Automated News Summarization with BERT-powered encoders

Dang Ngoc Huy
h.dang@mpp.hertie-school.org

Joshua Aje
j.aje@mpp.hertie-school.org

Ole Teutloff
o.teutloff@mpp.hertie-school.org

Abstract

In this paper, we adapt the BERT powered text summarization model developed by Yang Liu and Mirella Lapata [5] and apply it to produce summaries for the large and diverse Cornell Newsroom dataset. We implement an extractive, abstractive and mixed model architecture. Despite training only on a small subset of the training data (7%), the BERT powered models achieve results close to the top implementations on the Cornell Newsroom leaderboard. This confirms that BERT powered models are a promising approach to automated text summarization that might - given enough training and fine-tuning - rival existing solutions[1].

1. Introduction

Text summarization describes the practice of producing a succinct and accurate summary of text by focusing on the essential information and meanings. In today's digital age, data is being created, collected and ingested at an unprecedented rate, with increasing variety and scope. Researchers, policy makers and citizens alike are bombarded with large amounts of text information daily, most of which is noise that conceals the vital signal. Manual summarization, however, is a costly undertaking with a built-in time lag. The development of automated text summarization systems that reduce cost and time demands while maintaining a high quality, become increasingly popular. Not only does automated text summarization help to reduce the complexity of the document in question, it also weeds out the unnecessary and redundant information, decreasing the time spent on search and reading. As a result, such summarization systems can help to process and understand large amounts of text data faster; ultimately increasing productivity. The recent progress in Machine Learning, Deep Learning and Natural Language Processing has provided an assortment of novel techniques for automatic text summarization. The current approaches rely on two methods: extractive and abstractive summarization. Extractive summarization pulls sentences and words directly from the original document to represent its central meaning. By weighing the similarities, importance, and relevance of different sentences and segments of words, the method is able to produce a shortened representation of the text in question by joining words and sentences of significance together. Abstractive summarization, on the other hand, seeks to generate words and sentences based on the contextual and semantic understanding and meaning of the text in question. Here, the goal is not to extract, but to paraphrase, and produce new text material that can accurately reflect the content of the original document, even with new words and sentences that did not appear previously.

Bidirectional Encoder Representations from Transformers (BERT), a contextual pre-training method for language representations, represents a state-of-the-art neural network architecture that can outperform all others in 11 complex NLP tasks at the time of its creation (introduced in 2018 by [2]). Liu and Lapata [5] demonstrate that BERT represents a promising approach for text summarization. In this paper, we take this research a step further by exploring the potential of utilizing BERT powered text summarization on a large and diverse dataset. To this end, we adapt the model developed by Yang Liu and Mirella Lapata [5] and apply it to produce summaries for the Cornell Newsroom dataset [2]. We integrate our text summarization model into a dashboard framework that allows users to obtain a summary of a copied text. Moreover, we build we scrapers to obtain up to date newspaper articles from a selection of outlets as input for our models.

We find that BERT powered models are a promising method for automated text summarization. Due to computational

[1] The code for this project can be found here: Project Github Repo

[2] Available here: https://summari.es/

limitations we train the summarization models only on a small subset of the data. Nevertheless, the BERT powered models achieve results close to the existing state-of-the-art solutions.

In the next section, we provide a review of the current state-of-the-art in the discipline of text summarization using Machine Learning. Subsequently, the methodology section explains the implementation of the model architecture used in this paper (text summarization based on BERT). Then we provide details on our experiments on the Cornell Newsroom dataset before presenting our results in comparison to the Cornell Newsroom leaderboard. Lastly, we analyze the results, discussing achievements and limitations of our approach.

2. Related Work

The purpose of this project is to adapt the research conducted by Yang Liu and Mirella Lapata [5], applying BERT in the context of obtaining accurate news summaries for the Cornell Newsroom dataset.

Building on the success of Transfer Learning, BERT was a sensational innovation introduced by the Google AI Language team in 2018 in the seminal paper from Devlin, Chang, Lee and Toutanova [2]. BERT's key technical achievement is the application of the bidirectional training of Transformers, an attention mechanism that learns contextual relations between words in a text. Whereas previous methods scrutinized word sequence directionally from left to right, right to left or a combination of both, the Transformer encoder architecture of BERT looks at the entirety of the word sequence simultaneously. As a result the model is able to learn the context and relationships of a word based on all of its surrounding elements, not just exclusively from the left or right direction. This mechanism allows BERT to grasp a deeper understanding of the context of the text in question as well as the connective tissues of its genetic makeup of words, sub-words and sentences. With hyperparameter fine-tuning, BERT is able to perform a variety of NLP tasks using its pre-trained neural network to create word embeddings as features for modeling.

Automated text summarization is one of these tasks. Liu et al. [5] constructed a pre-trained encoder using fine-tuned BERT which has the capability to capture a succinct representation of the text to summarize. Their models employ approaches of extractive, abstractive and a mixture of both to help create better summaries. For the extractive model, in order to represent different sentences in the document, external tokens are enclosed at the beginning of each sentence while a specialized symbol is adopted to collect features of the preceding sentence. The pre-trained encoder is stacked with multiple Transformer layers, each of which represent distinct document-level features such as adjacent sentences or multi-sentence discourse. The abstractive model follows an encoder-decoder architecture, using the pre-trained encoder with randomly-initialized Transformer decoder. The final model combines both extractive and abstractive approaches, utilizing two stage progression with the encoder being fine-tuned twice through extractive and abstractive summarization tasks respectively [5].

To evaluate their models, three different news dataset from CNN/DailyMail news highlights, the New York Times Annotated Corpus, and XSum were used since each represents a different style of text summary from long format to brief one sentence summaries. For each dataset, Liu et al. [5] were able to demonstrate that the models accomplish state-of-the-art results for both extractive and abstractive purposes.

For our research, we apply the methodology employed by Liu et al. [5] to investigate the extent to which the BERT model is applicable for a different, more diverse dataset from Cornell Newsroom, which comprises over 1.3 million articles from a variety of news outlets, focusing on different issues and styles of writing.

The leading approaches on this dataset employ a variety of approaches.

The leading extractive was developed by Grusky et al. [3], the creators of the Cornell Newsroom dataset. The leading purely abstractive model employs an attention-based sequence-to-sequence method [9]. The leading mixed approach and the state-of-the-art result for this particular dataset is currently achieved by Shi, Wang and Reddy [10]. Their paper outlines the design and implementation of the LeafNATS toolkit, a learning framework based on Neural abstractive text summarization with sequence-to-sequence model (NATS). In the NATS paradigm, the model is trained on a large amount of data to learn the connection between the feature text and the target summary with an attention-based encoder-decoder mechanism, using sequence-to-sequence framework modeled through a Recurrent Neural Network. After model training and recording of the weights and parameters, the decoder can output a summary as a sequence of words, utilizing the Beam Search algorithm. Different from Greedy Search which only chooses one best output results for each individual time step, the Beam Search algorithm opts for multiple choices for an output sequence at each time step based on conditional probability, given the input sequence, and each subsequent choice of words is paired with the previous choices, ultimately producing multiple output sentences with vary-

ing degrees of probabilities.

Each of the methods introduced above has strengths and weaknesses. Therefore, it would be beneficial to learn how their performance compares applied to the same text data. Our research seeks to answer this question by implementing the BERT pre-trained encoder for the Cornell Newsroom dataset and to understand how it fares in comparison to the NATS approach and other leading methods.

3. Method

This paper implements the text summarization model developed by [5] to test it on the larger and more diverse "Cornell Newsroom" dataset. Liu et al. [5] make use of the popular BERT architecture introduced by Devlin et al. [2] adapting it to the particular requirements of the text summarization task. In the following, we introduce the details of the model architecture as well as the necessary preprocessing steps.

BERT: Bidirectional Encoder Representations from Transformers (BERT, [2]) is a way of modeling language which creates representations of text based on masked language modeling and a "next sentence prediction" task. The left part of Figure 1 (illustration from [5]) shows the architecture of the original BERT model. As we can see, [CLS] (classification token) is added to the input to mark the beginning of the text. The end of each sentence is marked by [SEP] token. The text is represented as a sequence of tokens, each token containing three types of embeddings: token embeddings which captures the meaning of the token, segmentation embeddings which differentiate between sentences, and position embeddings which contains the position of the token in the overall text sequence to give the model a sense of the order of the words. The three embeddings are aggregated into one input vector which is then passed into bidirectional Transformer with multiple layers. The output from the stacked transformers is an output vector for each token which contains rich contextual information, and can be fed as inputs in other models for various language tasks.

BERTSUM model architecture: Although BERT has been applied to a variety of NLP tasks, text summarization requires several adaptations. Creating a summary of a text requires a language understanding beyond single words and sentences. However, the masked-language training leads to output vectors related to individual tokens instead of sentences. Additionally, the segmentation embeddings of BERT represent different sentences but they refer to sentence-pairs. Segment embeddings for text summarization task requires distinguishing between multi-sentence inputs [5]. The BERTSUM model developed by [5] introduces a document-level encoder that is capable of encoding the whole document obtaining representations for the individual sentences. As we can see in Figure 1 (right), the original BERT is modified by prepending [CLS] to each sentence. Each [CLS] collects features for the preceding sentence. Moreover, BERTSUM modifies BERT's segment embeddings for multiple sentences by applying interval segment embeddings. It achieves this by attributing embeddings to the sentences depending on whether they are even or odd sentences (see right side of Figure 1). These embeddings are aggregated and passed through stacked transformer layers generating output vectors. The output of BERTSUM is a vector t_i that corresponds to the *i*-th [CLS] and represents the *i*-th sentence. In other words, BERTSUM generates contextual embeddings representing sentences.

BERTSUM - Extractive Model: The extractive model is created by stacking several summarization-specific layers on top of the BERTSUM encoder. Different types of summarization layers can exist within BERT — including the Simple Classifier, Inter-Sentence Transformer layers, and Long Short Term Memory (LSTM) layers — and are jointly fine-tuned with BERT. These layers have their own specifications (for details on the different summarization layers see [5] page 2). For the extractive model, the inter-sentence layers are applied over the BERTSUM outputs to learn document level features which can then be used to create summaries. More precisely, the model estimates a score for each sentence denoting how representative the sentence is of the entire document [5]. The task of extractive summarization entails the assignment of a label $y \epsilon \{0, 1\}$ to each sentence based on the final predicted score obtained from the model. Sentences with label 1 embody the core text content and are included in the summary.

BERTSUM - Abstractive Model: The abstractive model follows the standard encoder-decoder model architecture. In this case, the pre-trained BERTSUM serves as the encoder and generates, as described previously, encoded input text which is fed to the decoder. The decoder is a 6-layered Transformer initialized randomly, and trained from the scratch to learn to decode the encoder output into an output sequence. Training the decoder from scratch raises the possibility of a mismatch between the encoder and the decoder. To deal with this mismatch two separate optimizers are employed with different learning rates. Abstractive summaries are generated one word at a time. At every timestep of the decoding process, the decoder predicts the next

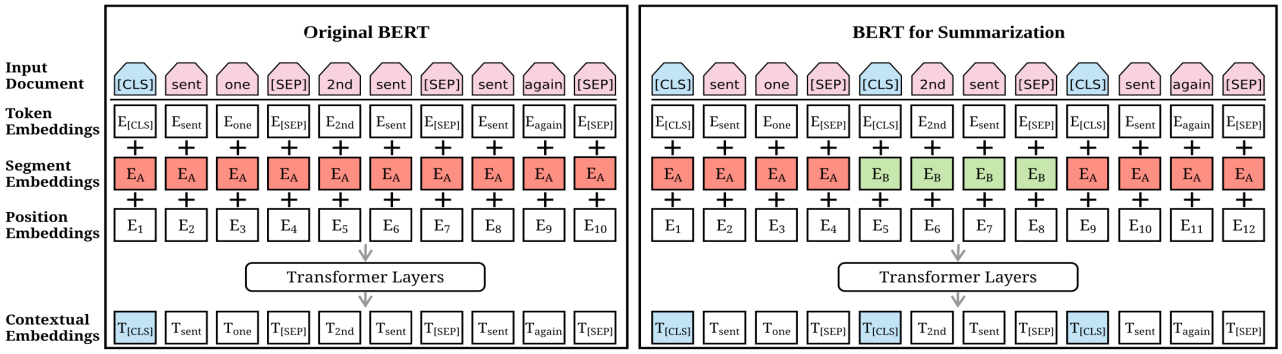

Figure 1. Architecture of the original BERT (left) and BERTSUM (right). Illustration and description from [5] page 3: "The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences."

word using self-attention and attention with respect to the encoder state (source-to-target attention). More specifically, it uses the summary generated so far y_t with positional encodings to generate a self-attended representation of the decoder state s_t. Attending to the encoder state, the representation s_t and the input text encodings are used to compute a final representation of the decoder state d_t. A probability distribution for the next word P_{vocab} over the vocabulary is then generated by passing d_t through a linear layer and a softmax layer. The word corresponding to the vector cell with the highest probability is produced as the output for this time-step.

BERTSUM - Mixed Model: The final model combines both extractive and abstractive approaches, utilizing two stage progression with the encoder being fine-tuned twice through extractive and abstractive summarization tasks respectively.

Preprocessing: We follow the preprocessing steps of [5] in transforming our data into working input for the model, namely: sentence splitting and tokenization, conversion to simpler json files, and conversion to required PyTorch format. However, since we are working with a different dataset, we prepend an initial data mining procedure to the three steps highlighted above.

In the first step, we extract the data from their original (JSON Lines) format into individual text files (with each file containing both the article and corresponding summary text). To differentiate summary text from the article text, we prepend the character <s> to each sentence of the summary text. Next, we use the Stanford CoreNLP Pipeline to tokenize the texts and split the tokenized texts into sentences. This tokenization process outputs the results as json files, containing sentences, along with their tokens and other superfluous token meta-data. Consequently, we simplify each generated json file by extracting the tokens, converting them to lower case and returning them in a dictionary with keys "scr" (value: sentence-split tokenized article text) "tgt" (value: sentence-split tokenized summary text). This step is conducted using parallel preprocessing.

BERT is a pre-trained model that expects input data in a specific format, hence in the final preprocessing step we generate features that will serve as input to the model. First, we insert special tokens to mark the beginning ([CLS]) and separation/end of sentences ([SEP]). Then, using the BERT tokenizer, we wordpiece-tokenize the article and summary text, after which we generate token IDs. Next, we generate the segment IDs, which specifies which sentence each token in a tokenized text belongs to. Finally, we convert our data to Pytorch sensors required by BERT PyTorch.

Newspaper scraper: In order to collect up-to-date newspaper article on which to test our final summarization model, we implement three automatic data collection approaches. For newspaper outlets that offer an API to access their articles content we implement a API-request approach. This is the case for the Guardian. Our custom-made function for the Guardian API collects the five most up-to-date articles on the Guardian website. However, most large newspaper outlets (such as for example New York Times, Washington Post or Fox News) do not provide users with access to the full article text via their respective API. For

Fox News we built a web scraper using the BeautifulSoup package [8]. For New York Times, BBC and the Daily Telegraph we built a web scraper using the Newspaper3k package [6]. In both cases we collect the five most up-to-date articles on the respective website. These scrapers will be implemented in future update of the web and mobile applications as a built-in functionality.

Dashboard and mobile application: To demonstrate a real-world use case, we implement a web application with the Flask framework. We use HTML, Javascript and a Jquery front-end.

4. Experiments

Data: The objective of this project is to test the approach by [5] on the larger and more diverse "Cornell Newsroom" dataset. Introduced by [**?**], this dataset consists of 1.3 million articles and corresponding summaries from 38 major publications written between 1998 and 2017, ranging in different styles, subject matters and targeted audience. The diversity ranges from the major US newspapers (such as Washington Post, New York Times, Los Angeles Times), entertainment and sports (for example Fox Sports, The Post Game, TMZ), Financial Writing (The Economist, Forbes, Bloomberg), digital media (AOL, Buzz Feed), television networks (CNN, BBC, Fox News) to international outlets (among others Aljazeera, Reuters, The Guardian). The dataset is open-source and available on Summari.es.

Evaluation method: Automatic text summarization models are commonly evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE scores) [4]. ROUGE scores capture how an automatically generated text summary compares to a gold standard summary (often human-generated). ROUGE scores are calculated by counting the overlapping units (n-grams, word pairs or word sequences) between the automatic summaries and the ones created by humans. Several types of ROUGE scores exist. R-1, R-2 and R-L are commonly used metrics [10, 5]. R-1 and R-2 stand for the overlap of unigrams and bigrams. R-L represents the longest common subsequence that overlaps [4]. Most researchers in the field of automatic text summarization use these three scores. Moreover, the leaderboard of Summari.es, which contains the Cornell Newsroom dataset and all information on efforts to provide accurate automatic summarization for this data, is based on R-1, R-2 and R-L. Therefore, we focus on these three ROUGE scores to evaluate the performance of our model.

ROUGE-score range from 0 to 1. In most cases, a ROUGE score of 1 is impossible since an extractive system cannot reproduce the exact summaries made by human. Therefore, we need to identify the upper bound summary in terms of ROUGE score performance that can realistically be achieved by the system so as to know the limit for improvement. This is known as the oracle summary. Similar to [5], a greedy algorithm was employed to recursively identify and select the sentences that reach maximum ROUGE-2 score as the oracle sentences. These sentences were then labelled as 1, and 0 for others. The summaries generated by the model will then be evaluated against these oracle sentences to measure how far the model is from the upper bound limit of automatic summarization capability.

Moreover, we employ precision and recall. Recall for ROUGE score pinpoints the discrepancy between the generated summary and the reference summary. For individual words this corresponds to the number of overlapping words divided by the total number of words in the reference summary. However, the model could produce a long sentence containing all words of the reference summary without it not necessarily being a good summary. Therefore, we use precision to record how relevant the model summary. Precision is calculated as the number of overlapping words divided by the total number of words in the model summary. These two metrics constitute the F-measure that makes up the principal metric for ROUGE scores.

However, calculating ROUGE scores is computationally expensive. Therefore, following the approach of [5], we use cross-entropy to evaluate the training progress of the model. Representing the difference between two probability distributions, cross-entropy is commonly used as a loss function in machine learning. Figure 2 shows the training progress of our three models. Upon completion of the training process, we use ROUGE scores to check the performance of the final model on the test set. Table 1 displays these results in comparison to the Cornell Newsroom leaderboard.

Experimental details: We train one model each for the extractive, abstractive and a mixed approach to determine which has the better capability to capture and summarize the meaning of the text in question. The extractive model was executed for 50,000 training steps (each step is a gradient update) on 1 GPU available with the Google Colaboratory platform (Colab) with gradient accumulation every two steps. The Adam optimizer is utilized with β_1=0.9, and β_2=0.999. We fit the model on a subset of 100,000 articles from the 1.3 millions sample, training on 60,000 articles, validating on 20,000 articles and reserving 20,000 articles as an unseen test set. Due to computational limitation and

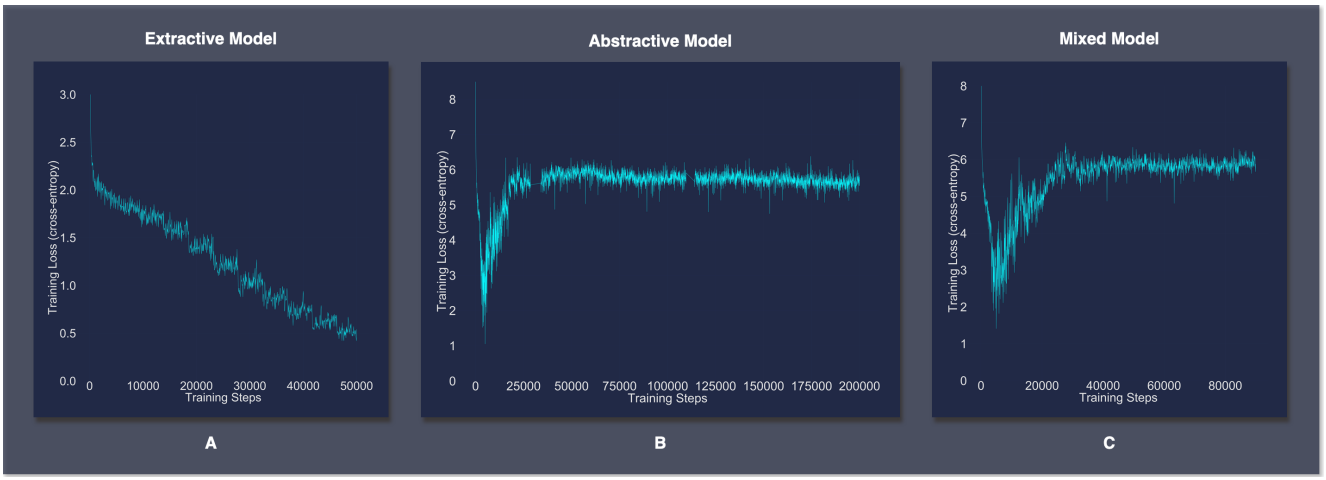

Figure 2. Training loss decay of extractive model (A), abstractive model (B) and mixed model (C).

running time constraints of the Colab GPU, this was the optimal amount of samples that our research could implement, after some experimentation.

To reduce redundancy and increase the diversity of the generated summaries, Trigram Blocking was utilized to select sentences, preventing overlapping of triagrams [7]. The objective is minimizing the similarity between the sentence being selected for the model as the output and sentences which have been already selected as part of the standard summary, which is equivalent to the approach taken in Maximal Marginal Relevance (MMR; [1])

Model checkpoints are collected and evaluated with the validation dataset for every 1,000 steps to prevent loss of training due to the regular GPU usage timeouts of Colab. Moreover, the checkpoints serve as potential candidates for the best model depending on the performance on the validation set. Figure 2 (A) displays the training progress of the extractive model. As we can see, the training loss (cross-entropy) decreases steadily reaching a minimum at 50,000 steps. However, the training progress is not a smooth gradual drop. Instead, there are multiple spikes, meaning that the loss does not decrease on every iterations. This particular behavior is a trademark consequence of using the Adam Optimizer with Mini-Batch Gradient Descent. Some mini-batches are, by pure random chances, harder than others to train and validate for and thus inducing the noisy spikes in the cost function when using Adam. Therefore, to identify the best model weights, we evaluated the checkpoints surrounding the last training steps.

The abstractive model, on the other hand, has a encoder-decoder structure. There are 768 hidden units in the Transformer decoder and 2,048 hidden size for the feed-forward layers. The model was executed for 200,000 steps on 60,000 articles, similar to the extractive model, with 1 GPU available on Google Colab. The gradients are collected on every succession of five training steps, and checkpoints with model weights are saved and evaluated on the validation articles every 2000 steps. The decoding process of the model employed the Beam Search algorithm to generate more variety in terms of output sentences, each of which has a certain degrees of probabilities of being chosen as the candidate sentence for inference. The decoding work is finished when the end-of-sequence token is encountered. Figure 2 (B) shows the training loss decay of the abstractive model. In contrast to the extractive model, the loss does not decrease steadily. Instead, we observe a sharp decline at the start, followed by a slow increase before we reach a plateau (from about 30,000 steps onwards). Measured by cross-entropy, the model seems to not improve further after reaching the plateau. This might be due to a number of reasons, one of which is the high learning rate, leading to the model moving too much in the direction opposite to the gradient, resulting in distance further away from the local minima and thus increasing the loss. Another possibility could be that the extractive and abstractive summarization differ quite considerably in their internal components. It might be that common evaluation measures (cross-entropy and subsequent also ROUGE scores) are of limited utility when it comes to evaluating the quality of abstractive summaries.

Lastly, the mixed model combines the outputs of the two previous models by loading in the weights of the final extractive model and then training on the dataset using the abstractive approach. Thereby, the model should capture the summarization capabilities of both, the extractive and abstractive, methods. The mixed model was trained for only 90,000 steps due to stagnating training loss. In Figure 2 (C) we can see the training progress of the mixed model. As expected, it looks very similar to the training plot of the

abstractive model.

The main challenge for the experiments is the computational limitation of working with Google Colab. Due to the constraints of the platform, GPU usage cannot exceed 12 hours of continuous model training, leading to sudden disruption of work, pending a blackout penalty of 24 hours. GPU usage also encounters timeout without warning, after which the computational capacity is downgraded. A a result, our research had to limit the upper bound of the number of training samples we could process for the project. We settled on 60,000 news article for our main training (about 7% of the complete training set) as that can provide us with sufficient samples while also permitting for a full cycle training, with tuning and debugging in between to make sure the training went by smoothly. The final fit of the extractive model on this 60,000 articles took 3 days, whereas the abstractive and mixed models took more than 1 week of intermittent training each to finish [3]. This allowed for only minimal hyperparameter tuning, particularly for identifying the right learning rate that could help the models to achieve optimal results.

5. Results and Analysis

Table 1 displays the results in comparison to the respective state-of-the start method from the Cornell Newsroom leaderboard. Despite being trained on only 60,000 articles, the BERTSUM model achieves scores close to the existing state-of-the-art results in all ROUGE scores. Except for the extractive model, we validated the models on a reduced test set of 20,000 articles (proportional to our training set)[4].

In particular, the extractive BERTSUM model achieves scores close to the top leaderboard result. The leading extractive model is the Lede-3 Baseline by [3]. Our model achieves ROUGE scores (R-1, R-2 and R-L) that are about 3 percentage points lower than the Lede-3 Baseline model. The leading purely abstractive approach (Seq2Seq + Attention by [9]) performs similar to our abstractive model.The mixed BERTSUM model achieves ROUGE-scores that are about 10 points lower than the leading Pointer-Generator method by [10].

These results confirm the suitability of BERT based models for automated text summarization. Trained on only 7 % of the training data, our models perform similar to the top non-BERT implementations. It is not unlikely that the BERTSUM model, trained on the full training set, would rival the existing approaches.

Table 1. BERTSUM performance results on test set compared to top models from the Cornell Newsroom leaderboard.

System	R-1	R-2	R-L
Extractive			
Lede-3 Baseline (by [3])	32.02	21.08	29.59
BERT (full testset)	28.74	17.76	26.15
Abstractive			
Seq2Seq + Attention (by [9])	5.99	0.37	5.41
BERT Model	7.04	0.32	6.78
Mixed			
Modified P-G (by [10])	39.91	28.38	36.87
BERT Model	30.01	17.77	27.00

However, comparing numerical metrics alone, might provide us with holistic picture of how well the models can produce meaningful summaries in a real-world setting. Therefore, we test the models on real use cases which we obtain using our news article web scrapers. We display one example for illustrative purposes (see appendix). This includes an excerpt from the article *Europe had hottest year on record in 2019* by Fiona Harvey, the Guardian, April 22, 2020 [5]. The original article followed by the extratcive, abstractive and mixed summary generated by the BERTSUM model can be found in the Appendix.

As we can see, the extractive model presents the original article's first three sentences as its summary. The abstractive model, on the other hand, makes an effort to form and create new sentence structures based on the provided text, with surprisingly good-quality grammar and sentence structure. Finally, the mixed model does both, extracting sentences while also changing sentence structures, omitting certain words. All three summaries successfully capture the main idea behind the given article while maintaining a good grammatical structure. This exemplifies how far automated text summarization has already advanced and how promising BERT-based models are to further improve summarization quality.

Limitations: The results of our final models suggest that there is still considerable room for improvement. Limited computational resources represent the key limiting factor for our project. The restricted computing power forced us to drastically reduce the size of the training set to 7 % (60,000

[3]Code documentation: `https://github.com/huydang90/News-Summarization-with-BERT/tree/master/Documentation`

[4]For example, the testing time of the abstractive BERTSUM on the full test set amounted to approx. 81 hours on Colab which was impossible to complete without a professional account.

[5]Link to test case: "Europe had hottest year on record in 2019"

articles) of the original dataset. Moreover, we could not perform extensive hyperparameter optimization. Our models nevertheless come close to the performance of the top models on the Cornell Newsroom leaderboard. More computing power would have allowed us to train our models on the full training set. Moreover, we realized that the current evaluation metrics (ROUGE scores as well as cross-entropy) are not ideal to evaluate the quality of abstractive summaries.

Abstractive summaries are inherently difficult to evaluate as they can contain entirely different words and sentences compared to the original text and gold standard summary. As a result, ROUGE scores for abstractive models are generally quite low. Qualitative evaluation is a better evaluation method which is however not automatable (very limited scalability due to high cost and time intensity). Designing a better evaluation method for abstractive summarization represents a challenging but important avenue for further research.

Dashboard integration: To test how the model can be used in real-life setting, we implement a web application through the Flask framework, with HTML, Javascript and Jquery front-end. Users can input their text and select one of our three models (extractive, abstractive or mixed model) to generate the summary. The back-end engine then uses the selected model to generate the summary. As a proof of concept, we built a mobile application prototype with Dart under the Flutter framework (see Appendix), with hard-coded functionalities. The mobile app can run on both Android and iOS devices and showcases how the model could be deployed as a mass-consumer product.

6. *Conclusions

In this project, we set out to explore the potential of BERT powered text summarization on a large and diverse dataset. We adapt the model developed by Yang Liu and Mirella Lapata [5] and apply it to produce summaries for the Cornell Newsroom dataset. Despite training the models only on a small subset of the data (due to computational limitations), we achieve performance close to the state-of-the-art approaches. In particular, the extracting model gets close to the leading result. A qualitative examination of real-world test cases confirms that our models produce meaningful summaries. Moreover, we integrate our text summarization models into a web application framework that allows users to obtain a summary o text and implement mobile application UI as a proof-of-concept for how the models can work in real use case.

Despite several limitations, our research demonstrated that BERT based models represent a promising approach to automated text summarization that deserves further exploration. Most notably, the summaries generated by our models are understandable, meaningful and able to capture the main essence of the text or news article in question.

7. Contributions

All efforts in data preprocessing, experimentation, analysis, dashboard creation and progress report writing was divided equally among the group members. For the purpose of this report, main responsibilities will be shared as followed:

Joshua Aje led the pre-processing, and wrangling of the dataset to shape it to a form manageable by the model, and test-run the baseline model. Together with Ole, Joshua contributed predominantly on writing the Methodology of the final paper. Ole Teutlof was in charge of experimenting with the models, coding the news outlet scraper and work on the Methods, Experiments and Results Analysis of the written paper.

Dang Ngoc Huy was responsible for running the mixed model, working on the web application for summarizing news articles, while contributing to writing the Related Works, Experiments and Results of the paper.

8. Sharing project

The research project serves as a joint project for the courses E1296 and E1326.

References

[1] J. G. Carbonell and J. Goldstein. The use of mmr and diversity-based reranking for reodering documents and producing summaries.(1998). *Pro- ceedings of the 21st Annual International ACL SI- GIR Conference on Research and Development in Information Retrieval*, 1998.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] M. Grusky, M. Naaman, and Y. Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[4] C.-Y. Lin. Rouge: A packagefor automatic evaluation of summaries. In *ProceedingsofWorkshop on Text Summa-*

rization Branches Out, Post2Conference Workshop of ACL, 2004.

[5] Y. Liu and M. Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

[6] L. Ou-Yang. Newspaper3k: Article scraping curation. 2012.

[7] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[8] L. Richardson. Beautiful soup documentation. 2007.

[9] A. M. Rush, S. Chopra, and J. Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[10] T. Shi, P. Wang, and C. K. Reddy. LeafNATS: An open-source toolkit and live demo system for neural abstractive text summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 66–71, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

9. *Appendix

Automated Summarization models in action

Original Article

Europe had its hottest year on record last year, new data has confirmed, with periods of exceptional heat last February, June and July, and one of the wettest Novembers on record. Previous records were broken by only a small margin, but the findings confirmed that 11 out of the 12 warmest years in Europe have occurred in the past two decades, according to the European State of the Climate 2019 report, published on Wednesday. There was drought in central Europe over the summer, but at the end of 2019 four times the normal amount of rain fell in western and southern Europe. [...] The World Meteorological Organisation (WMO) also produced its final report for 2019, confirming that last year was globally the second hottest on record, with the oceans reaching the highest temperatures ever recorded.

Extractive Model Summary

Europe had its hottest year on record last year, new data has confirmed, with periods of exceptional heat last February, June and July, and one of the wettest Novembers on record. Previous records were broken by only a small margin, but the findings confirmed that 11 out of the 12 warmest years in Europe have occurred in the past two decades, according to the European State of the Climate 2019 report, published on Wednesday. There was drought in central Europe over the summer, but at the end of 2019 four times the normal amount of rain fell in western and southern Europe.

Abstractive Model Summary

Europe had its hottest year on record last year. New data has confirmed that 11 out of the 12 warmest years in Europe have occurred in the past two decades.

Mixed Model Summary

Europe had its hottest year on record last February, June and July, and one of the wettest Novembers on record. Previous records were broken. But 11 out of the 12 warmest years in Europe have occurred in the past two decades.

Table 2. Extractive BERTSUM - Full results in comparison to state-of-the-art solution

Comparison of F-measure ROUGE score			
System	R-1	R-2	R-L
Lede-3 Baseline	30.63	21.41	28.57
BERT model	32.32	21.99	29.92
Comparison of Recall ROUGE score			
Lede-3 Baseline	58.43	38.21	53.91
BERT model (15k steps)	59.07	36.08	53.70
Comparison of Precision ROUGE score			
Lede-3 Baseline	23.51	16.98	22.07
BERT model	27.35	20.16	25.72

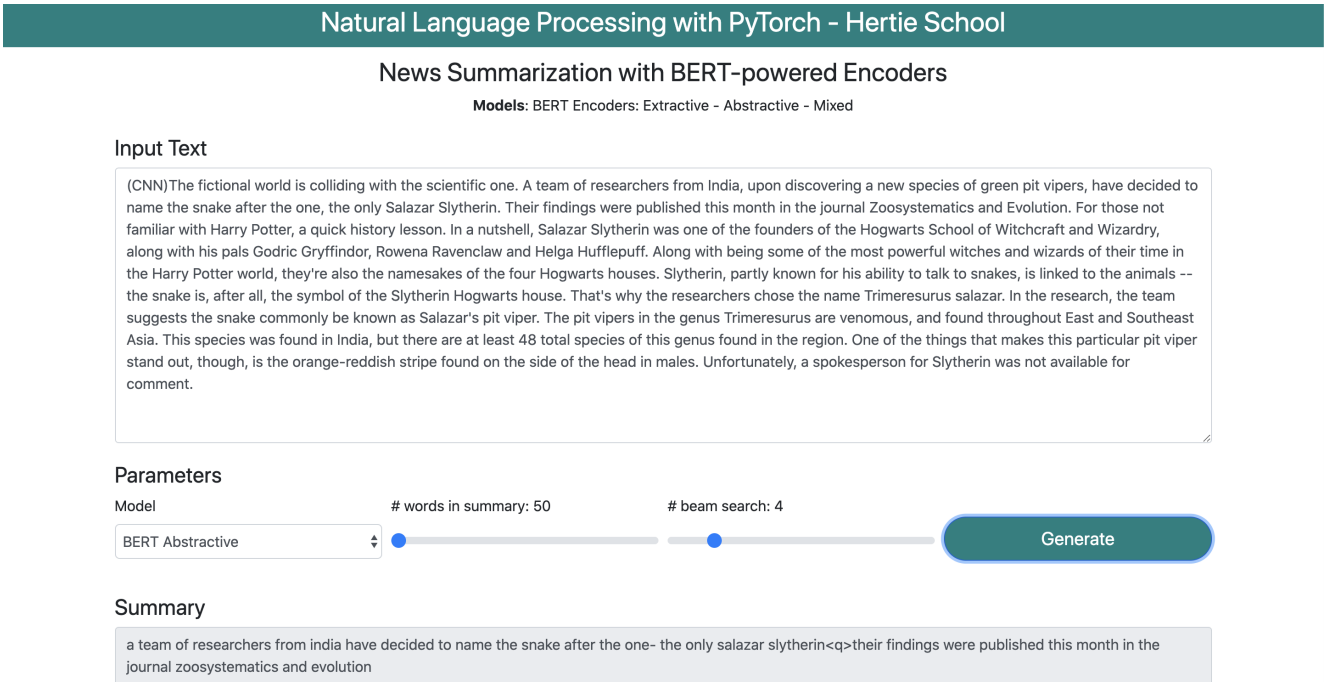

Figure 3. Snapshot of web application for automated news summarization. With the app, the users will be able to copy news articles or any piece of writings and generate summaries based on three different models: extractive (summary taken directly from text), abstractive (summary generated by encoders) or a mixture of both.

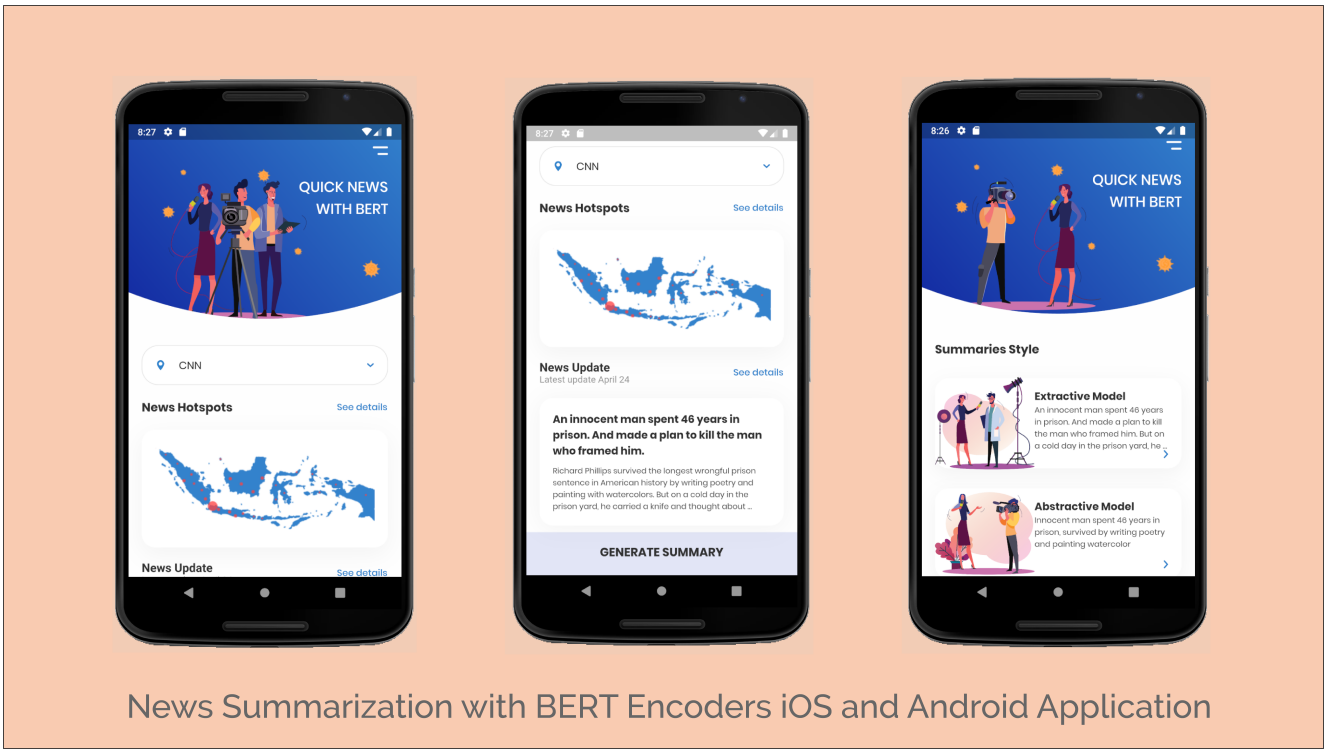

Figure 4. Snapshot of iOS and Android modbile application UI for automated news summarization. Quick News with BERT is a proof of concept for how these automated summary models can be utilized in real life setting. The user will be able to choose news sites that are of their interest and see the latest updated articles and generate their summaries quickly and efficiently.

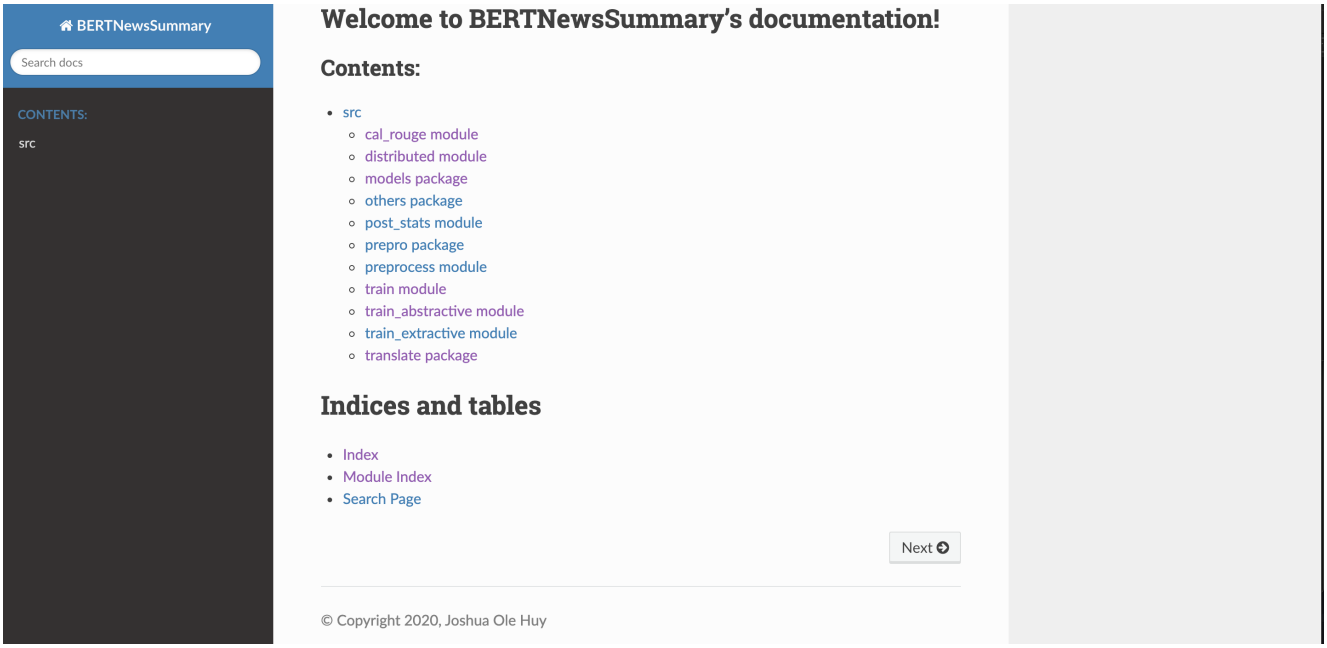

Figure 5. Snapshot of Project Code Documentation.

Table 3. Abstractive BERTSUM - Full results in comparison to state-of-the-art solution

Comparison of F-measure ROUGE score			
System	R-1	R-2	R-L
Seq2Seq + Attention Baseline	5.91	0.43	5.36
BERT model	7.04	0.34	6.78
Comparison of Recall ROUGE score			
Seq2Seq + Attention	4.89	0.38	4.44
BERT model	7.62	0.38	7.36
Comparison of Precision ROUGE score			
Seq2Seq + Attention	9.69	0.63	8.85
BERT model	8.63	0.39	8.32

Table 4. Mixed BERTSUM - Full results in comparison to state-of-the-art solution

Comparison of F-measure ROUGE score			
System	R-1	R-2	R-L
Modified P-G	39.91	28.38	36.87
BERT model	30.01	17.77	27
Comparison of Recall ROUGE score			
Modified P-G	34.31	16.35	29.41
BERT model	40.67	21.56	36.07
Comparison of Precision ROUGE score			
Modified P-G	23.72	12.65	20.64
BERT model	29.27	18.89	26.79