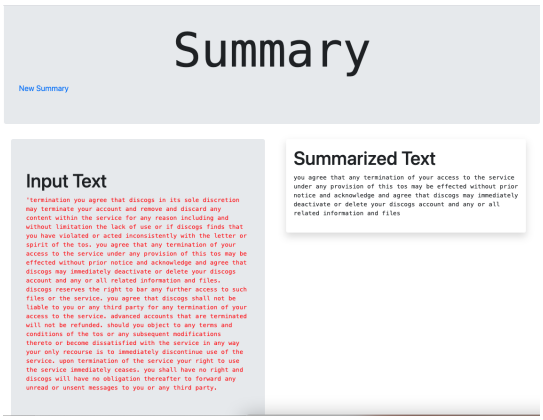

Fig. 12. Result Panel for the "Short" Summary

Fig. 13. Result Panel for the "Long" Summary

We based our system on the pre-trained Transformer-based model and explored how to adjust it for a specific task of policy summarization. Also, we used a state-of-the-art technique to incorporate user preferences for the summary, such as length. Based on our experiments, we conclude that given a small domain-specific dataset, it is better to fine-tune only a small part of the entire architecture, namely the last layer of the encoder and decoder. It prevents the model from overfitting and saves computations.

For future work, we propose finding a way to create an extensive and high-quality dataset for the legal domain since this obstacle remained present in this study.

ACKNOWLEDGMENT

The authors are deeply thankful to Professor Mahima Agumbe Suresh for her invaluable comments and assistance in conducting this study.

Input:
I thought you would be interested in the weather forecast for the Wedding.
The data includes max temp, max temp deviation from normal (ie if max is 88 and normal max is 86 max temp deviation is +2), min temp, min temp deviation from normal and percent of precipitation.
So far things are looking good.
We are getting a cold front this weekend (hitting Texas on Monday) and looks like things should be beautiful for next weekend.
I wouldn't be that worried about the rain either - the reason the POP is so high right now is tropical moisture in the Gulf of Mexico and that will blow through this weekend.
After that the tropics settle down and it is much less likely to get tropical moisture.

Target: Weather

Predictions:

No fine-tuning	**Embeddings only**	**Encoder's last layer**
We are getting a cold front this weekend (hitting Texas on Monday) and looks like things should be beautiful for next weekend.	Weather Forecast	Wedding Weather Forecast
Decoder's last layer	**Encoder & Decoder last layer**	**Full fine-tuning**
Wedding	Wedding Weather Forecast	Wedding Weather

Fig. 14. An example of the input from the AESLC evaluation set, the target summary, and the model's outputs before and after partial and full fine-tuning. The highlighted text helps to track similarities between input and the produced summaries.

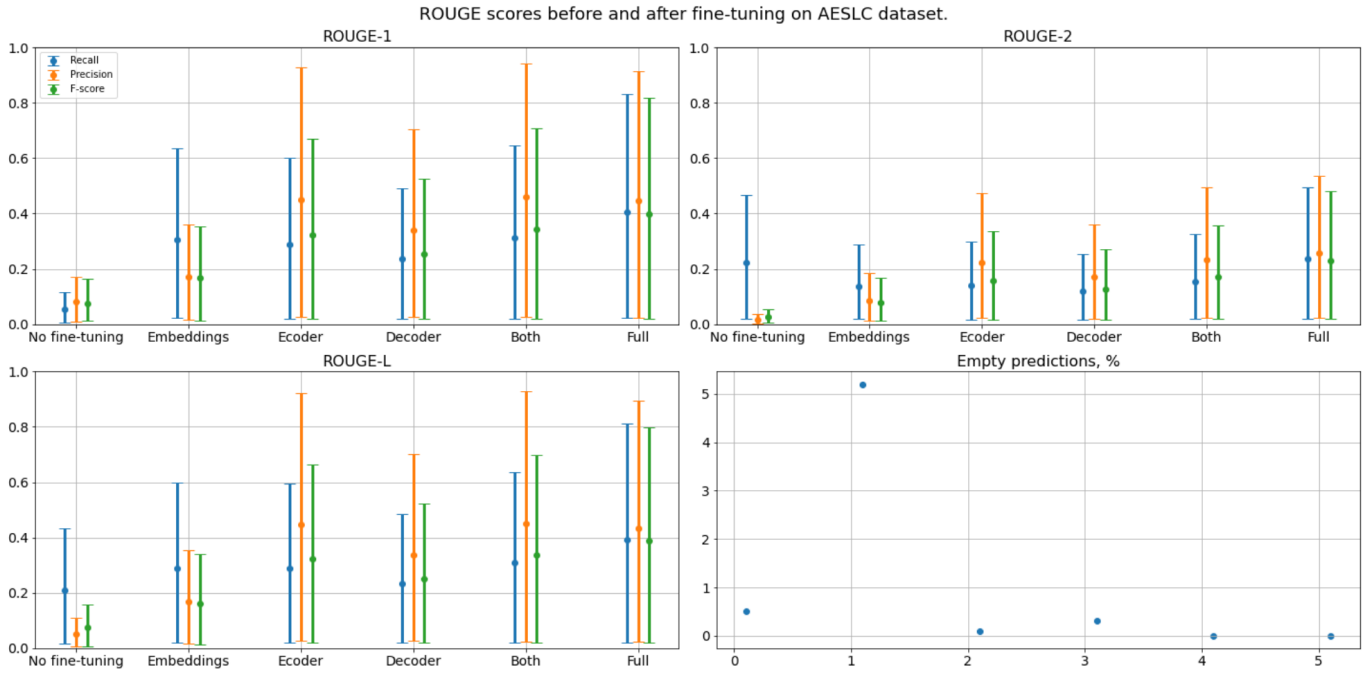

Fig. 15. Results of the fine-tuning experiments on the AESLC dataset. The top two and bottom left graphs show the ROUGE-1, ROUGE-2, and ROUGE-L scores before and after fine-tuning different PEGASUS architecture parts: embeddings layer, encoder's last layer, decoder's last layer, both, encoder and decoder last layer, and full model. The bottom graph on the right shows the percentage of empty outputs that the model occasionally produces.

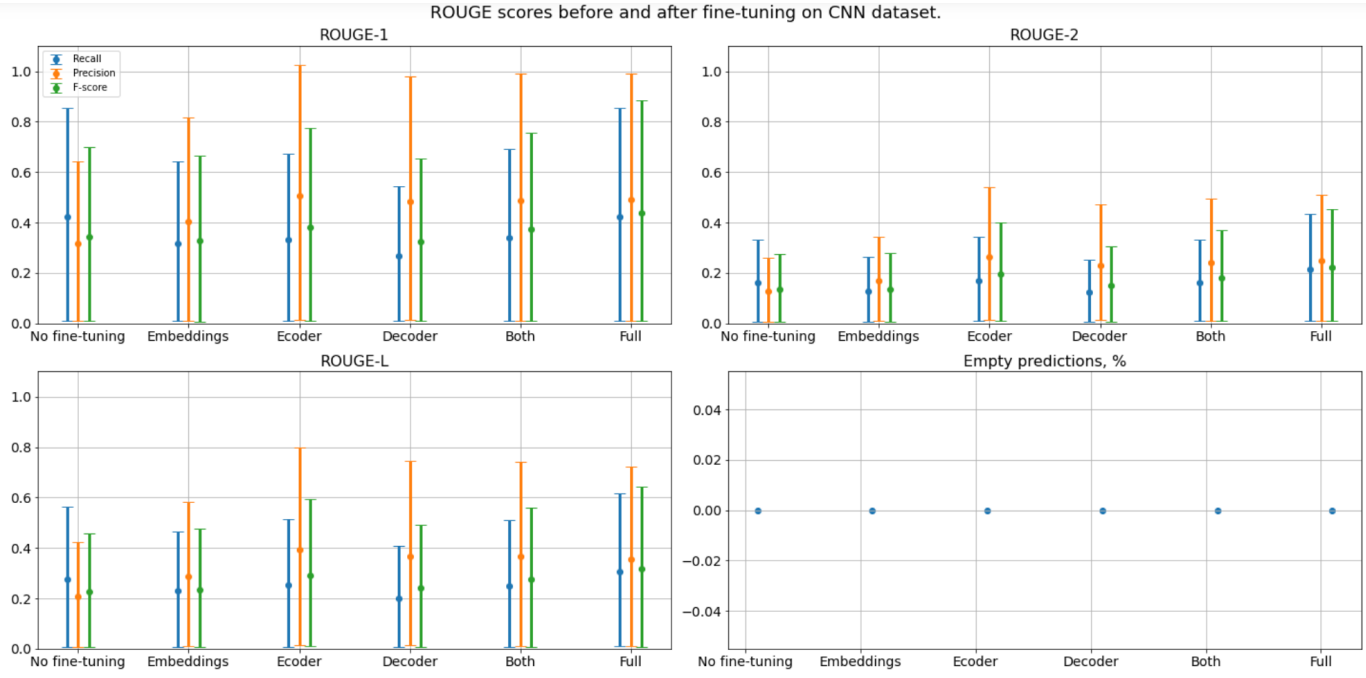

Fig. 16. Results of the fine-tuning experiments on the CNN/Daily Mail dataset. The top two and bottom left graphs show the ROUGE-1, ROUGE-2, and ROUGE-L scores before and after fine-tuning different PEGASUS architecture parts: embeddings layer, encoder's last layer, decoder's last layer, both, encoder and decoder last layer, and full model. Here, no empty outputs ever occurred.

Input:
termination you agree that discogs in its sole discretion may terminate your account and remove and discard any content within the service for any reason including and without limitation the lack of use or if discogs finds that you have violated or acted inconsistently with the letter or spirit of the tos. you agree that any termination of your access to the service under any provision of this tos may be effected without prior notice and acknowledge and agree that discogs may immediately deactivate or delete your discogs account and any or all related information and files. discogs reserves the right to bar any further access to such files or the service. you agree that discogs shall not be liable to you or any third party for any termination of your access to the service. advanced accounts that are terminated will not be refunded. should you object to any terms and conditions of the tos or any subsequent modifications thereto or become dissatisfied with the service in any way your only recourse is to immediately discontinue use of the service. upon termination of the service your right to use the service immediately ceases. you shall have no right and discogs will have no obligation thereafter to forward any unread or unsent messages to you or any third party.

Target: discogs can delete your account without prior notice and without a reason.

Predictions:

No fine-tuning (zero-shot)	**Fine-tuning on CNN dataset only**	**Fine-tuning on Legal dataset only**	**Fine-tuning on both datasets**
you agree that any termination of your access to the service under any provision of this tos may be effected without prior notice and acknowledge and agree that discogs may immediately deactivate or delete your discogs account and any or all related information and files.	discogs in its sole discretion may terminate your account and remove and discard any content within the service . discogs reserves the right to bar any further access to such files or the service .	**10 epochs on Legal:** you agree that discogs can terminate your account for any reason. **20 epochs on Legal:** discogs can delete your account without prior notice and without a reason. **Encoder and decoder last layer only:** you agree that discogs may terminate your account without prior notice.	**10 epochs on Legal:** you agree that discogs in its sole discretion may terminate your account and remove and discard any content within the service for any reason. **20 epochs on Legal:** the service can delete your account without prior notice and without a reason.

Fig. 17. An input example from the Legal evaluation set shows summaries that the model produces before any fine-tuning (zero-shot); after fine-tuning on the non-legal dataset (CNN/Daily Mail), fine-tuning on the Legal dataset only, and after fine-tuning on both, CNN Daily Mail and Legal. The highlighted text helps to track similarities between input and the produced summaries.

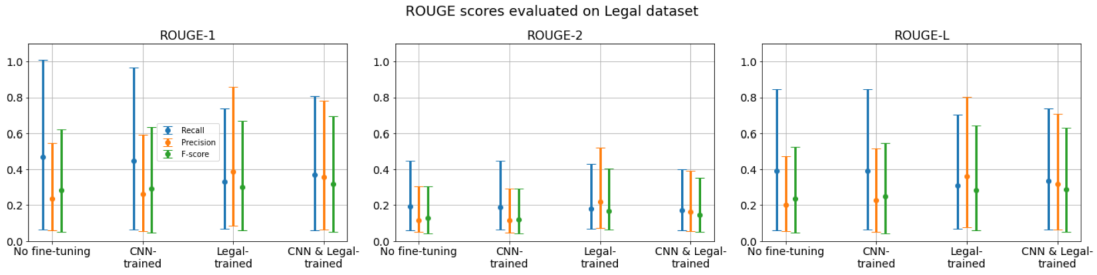

Fig. 18. Evaluation on Legal dataset. The ROUGE-1 scores before and after fine-tuning on CNN/Daily Mail and/or Legal datasets. The Legal dataset score is after 10 fine-tuning epochs and CNN & Legal is after 20 fine-tuning epochs

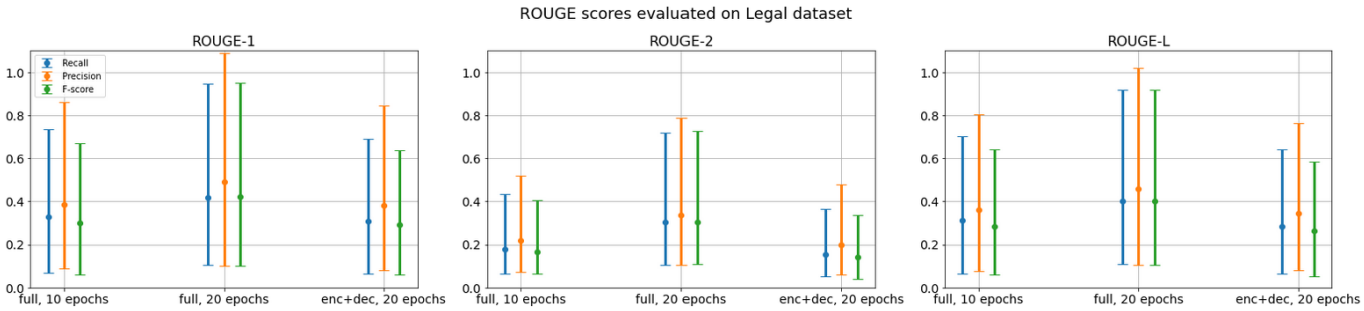

Fig. 19. Evaluation on Legal dataset. The ROUGE-1, ROUGE-2, and ROUGE-L scores after fine-tuning on Legal dataset for 10 and 20 epochs full model and 20 epochs encoder-decoder last layer only.

Input:
for example when you go to a website with a like button we need to know who you are in order to show you what your facebook friends have liked on that site. the data we receive includes your user id the website you re visiting the date and time and other browser related info. if you re logged out or don t have a facebook account and visit a website with the like button or another social plugin your browser sends us a more limited set of info. for example because you re not logged into facebook you ll have fewer cookies than someone who s logged in. like other sites on the internet we receive info about the web page you re visiting the date and time and other browser related info. we record this info to help us improve our products. as our data policy indicates we use cookies to show you ads on and off facebook. we may also use the info we receive when you visit a site with social plugins to help us show you more interesting and useful ads.

Long summary:
the data we receive includes your user ID the website you re visiting the date and time and other browser related info. if you re logged out or don t have a facebook account and visit a website with the like button or another social plugin your browser sends us a more limited set of info. if you re not logged into facebook you ll have fewer cookies than someone who s logged in.

Words count:
74

Short summary:
for example when you go to a website with a like button we need to know who you are in order to show you what your facebook friends have liked on that site .

Words count:
34

Fig. 20. An input example from the Legal test set and two outputs that the model produced for ”long” and ”short” length preference. Even though the summaries' quality here is low due to insufficient training, the model can differentiate between user preference for length.

REFERENCES

[1] P. Janjanam and C. P. Reddy, "Text Summarization: An Essential Study," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 2019 [Online].Available: https://ieeexplore.ieee.org/abstract/document/8862030. [Accessed: 01-Jul-2020]

[2] L. Manor and J. J. Li, "Plain English Summarization of Contracts," 02-Jun-2019. [Online]. Available: https://arxiv.org/abs/1906.00424. [Accessed: 29-Jul-2020].

[3] M. Mohd, R. Jan, and M. Shah, "Text document summarization using word embedding," Expert Systems with Applications, vol. 143, p. 112958, 2020 [Online]. Available: https://doi.org/10.1016/j.eswa.2019.112958

[4] O. Rouane, H. Belhadef, and M. Bouakkaz, "Combine clustering and frequent itemsets mining to enhance biomedical text summarization," Expert Systems with Applications, vol. 135, pp. 362–373, 2019 [Online]. Available: https://doi.org/10.1016/j.eswa.2019.06.002

[5] B. Mutlu, E. A. Sezer, and M. A. Akcayol, "Multi-document extractive text summarization: A comparative assessment on features," Knowledge-Based Systems, vol. 183, p. 104848, 2019 [Online]. Available: https://doi.org/10.1016/j.knosys.2019.07.019

[6] S. G. Jindal and A. Kaur, "Automatic Keyword and Sentence-Based Text Summarization for Software Bug Reports," in IEEE Access, vol. 8, pp. 65352-65370, 2020, doi: 10.1109/ACCESS.2020.2985222.

[7] K. Merchant and Y. Pande, "NLP Based Latent Semantic Analysis for Legal Text Summarization," NLP Based Latent Semantic Analysis for Legal Text Summarization - IEEE Conference Publication, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8554831. [Accessed: 01-Jul-2020].

[8] S. P. Singh, A. Kumar, A. Mangal, and S. Singhal, "Bilingual automatic text summarization using unsupervised deep learning," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), 2016 [Online].
Available:https://ieeexplore.ieee.org/stamp
/stamp.jsp?tp=arnumber=7754874.[Accessed:01-Jul-2020]

[9] P. Janaszkiewicz, J. Krysińska, M. Prys, M. Kieruzel, T. Lipczyński, and P. Różewski, "Text Summarization For Storytelling: Formal Document Case," Procedia Computer Science, vol. 126, pp. 1154–1161, 2018.

[10] R. Nallapati, B. Zhou, C. D. Santos, C. Gulcehre, and B. Xiang, "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond," Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, 2016 [Online]. Available: https://arxiv.org/pdf/1602.06023.pdf. [Accessed: 01-Jul-2020]

[11] P. Li, W. Lam, L. Bing, and Z. Wang, "Deep Recurrent Generative Decoder for Abstractive Text Summarization," Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017 [Online]. Available: https://www.aclweb.org/anthology/D17-1222.pdf

[12] R. Paulus, C. Xiong, and R. Socher, "A Deep Reinforced Model for Abstractive Summarization," arXiv.org, 13-Nov-2017. [Online]. Available: https://arxiv.org/abs/1705.04304. [Accessed: 01-Jul-2020].

[13] L. Wang, J. Yao, Y. Tao, L. Zhong, W. Liu, and Q. Du, "A Reinforced Topic-Aware Convolutional Sequence-to-Sequence Model for Abstractive Text Summarization" Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, 2018 [Online]. Available: https://arxiv.org/pdf/1805.03616.pdf. [Accessed: 01-Jul-2020]

[14] B. Rekabdar, C. Mousas, and B. Gupta, "Generative Adversarial Network with Policy Gradient for Text Summarization," 2019 IEEE 13th International Conference on Semantic Computing (ICSC), 2019 [Online]. Available: https://web.ics.purdue.edu/ cmousas/papers/conf19-IEEEICSC.pdf. [Accessed: 01-Jul-2020]

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv.org, 24-May-2019. [Online]. Available: https://arxiv.org/abs/1810.04805. [Accessed: 01-Jul-2020].

[16] Y. Chen, Z. Gan, Y. Cheng, J. Liu, and J. Liu, "Distilling the Knowledge of BERT for Text Generation", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/1911.03829v1. [Accessed: 30- Jun- 2020]

[17] M. Moradi, G. Dorffner, and M. Samwald, "Deep contextualized embeddings for quantifying the informative content in biomedical text summarization," Computer Methods and Programs in Biomedicine, vol. 184, p. 105117, 2020 [Online]. Available: https://doi.org/10.1016/j.cmpb.2019.105117

[18] Z. Liang, J. Du, and C. Li, "Abstractive Social Media Text Summarization using Selective Reinforced Seq2Seq Attention Model," Neurocomputing, 2020 [Online].
Available: https://www-sciencedirect-com.libaccess.sjlibrary.org
/science/article/pii/S0925231220307840. [Accessed: 01-Jul-2020]

[19] M. Yang, X. Wang, Y. Lu, J. Lv, Y. Shen, and C. Li, "Plausibility-promoting generative adversarial network for abstractive text summarization with multi-task constraint," Information Sciences, vol. 521, pp. 46–61, 2020.

[20] W. Kryściński, N. Keskar, B. McCann, C. Xiong and R. Socher, "Neural Text Summarization: A Critical Evaluation", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/1908.08960. [Accessed: 30- Jun- 2020]

[21] A. Rush, S. Chopra and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/1509.00685. [Accessed: 30- Jun- 2020]

[22] Y. Bengio and J.-S. Senecal, "Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model," IEEE Transactions on Neural Networks, vol. 19, no. 4, pp. 713–722, 2008 [Online]. Available: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf. [Accessed: 01-Jul-2020]

[23] A. Fan, D. Grangier, and M. Auli, "Controllable Abstractive Summarization," Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, 2018 [Online]. Available: https://arxiv.org/pdf/1711.05217.pdf. [Accessed: 01-Jul-2020].

[24] Gehring, M. Auli, D. Grangier, D. Yarats and Y. Dauphin, "Convolutional Sequence to Sequence Learning", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/1705.03122. [Accessed: 30- Jun- 2020]

[25] W. Kryściński, R. Paulus, C. Xiong, and R. Socher, "Improving Abstraction in Text Summarization," ACL Anthology, 01-Jan-1970. [Online]. Available: https://www.aclweb.org/anthology/D18-1207/. [Accessed: 01-Jul-2020].

[26] Zhang, Y. Zhao, M. Saleh and P. Liu, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization", arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/1912.08777. [Accessed: 30- Jun- 2020]

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need - arXiv," arxiv.org, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762. [Accessed: 01-Jul-2020].

[28] A. Kanapala, S. Pal, and R. Pamula, "Text summarization from legal documents: a survey," Artificial Intelligence Review, vol. 51, no. 3, pp. 371–402, 2017.