Building an Optimized algorithm that provides summaries of legal documents

Aman Burman[1] and Eric Bradford[2,3]

[1] *Dubai College*
[2] *Technical Product Manager at Apple*
[3] *Masters of Engineering from Massachusetts Institute of Technology*

ABSTRACT

We analyze the accuracy of various NLP algorithms in providing text summarization and fine-tune a particular model on a dataset to provide accurate text summaries of legal documents. The legal industry is built around documents as they provide evidence and reduce doubt in the court. Due to the large volume of documentation in the legal industry, the processing and summarization of these documents is important to a number of individuals. For example, lawyers, clients and professionals may need access to summaries of the documents for reference to similar cases. In this paper, we have developed an algorithm that trains the T5 algorithm on the legal domain in order to create more accurate summaries of legal documents. We were able to create a user interface that allows for the input of documents and makes use of the algorithm we created to output a summary of the document which can be copied by the user.

1. INTRODUCTION

Automation is key in countless industries including (Yarlagadda (2017)) healthcare, manufacturing and the business sector. In recent years, there has been a technological revolution (Brownsword (2008)) specifically in the legal industry with firms and courts now using artificial intelligence to quickly resolve laborious tasks, document automation to make file management more effective and even blockchain for smart contracts - programs stored on the blockchain that are activated once preset conditions are met. (Zheng et al. (2020))

In particular, machine learning and natural language processing (Haney (2020)) can be essential in the summarization of legal documentation. In the United States alone, the Supreme Court had 66 case filings (of the United States (2021))for 2021 alone with hundreds of millions of court cases in the state trialed courts and hundreds of thousands in the federal trial courts. With each case, there are countless documents ranging from dockets, pleadings, motions, memoranda, briefs, orders, and expert testimony. The process of individuals manually going through each of the documents and picking out key details can be extremely labour intensive and very inefficient (Roitblat et al. (2010)). Some court cases do have opinions, which is the analysis for the decisions the judge (Maltz (2000)) has come to. There are also third-party companies as well as researchers that have come up with algorithms to summarize court cases (Bhattacharya et al. (2019)). However, we plan on creating an algorithm that creates the most accurate summaries specific to the legal industry which would be available for the general public to make use of.

So far, most of the available text summarization models and algorithms make use of an extractive-based summarization where important parts of text are summarized. (Pilault et al. (2020)) Abstractive-based summarization is a more difficult process of summarization as it depends on creating new sentences based on a machine's understanding of text. A lot of research is currently being done in the field of abstractive-summarization (Moratanch & Chitrakala (2016)).

To tackle this issue, I have created a dataset on motion law documents and fine-tuned a pre-trained optimized text summarizer that is able to condense text and provide accurate summaries that encompass all of the information in a document. In order to create this algorithm, I explored various supervised and unsupervised machine learning models including: TextRank, Latent Semantic Analysis, T5 transformers, BART transformers, GPT-2 Transformers, XLM Transformers, PEGASUS and BERT. By evaluating the summaries using the ROUGE evaluative metric, I was able to determine which models were most efficient and fine-tuned them for text-summarization of legal documents.

Additionally, I created an app that effortlessly allows users to upload court case documents and applies the fine-tuned model to it to generate summaries. The app provides a user interface for the refined text summarizer model we trained in the paper.

2. BACKGROUND

2.1. *Models*

The majority of text summarization in the past has been extractive text summarization. One of the first systems built specifically for the legal industry was "Fast Legal EXpert CONsultant" (FLEXICON). (Polsley et al. (2016)). FLEXICON recognized technical legal language through a combination of in-built legal knowledge and computational linguistic methods. (Gelbart & Smith (1991)). The system implemented electronic ”headnotes” which was revolutionary.

Term frequency–inverse document frequency (TF-IDF) also implements extractive summarization. It makes use of probabilistic and statistical measures to parametrize the frequency of the appearnace of certain words to build the summarization of texts which is a very common method for many models. It assigns scores based on the frequency. TF-IDF is very similar to PageRank but is used in a vastly different method (Kore et al. (2020))

Keyword extraction is another method that many models use. Keyword extraction identifies the most imoprtant concepts in a text by identifying terms that best describe the content and subject of a document (Beliga (2014)).

SALOMON was later developed and started to move towards abstractive text summarization. The main manner in which they do this is by utilizing abstracting. Abstracting is a form of capturing information. A significant part of the structure of SALOMON is that it identifies significant paragraphs by representing each of them as a vector. The paragraphs are then grouped on similarity. They are compared with each other by calculating the cosine between the angles of the vectors.(Uyttendaele et al. (1998))

In the paper, we discuss and build upon previous research upon text summarization in the legal industry in order to fine-tune the optimized algorithm to provide accurate and precise summaries of legal documents.

Transformer Model	Description	Abstractive or Extractive
T5	A universal Transformer architecture that formulates all tasks in a text-to-text framework.	Abstractive
GPT-2	It is trained as an auto-regressive language model and functions by taking word vectors and estimating probabilities for next words as outputs.	Abstractive
BERT	Implements masked language modeling to interpret certain phrases in the context of surrounding sentences.	Abstractive
TextRank	Uses part of speech tagging to rank words on how frequent they have been used. Using this ranking, it is able to form summaries.	Extractive
PEGASUS	It uses a pretraining objective to predict masked sentences in multi-sentence texts.	Abstractive

Table 1. The table provides descriptions for the transformer models that we analyze throughout the paper. The third column indicates whether each of the models is abstractive or extractive which is crucial in the 4 when identifying which model we will fine-tune.

2.2. *Evaluative Metric*

(Owczarzak et al. (2012)) There are many evaluative metrics available (Steinberger et al. (2009)) that can compare model generated text with reference text. Therefore, these metrics can be used to compare and deduce how efficient text summarization models are at summarizing text.

Examples of the metrics are (Fabbri et al. (2021)) ROUGE, ROUGE-WE, S3, BertScore, MoverScore, Sentence Mover’s Similarity (SMS), SummaQA, BLANC, SUPERT, BLEU, CHRF, METEOR and CIDEr. Many of these models are relatively new and extensions on pre-existing models. In the case of our research paper, we have used the ROUGE evaluative metric to compare the performance of the models to deduce which one is the best.

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation (Lin & Och (2004)). It is in fact, a set of metrics, not one in particular. In ROUGE-N, the N refers to the number of ’N-grams’, which is the number of continuous tokens in a piece of text, the metric evaluates. For example, ROUGE-1 analyzes the overlap of unigrams (Lin & Och (2004)) which consists of comparing on a singular word basis. Whereas, ROUGE-2 measures bigrams where it compares two consecutive words to the reference provided by the user. It can even measure trigrams and higher order n-gram overlaps.

Along with the ROUGE-N metric, values for recall, precision, or f1 scores can also be obtained.

Recall measures the number of N-grams which are found in both the summary formulated by the chosen model and the provided reference. The equation for calculating the recall is:

$$r = \frac{count_{match}(gram_n)}{count(gram_{nreference})} \tag{1}$$

In the equation, $count_{match}(gram_n)$ is the number of matching n-grams found between the model and the reference. $count(gram_{nreference})$ is the total n-grams in the reference.

ROUGE can also measure the precision of summarized text compared to the inputted reference. While recall measures whether information is being captured, precision ensures that the information that is being tokenized is relevant. The equation for the precision is:

$$p = \frac{count_{match}(gram_n)}{count(gram_{nmodel})} \tag{2}$$

In the equation, $count_{match}(gram_n)$ is the number of matching n-grams found between the model and the reference. $count(gram_{nmodel})$ is the total n-grams in the model.

F1 scores combine the recall and precision scores to make an even more evaluative judgement on the efficiency of a model. This is the reason why we will be evaluating the performance of our models by analyzing the ROUGE F1 scores instead of the individual recall and precision scores.

$$f1 = \frac{2 * precision * recall}{precision + recall} \tag{3}$$

Apart from ROUGE-N, ROUGE-L and ROUGE-S also exist. ROUGE-L measures the longest common subsequence (LCS) in a given inputted text compared to a reference text.

3. METHOD

3.1. *Dataset*

For the contexts of the project, two separate groups of datasets were implemented. One set was used to compare the performances of the different models whilst the other was used for the fune-tuning of the T5 model.

The process of fine-tuning a model requires a lot of data processing which takes a lot of time and requires a lot of data. However, comparing the different models did not require a dataset that was extremely rich. This is the reason why we used different datasets for the different methods.

3.1.1. *Dataset for finding the most accurate summarizer model*

The dataset that we made use of to find the most accurate summarizer model was the United States Department of Justice Case summaries. The website for the website is: https://www.justice.gov/crt/case-summaries. The court cases on the website are grouped into various topics areas such as disability, national origin, race, religion and sex.

We first read the various publicly available files in the dockets of the cases such as briefs, settlement agreements, motions and decrees. We then used the chosen unsupervised models and generated summaries for each model on the PDFs of the documents. For each summary that was created, it was compared to the summaries created by the US Department of Justice by generating Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores. We generate the recall, precision and f1 for the ROUGE-1, ROUGE-2 and ROUGE-l metrics. We repeated this process for around 100 court cases from the dataset and generated means for each of the models. Using the means we were able to determine which model was the most accurate and precise at generating summaries. We then went forward with this model for the fine-tuning.

In our paper, we are specifically looking to train abstractive summarizer models to create unique summaries of legal documents.

3.1.2. *Dataset for fine-tuning the model*

For the fine-tuning of the model, we needed a large data set of court cases so that the summaries that were generated were as accurate as possible. For this, we used the special master dockets in the Supreme Court of the United States website.(STATES (2017)). The dockets contained every single file from special court cases from the Supreme Court. we split the data into 70/30 for training and testing. Additionally, we used the cases on the "Caselaw Access Project" for the fine-tuning of the model.

3.2. *Creating a UI for the model*

Using streamlit, we were able to create a user interface for the model. The interface allows the user to choose a court case file they wish to upload and generate summaries for. The user can also wish to input certain text which they want to summarize in the textbox instead of uploading a file. The user then chooses which model they wish to use to generate the summary. They are provided with the fine-tuned model created as well as other models such as T5, TextRank,GPT-2, and others. Finally, the interface computes a summary for the user based on the options they have selected.

4. RESULTS AND DISCUSSION

After applying the transformer models on 100 datasets of court cases, we calculated average ROUGE-1 F1 and ROUGE-2 F1 scores.

From 1, we know that TextRank is the only models that is not abstractive. Additionally, in our paper, we are looking to fine-tune abstractive models as opposed to extractive. Thus, when comparing the averages of the two models in 3, we can see that BERT provides the best summaries followed closely by T5 and Pegasus.

For the purposes of the paper, we have decided to fine-tune the T5 algorithm on a dataset of legal documents to create our optimized algorithm because it is much faster and easier to train compared to BERT. This is

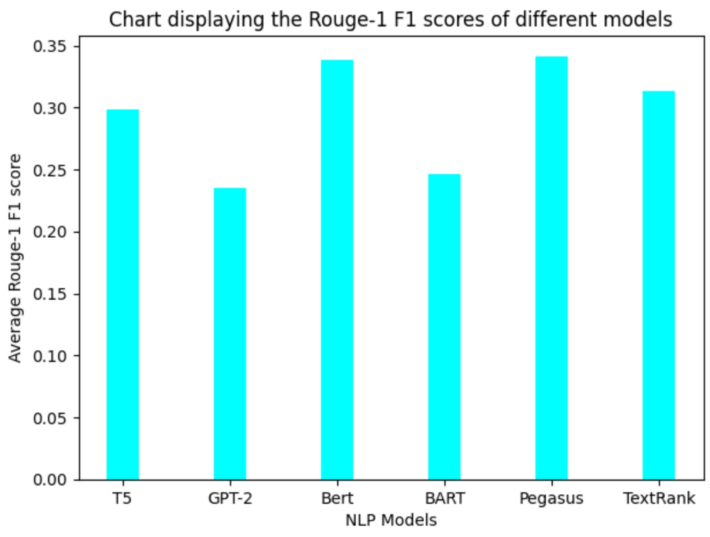

Figure 1. The bar chart displays the average ROUGE-1 F1 scores for various NLP models that we used to deduce the model we would eventually use for fine-tuning.

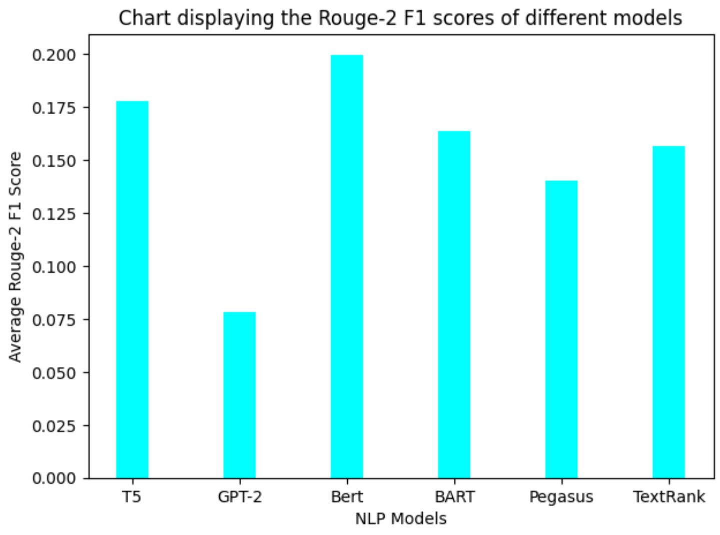

Figure 2. The bar chart displays the average ROUGE-2 F1 scores for various NLP models that we used to deduce the model we would eventually use for fine-tuning.

mainly due to the many parameters that BERT is built on.

Comparing the two bar charts from 5 and 6, we can see that the average ROUGE scores after training significantly improve.

4.1. *GitHub*

All of the code written by us to create the UI as well as training the T5 model to create the new algorithm

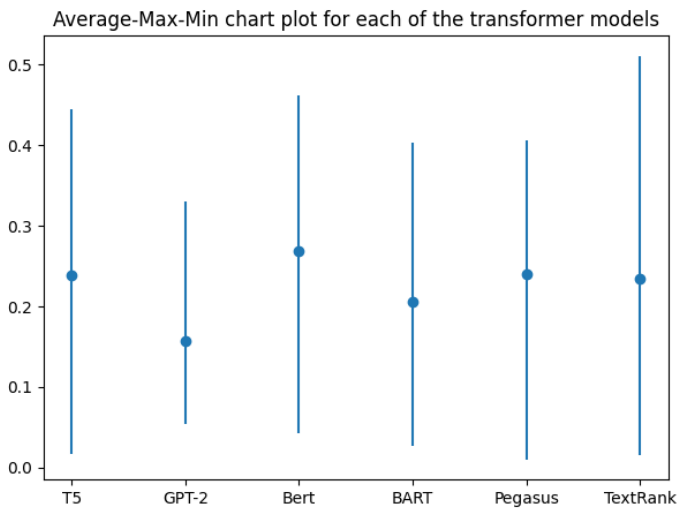

Figure 3. The bar chart displays the minimum, maximum and average for teh ROUGE-1 F1 and ROUGE-2 F1 scores for each of the models.

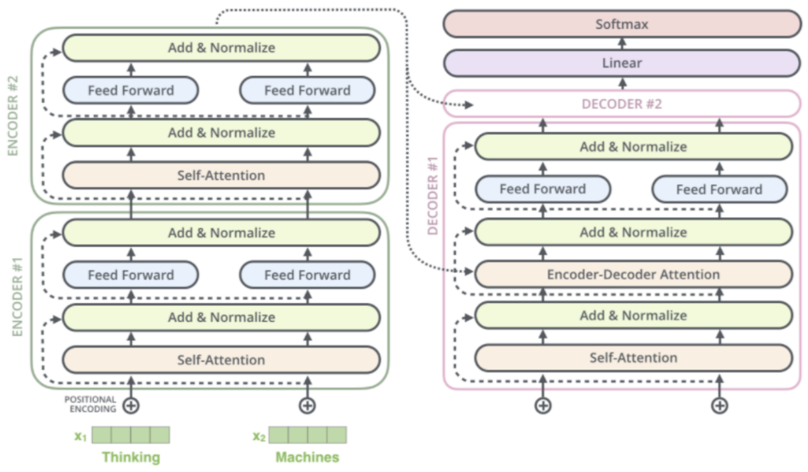

Figure 4. Detailed description of how text is summarized internally using the T5 model. (Chen (2020))

can be found on our GitHub. The link for which is: https://github.com/aman8533/Building-an-Optimized-algorithm-that-provides-summaries-of-legal-documents

5. CONCLUSIONS

In this paper, we have presented a novel algorithm that fine tunes the T5 algorithm so that it provides more accurate summaries of legal documents. Instead of the typical extract summary based results from popular models, the algorithm provides paraphrased phrases from the original text using a different vocabulary set. This provides readily available summaries of court cases to users who are in need of them. The fine-tuned model

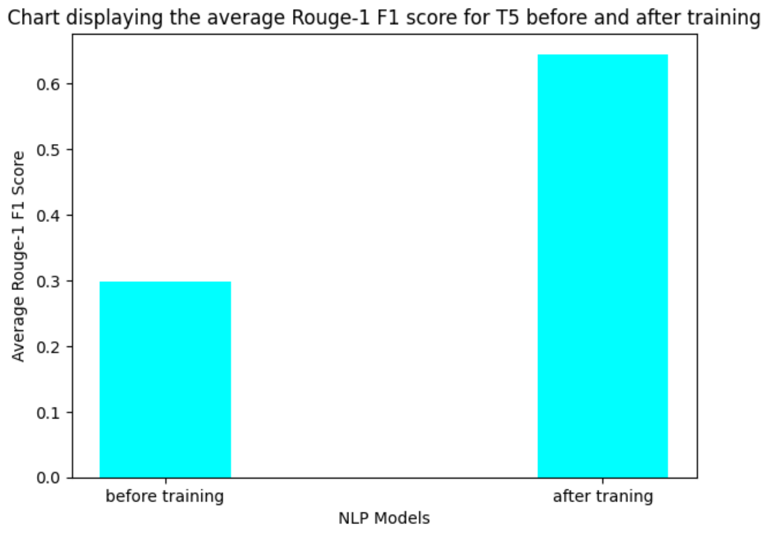

Figure 5. The bar chart displays the average ROUGE-1 F1 scores of the T5 model before and after being fine-tuned on the legal documents dataset.

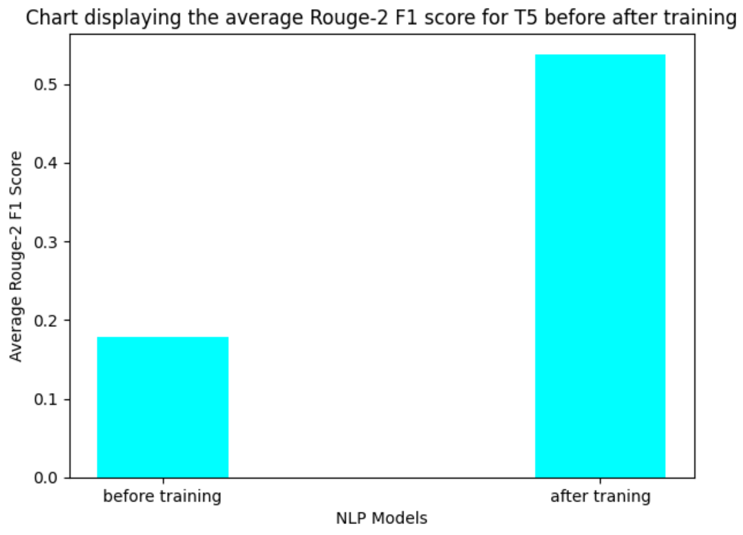

Figure 6. The bar chart displays the average ROUGE-2 F1 scores of the T5 model before and after being fine-tuned on the legal documents dataset.

did not require computationally expensive training and can easily be run using the user interface we have created to summarize arbitrary documents and text.

We obtained a bar chart in 1 with the average rouge scores for each of the models. We repeated the process of obtaining the average for the ROUGE-2 F1 scores as opposed to ROUGE-1 F1 scores in 2. We then culminated all the data in 3. Initially, we chose BERT as the algorithm we wanted to further train by comparing the ROUGE scores for other algorithms including GPT-2, T5, BART, Pegasus and TextRank. However, after digging into the algorithms deeper, we decided to progress with T5 due to the computationally intensive process of training the BERT model.

4 indicates the manner in which T5 goes about summarizing text. Different algorithms use different embeddings and methods like tokenizing to process text. T5 functions by masking the input text and uses statistical analysis to predict the masked phrases. This is similar to the way BERT functions except T5 replaces consecutive tokens with a singular Masked keyword (Turbolab (2021)).

From there, we were able to train T5 and plotted bar charts comparing the performance of the algorithm before training and after traning. As seen from 5 and 6, we can see significant improvements in the performance. We can conclude that the fine-tuning through training T5 on legal datasets made the algorithm a much better summarizing tool for legal documents.

In the future, we would also consider processing time when considering and fine-tuning the algorithm along with accuracy which we have already taken into account.

On top of that, we would like to create a functionality on the user face of the website where it will display the predicted time needed to generate summaries. Longer documents that are inputted will take more time to be processed by the system. Thus, it is important to create a distinction between how much time will be taken for particular summaries to keep users informed. This is especially important if very large files are fed in. In this case, several minutes and even hours can elapse before the system is able to produce an output. In some cases, the code may even crash due to the large amount of data that must be processed.

ACKNOWLEDGMENTS

This work was supported by Mr. Eric Bradford, Bachelor's in Computer Science from MIT, Master's in Computer Science and Electrical Engineering from MIT and Technical Product Manager at Apple on the Conversational AI team.

REFERENCES

Beliga, S. 2014, University of Rijeka, Department of Informatics, Rijeka, 1

Bhattacharya, P., Hiware, K., Rajgaria, S., et al. 2019, in European Conference on Information Retrieval, Springer, 413–428

Brownsword, R. 2008, Rights, regulation, and the technological revolution (Oxford University Press)

Chen, Q. 2020, T5: a detailed explanation, , , last accessed 7th November 2022. https://medium.com/analytics-vidhya/t5-a-detailed-explanation-a0ac9bc53e51

Fabbri, A. R., Kryściński, W., McCann, B., et al. 2021, Transactions of the Association for Computational Linguistics, 9, 391

Gelbart, D., & Smith, J. 1991, in Proceedings of the 3rd international conference on Artificial intelligence and law, 225–234

Haney, B. 2020, Brian S. Haney, Applied Natural Language Processing for Law Practice

Kore, R. C., Ray, P., Lade, P., & Nerurkar, A. 2020, Int. J. Eng. Comput. Sci, 9, 25039

Lin, C.-Y., & Och, F. 2004, in Ntcir workshop

Maltz, E. M. 2000, Hous. L. REv., 37, 1395

Moratanch, N., & Chitrakala, S. 2016, in 2016 International Conference on Circuit, power and computing technologies (ICCPCT), IEEE, 1–7

of the United States, S. C. 2021, Opinions of the Court - 2021, , . https://www.supremecourt.gov/opinions/slipopinion/21

Owczarzak, K., Conroy, J., Dang, H. T., & Nenkova, A. 2012, in Proceedings of workshop on evaluation metrics and system comparison for automatic summarization, 1–9

Pilault, J., Li, R., Subramanian, S., & Pal, C. 2020, in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 9308–9319

Polsley, S., Jhunjhunwala, P., & Huang, R. 2016, in Proceedings of COLING 2016, the 26th international conference on Computational Linguistics: System Demonstrations, 258–262

Roitblat, H. L., Kershaw, A., & Oot, P. 2010, Journal of the American Society for Information Science and Technology, 61, 70

STATES, S. C. O. T. U. 2017, Special Master Dockets, , , last accessed 1st September 2022. https://www.supremecourt.gov/SpecMastRpt/SpecMastRptWebSites.aspx

Steinberger, J., et al. 2009, Computing and Informatics, 28, 251

Turbolab. 2021, Abstractive Summarization Using Google's T5, , , last accessed 7th November 2022. https://turbolab.in/abstractive-summarization-using-googles-t5/

Uyttendaele, C., Moens, M.-F., & Dumortier, J. 1998, Artificial Intelligence and Law, 6, 59

Yarlagadda, R. T. 2017, International Journal of Creative Research Thoughts (IJCRT), ISSN, 2320

Zheng, Z., Xie, S., Dai, H.-N., et al. 2020, Future Generation Computer Systems, 105, 475