Abstract

Automated summarisation is the task of generating a shorter version of a document while retaining its most important information. It has recently received much attention from scholars due to the progress in Natural Language Processing methods, notably via the application of Recurrent Neural Networks or Transformers on top of pre-trained encoders in completing this task (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhou et al., 2018; Liu, 2019; Wang et al., 2019). In the case of this dissertation, an automated summariser is built using state of the art machine learning techniques to summarise documents produced by international development organisations when presenting the details of a project. Specifically, we replicate a proposed methodology (Liu, 2019) to develop BertSummaDev and NeuSummaDev, respectively a fine-tuned BERT classifier and a LSTM built on BERT outputs. In summarising development projects, both models outperform by a large margin the chosen baseline model and two well-established extractive summarisers: LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004). Interestingly, NeuSummaDev outperforms BertSummaDev with a ROUGE-1 score of 96.8 compared to 91.9 on our dataset, which goes in contradiction with Liu's findings. We deploy our extractive summariser in a shareable app, SummaDev, for real-life applications. This has the potential to enhance local populations' understanding of the development projects implemented in their regions, as well as to produce productivity gains in development organisations. Lastly, we introduce SummaDevDocs, a data-set for development paper summarisation which gathers 3K development project descriptions associated with fully extractive gold summaries.

The coding details, a link to the app and to the data, are found on this github repository.

Acknowledgement

I would like to thank both my University College London supervisor, Gazi Arif, and my sponsor supervisor, Peter Addo, for their guidance and support throughout this project.

Statement of Impact

We created a shareable application, SummaDev, that summarises development project descriptions in a few sentences.

We also introduce a data-set, SummaDevDocs, for future research around development project papers' summarisation.

All the details of the code can be found on the following github repository.

Contents

List of Figures

List of Tables

List of Equations

Abbreviations list

Abbreviation	Description
CNN	Convolutional Neural Network
BERT	Recurrent Neural Network
ELMo	Embeddings from Language Model
GloVe	Global Vectors for Word Representation
GPT	Generative Pre-trained Transformer
IATI	International Aid Transparency Initiative
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RNN	Recurrent Neural Network
SVD	Singular Value Decomposition
SVM	Support Vector Machines
Tf*idf	Term frequency-inverse document frequency

1 Introduction

Automated summarisation is the task of generating a shorter version of a document while retaining its most important information. This technique was first introduced in the 1950s (Luhn, 1958). Yet, it has recently received much attention from scholars, due to the progress in Natural Language Processing methods, notably via the application of Recurrent Neural Networks (RNN) or Transformers on top of pre-trained encoders in completing this task (Cheng and Lapata, 2016; Nallapati et al., 2017; Zhou et al., 2018; Liu, 2019, Wang et al., 2019). In fact, summarisation is a task that requires a complex level of understanding of the text as it extracts a few sentences or words while retaining the meaning of the document (Sparck Jones, 1993). Furthermore, automated summarisation is extremely applicable due to its potential for information access task applications. In the case of this dissertation, an automated summariser is built using state of the art machine learning techniques to summarise documents produced by international development organisations when presenting the details of a project. Although there already exist a various number of extractive summarisation models (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Zheng and Lapata, 2019; Zhong et al., 2020), it is expected that training a model to development papers only will provide better results than applying an existing automated summarisation model that was trained on other texts data. Indeed, development institutions have their own vocabulary, as do research papers that are written using a vocabulary which is specific to their field (Cachola et al., 2020).

We chose to implement extractive summarisation, since, although abstractive summarisation has recently sparked strong interests in research, it still performs more poorly than extractive summarisation (Widyassari et al., 2020). This shows relevance in various aspects. Firstly, this model will be used in a public application, SummaDev, for real-life applications. In fact, the tool will be available for development organisations' employees to summarise long project documents, thereby enabling notable gains in productivity. Furthermore, the tool has the potential to enhance the local populations' understanding of the development projects that are taking place in their regions. For these reasons, it seemed wiser to develop an extractive summarisation model rather than an

abstractive one.

Academically, this paper replicates two methodologies proposed by Liu (2019) to implement extractive summarisation. These methodologies were chosen since they have demonstrated to outperform any other extractive summarisation models on the CNN/Daily News data-set with regards to their Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scores. Namely, the proposed methodologies were as followed: a) adding a Bidirectional Encoder Representations from Transformers (BERT) classifier on top of BERT encoded text; b) adding a Long short-term memory (LSTM) neural network on top of BERT encoded text. In this present work, we fine tune these models to the task of summarising development project papers, that we respectively name BertSummaDev and NeuSummaDev. We then compare the performance of these models against once each-other in order to assess the consistency with Liu's findings. Furthermore, we test their performances against LexRank and TextRank, two well-established extractive summarisation models, and a baseline model to quantify the added value of these models.

Lastly, in completing this work, we constructed a data-set for the task of development papers summarisation, that we name SummaDevDocs. It gathers 3K rows of development project descriptions, associated with gold extractive summaries. This is available for any researchers who wish to build other extractive summarisation models or abstractive summarisation models.

2 Literature Review

2.1 The application of Machine Learning in International Development

The main challenge in the field of international development research resides in the lack of qualitative data provided by developing countries. However, this has been partly remediated with the advent of Machine Learning (ML) as a key technology. Therefore, researchers have increasingly investigated development problematics using ML techniques (Rubio et al., 2010; Chen and Neil, 2015; Jean et al., 2016). This section aims to give a brief overview of the recent research that has used ML related to international development. It will be divided into 4 main areas of development: social, economic, environmental, and institutional. These pillars were chosen as they have long been used by the United Nations when classifying development issues (De-Arteaga et al., 2018).

2.1.1 Social

Non-traditional data source consists of data that was not produced by an official survey institution. Namely, it can be collected through satellite imagery, social media or via mobile-phone data. These data sources are increasingly used for research in development due to their abundancy in developing countries when compared to traditional data. It is even more used in the context of researching social patterns in developing countries. For instance, Rubio et al. (2010) compare the population mobility between developing and developed countries using cell phone traces. Similarly, Wesolowski and Eagle (2010) use cell phone data to study the mobility patterns in one of the largest slums in Kenya. Other than mobility, Education represents an important facet of the social pillar. One crucial aspect of education is access to information, of which developing countries lack substantially. For this reason, Barnard et al. (2010) propose a panel of Natural Language Processing (NLP) techniques to enable their inhabitants to access information on the Internet.

2.1.2 Economic

Poverty is a fundamental issue related to Development Economics to which researchers have implemented ML techniques to progress in the fight against it. For instance, Jean et al. (2016) trained a convolutional neural network (CNN) on survey and satellite imagery data to predict both the average household consumption expenditure and asset wealth in Nigeria, Tanzania, Uganda and Malawi. Further, Wu et al. (2020) explore the links between financial development and economic growth in China, Japan, and India with the aid of an iterative classifier optimizer with adaboost as the iterative classifier.

2.1.3 Environment

Most of the research concerning the environment in developing countries deal with the natural disasters issue, which severely impacts developing countries due to inadequate infrastructure (Alcantara-Ayala, 2002) and the lack of resource to respond to catastrophe (Ferris, 2010). Using spatial data, Tehrany et al. (2014) developed a Support Vector Machine (SVM) model to identify the correlation between flood frequency and some conditioning factors in Malaysia. ML for development can also be used to improve one country's response to natural disaster ability, as Kapoor (2010) attempted to do by predicting in the Democratic Republic of Congo's seismic occurrence and influence with the changes in background phone communication activities. Similarly, Jia et al. (2017) use spatial data in attempting to predict land use by developing both a RNN and a LSTM. Lastly, research in at the intersection of environment in developing countries and ML has also focused on migration movements, notably with Ermon et al.'s (2015) work which models movement trajectories associated to environmental shocks, framing the problem as an Inverse Reinforcement Learning task.

2.1.4 Institutional

Corruption is a fundamental institutional issue in many developing countries. Researchers have therefore attempted to either understand it better or remediate it using ML techniques. In conjunction with the World Bank, Grace et al. (2016) developed an automated tool to detect the risk fraud in development banks' contracts using classification methods. Using supervised ML techniques,

Cantú and Saiegh (2011) aim to identify fraud in Argentinian's electoral process. Another consequence of poor institutional solidity is violence. With that in mind, Parvez et al. (2016) propose a prediction model to anticipate street crime in Dhaka, Bangladesh. In a similar fashion, Chen and Neill (2015) use network analysis and social media data to discover patterns of human rights violations.

2.2 Automatic Summarisation

2.2.1 Pretrained Language Models

The use of pre-trained language models has demonstrated to impressively increase NLP task models' performances (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018). A few pre-trained language models have emerged from recent literature to show state-of-the-art results. These build on word embedding techniques, firstly introduced by Bengio et al. (2003) who developed a neural probabilistic language model which learns a distributed representation for each word along with a probability function for word sequences. Before Bengio and his co-authors' work, text representation was based on n-grams models which suffered from a curse of dimensionality, in the sense that the data the n-grams model was trained on were likely to be drastically different from the test data, thereby severely affecting one model's result (Bengio et al., 2003). Researchers have thereafter built upon the work initiated by Bengio to further develop techniques for word embeddings and develop state-of-the-art language models, especially by attempting to include context of the words in their embedding. It was Collobert and Weston (2008) who pushed the work of Bengio to a highly effective and applicable pre-trained word embedding model in their paper 'A unified architecture for natural language processing', by defining a CNN architecture that can be applied across a variety of NLP tasks (including part-of-speech tagging,named-entity recognition or learning a language model). By decoupling the word vector training from the downstream training objectives, the authors paved the way to later use the full context of words for learning their representations rather than their preceding context (Pennington et al, 2014). Mikolov et al,. (2013) have made the use of state-of-the-art pre-trained embeddings more accessible and efficient by avoiding the use of dense matrix in their word2vec model. By

introducing GloVe, Global Vectors for Word Representation, Pennington and his co-authors (2014) have pointed out and remediated the lack of use of statistical information on the entire corpus, since these were only used on a window context in previously developed language model. More progress in developing performant word-embedding techniques have been made recently. Of them, Peters et al. (2019) have developed 'Embeddings from Language Model' (ELMo) representations, which are based on a feature-based approach. ELMo representations are a function of all the internal layers of a bidirectional language model trained on a large text corpus. In contrast, the Generative Pre-trained Transformer (OpenAI GPT) is trained by fine tuning all pre-trained parameters (Radford et al., 2019). As opposed to ELMo, OpenAI GPT uses the Transformer architecture instead of a Long Short-Term Memory. BERT, which stands for Bidirectional Encoder Representations from Transformers, builds on the Transformers architecture introduced by Peters et al. (2019) to further improve language modelling through its bidirectional method (Devlin et al., 2018). Its authors argue that the previously mentioned techniques limit the power of the pre-trained representations, particularly for fine-tuning approaches. In fact, contrarily to the other pre-trained models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Precisely, BERT achieves so by using a 'masked language model' pre-training objective, which aims to predict a masked word solely based on context. This is not the case for the other models, as OpenAI GPT uses a left to right architecture, which, by its architecture, foregoes context of the words on the right-hand side of a processed word. As for ELMo, although it uses a bidirectional LSTM, it simply concatenates the left to right and right to left information, meaning that the representation does not incorporate the left and right information simultaneously. In that sense, by improving the fine-tuning-based approaches with the incorporation of bidirectionality, BERT performs better in word contextualization than other pre-trained language models. It can further be fine-tuned to achieve various task specific objectives ((Devlin et al., 2018); Yang Liu, 2019). For these reasons, BERT has contributed to build state-of-the-art models, resulting in outperforming results in a wide variety of NLP tasks.

The Tansformer

OpenAI GPT and BERT both base their model architectures on the Transformer (Vaswani et al., 2017) which has proven to be extremely effective in building performant language models. Before

Vaswani et al. (2017) published their paper 'Attention is all you need', recurrent neural networks and long short-term memory were established as state-of-the-art approaches in language modelling. By building an architecture based on an attention mechanism rather than recurrence, the authors have built a model that significantly enhances the capacity to solve sequence modelling problems such as language modelling or machine translation. This rise in performance is partly due to the Transformer's model being designed entirely on self-attention to compute representations of its input and output without using sequence aligned RNNs or CNNs. Precisely, it replaces the recurrent layers in encoder-decoder architectures with multi-headed self-attention. As such, Devlin et al. (2018) have used such an architecture to build its pre-trained language model BERT. Given that it has rendered outstanding results in performing NLP tasks, researchers have started to investigate the use of this latter as an encoder to perform automatic summarisation (Liu, 2019; Wang et al., 2019).

2.2.2 Extractive Summarisation

Extractive summarisation consists of extracting content from the original document which recaps the main points of the entire document. Broadly, five main approaches have been explored to achieve extractive summarisation: statistical approaches, graph-based approaches, topic-based, latent semantic analysis approaches, and machine learning methods.

Statistical Methods

Traditional methods for extractive summarisation mostly rely on statistical methods. In such methods, sentences are extracted based on human engineered features, such as sentence length, their position or whether they contain a word that is in the title (McKeownn et al., 2002; Lin and Hovy, 2002). Some popular features have been term frequency and centrality, with the assumption that the most frequent words would be the most important ones (Luhn, 1958, Nenkova et al., 2006; Lloret and Palomar, 2009). In his work McCargar (2004) explores the potential that the term frequency or inverse document frequency (tf*idf) has as a feature in applying extractive summarisation. The fundamental idea behind the tf*idf concept is that frequent terms in a document are important if they are not so frequent in the entire collection (Lloret and Palomar, 2012). These statistical meth-

ods have the advantage of being quite simple to implement and have proven to extract summaries of good quality.

Graph Based Approaches

Graph-based ranking algorithm is another widely used method in extractive summarisation. The idea in this method is that nodes represent elements of the text, either words or sentences, and the edges between the nodes represent the relationship between these elements. Such graphs give information on the salient elements of the text. For instance, in LexRank (Erkan and Radev, 2004), the graph represents all candidate sentences, the ones that can potentially be summaries, and links the ones that have a similarity level above a certain threshold. By performing a random walk on the graph, the system finds the most central sentences. In a similar manner, TextRank (Mihalcea and Tarau, 2004), an algorithm based on PageRank (Brin and Page, 1998), proposes graph-based ranking unsupervised approaches for keyword and sentence extraction.

Latent Semantic Analysis Method

Latent Semantic Analysis (LSA), another method used in extractive summarisation, is an algebraic-statistical based method that learns and represents the meaning of words. It is applied in 3 main steps : the creation of an input matrix, the singular value decomposition (SVD) and the selection of the sentence. The input matrix informs on the weight of each word in a sentence, where rows are filled with words and the columns represent a sentence. The SVD models the relationship between words and sentences, while the sentence selection phase involves an algorithm that selects the sentence that should be used as a summary based on the SVD results. A varied palette of algorithms has been explored for the sentence selection phase within the literature (Gong and Liu, 2001; Steinberger and Jezek, 2004; Murray et al., 2005; Ozsoy et al., 2011) LSA-based algorithms present a few limitations, including that they do not perform well in creating short summaries (Ozsoy et al., 2011).

Machine Learning Approaches

Machine learning approaches proposed in the literature for automatic summarisation can either be supervised, semi-supervised or unsupervised. The first machine-learning based summarisation

methods generally involved a binary classifier (Kupiec et al., 1995), Bayesian methods (Aone et al., 1998) and Hidden Markov Models (Conroy and O'leary, 2001). Later, NetRank (Burges et al. 2005) was released, a gradient descent method using Neural Network to learn functions that are used in sentences scoring. Svore et al,. (2007) built upon this latter to create NetSum, a method that extracts summaries from newswire documents. Based on features, Support Vector Regression is used to score and extract sentences (Li et al., 2007). SVM have also been used in various extractive summarisation methods (Fuentes et al., 2007). However, the most current methodology used in extractive summarisation consists of deep neural network-based approaches. Generally, taking advantage of the ever-progressing language models, they use a neural encoder to represent text and a classifier to predict the sentence(s) that best summarises a document. For instance, Kageback (2014) uses a recursive autoencoder to derive the phrase embedding based on a binary parse tree. In a multi-document summarisation task, Yin and Pei (2015) applied Convolutional Neural Networks (CNN) to select sentences that minimise cost based on their 'prestige' and 'diverseness'. In their work, Cheng and Lapata (2016) treat summarisation as a sequence labelling task. To do so, they used a neural network as an encoder and an attention-based content extractor. Similarly, Nallapati and his co-author (2017) built on the idea of sequence labelling when they introduced SummaRuNNer, a simple recurrent neural network-based sequence classifier for extractive summarisation. The mentioned models all treat the sentence scoring and sentence selection phases as separated tasks. By introducing NEUSUM, Zhou et al. (2018) propose to jointly learn to score and select sentences based on a RNN. In doing so, the authors obtain better ROUGE scores than both SummaRuNNer and NN-SE, the models designed by Cheng and Lapata. In their paper, Narayan et al. (2018) argue that cross-entropy training is not ideal for extractive summarisation as it extracts unnecessary long and redundant sentences. They suggest remediating this with REFRESH, a model which optimises the ROUGE* evaluation metric and learns to rank sentences using a reinforcement learning objective. Yet, due to the difficulty of the task, neural models have reached a bottleneck in improving ROUGE scores (Liu, 2019). At the same time, the introduction of BERT (Devlin et al., 2018) have enabled outstanding increases in performance of various NLP tasks. However, applying BERT to automatic summarisation appeared not to be as straightforward given that it outputs vectors on the token level rather than on the sentence level. Liu (2019) modifies BERT to apply it to sentence extraction. By using BERT as both encoder and a sentence extractor, BERTSUM outperformed

every existing model on the ROUGE scores benchmark. Specifically, BERTSUM fine tunes BERT by building summarisation specific layers stacked on top of the BERT outputs.

Overall, the application of ML to automatic summarisation allows to easily test the performance of a high number of features. Yet, these approaches necessitate a large training corpus in order to obtain conclusive results (Lloret and Palomar, 2012). Although, this latter point has been mitigated with transfer learning and the use of pre-trained models, which by fine tuning requires considerably less training and therefore less data.

2.2.3 Abstractive Summarisation

In contrast to extractive summarisation, abstractive summarisation generates novel words, to create a sentence that summarises the content of the original source. This area has triggered a lot of attention from the literature recently due to its complexity and its possibility of improvement given the progress made in the field of machine learning and NLP.

With the emergence of deep learning as a viable alternative for many NLP tasks (Collobert et al., 2011) coupled with large the existence of large human summarization datasets, researchers have started considering this framework as an attractive, fully data-driven alternative to abstractive summarization. Notably, sequence-to-sequence models (Sutskever, 2014), which are deep learning-based models that map an input sequence into another output sequence, had previously shown success in achieving complex NLP tasks such as machine translation (Bahdanau et al., 2014). The task of abstractive summarisation, although close to machine translation, is more difficult. Yet, building on the sequence-to-sequence achievements, Nallapati et al. (2016) employ an encoder-decoder RNN architecture originally used in machine translation to summarisation. Building on the idea of sequence-to-sequence deep neural networks architectures, different types of models have been developed to address specific issues to the abstractive summary generation task. To avoid redundancy, coverage-based models have been developed. Precisely, See et al. (2017) have augmented the standard sequence-to-sequence attentional model, notably by using coverage to keep track of what has been summarised. With the aim of enhancing the abstractive models' ability to capture more salient words and to entail them logically, Pasunuru and Bansal (2018) suggest a reinforcement learning approach, with the introduction of new rewards that either give higher weights to the important words in the summary or to summaries that are more logically entailed. Narayan et al.,

(2018) later argue that the above models either lag behind the extractive summarisation models or showcase a small percentage of abstraction within their generate summaries. For this reason, they introduce extreme summarization, which is based entirely on convolutional neural networks, differing from the sequence-to-sequence architectures modelled by recurrent networks in the papers mentioned above. They argue that CNNs enable to represent larger context sizes and longer-range dependencies. The most state-of-the art abstractive summarisation models, however, result from research that has drastically outperformed NLP tasks, by combining pre-training with large external text corpora and Transformer based-sequence models (Liu and Lapata, 2019; Lewis et al., 2019; Chen et al., 2021). Building on the latter method, PEGASUS (Zhang et al., 2020) is a generative summarisation model that uses a pre-training objective of predicting gapped sentences. It is pre-trained on a large corpus of web and news articles.

3 Research methodology and Results

3.1 Methodology

3.1.1 Models Selection

We choose the ML based approach to complete the task of summarising development project papers. Broadly, it can be thought as a classification task, where we predict a label $y_i \in (0, 1)$ to each sentence, indicating whether the sentence should be included in the summary or not. We decide to replicate the state of the art extractive summarising models. This choice is motivated by several reasons. Firstly, since abstractive summarisation has only recently been attractive to researchers, it still performs more poorly than extractive summarisation (Allahyari et al., 2017). Hence, the choice of implementing an extractive summariser appeared to be more relevant since the summariser built for this paper aims to have a real-life application. Furthermore, as touched upon in the previous section, research in extractive summarisation has obtained exciting results by building on top of BERT outputs, whether by adding a transformer or a RNN layer to perform the task of classification (Liu, 2019). Specifically, the rational for choosing these models stems from their state-of-the art results showcased in Yang Liu 'Fine-tune BERT for Extractive Summarization' recent paper (2019). In fact, using the pallet of ROUGE scores, Liu's models outperform by a large margin the following extractive models which were previously the state-of-the art on the CNN/Dailymail dataset: REFRESH (Narayan et al., 2018) and NEUSUM (Zhou et al., 2018). Academically, we aim to assess whether we reach similar conclusions on our data using the methodology implemented by Liu (2019). Namely, using transfer-learning, both a fine-tuned stacked Long Short-Term Memory and a fine-tuned BERT with a simple classifier layer were built on top of BERT outputs and applied to summarise development documents.

Therefore, the methodology implemented in this present paper was to replicate with adaptation these models for the task of summarising development organisations' project papers with the aim of obtaining applicable results. Parallelly, the interest in choosing these two extractive summarisation models is driven by the desire to compare their performances on another set of texts, thereby contributing to research in the field of extractive summarisation. Laslty, we compare the two models' ROUGE score performance on our test data against LexRank and TextRank, two well-established extractive summarisation models. Furthermore, we use a pre-trained BertSequenceClassifier with no fine tuning as baseline model to assess the contribution of the models we fine tune. We name the stacked LSTM fine tuned model NeuSummaDev, and the BERT model fine tuned with a simple classifier BertSummaDev.

3.1.2 BertSummaDev

For this model, we make use of transfer learning by altering the pre-trained BERT model for it to perform classification, to fine-tune it by training it on our specific data for its parameters to suit our task. To do so, we use the huggingface pytorch BertForSequenceClassification pre-trained model which already contains a single linear layer on top of BERT for classification. Where the last layer is mathematically represented in the below equation, where σ represents the Sigmoid function.

$$\widehat{Y}_i = \sigma(W_o T_i + b_o) \tag{3.1}$$

As previously mentioned, BERT uses the architecture of the Transformer (Vaswani et al., 2017), which is constituted of an encoder part reading the text input, and a decoder making predictions for the task. We can see the details of the BERT input in the below figure 3.1, where each token is the result of the sum of three embeddings: token, segment and positional embeddings.

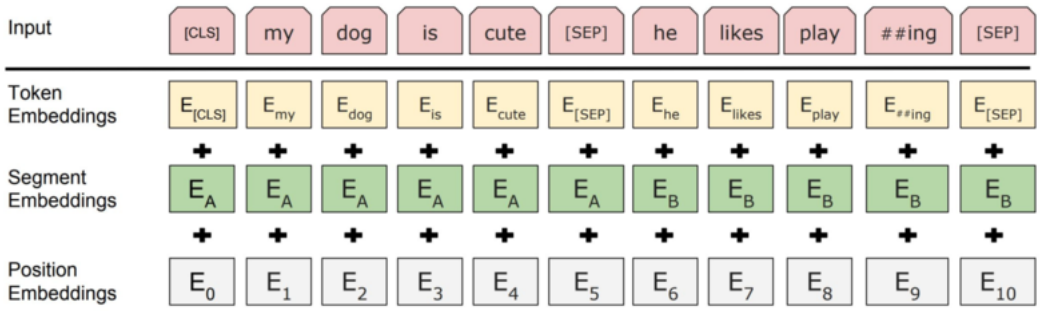

Figure 3.1: BERT input (source: Devlin et al., 2018)

BERT is then pre-trained on two unsupervised tasks, which are to predict masked tokens, and to predict the next sentence. The first task enables to obtain a true bidirectional pre-trained model which enables better context reading. The second task enables a better understanding of the relationships between the sentences. Such pre-training is effectuated on the BooksCorpus of 800 million words, and on English Wikipedia, made of 2,500 million words. Thanks to the self-attention mechanism in the Transfomer architecture, BERT can then easily be fine-tuned for many downstream tasks. We see the details for this on the below figure 3.2.

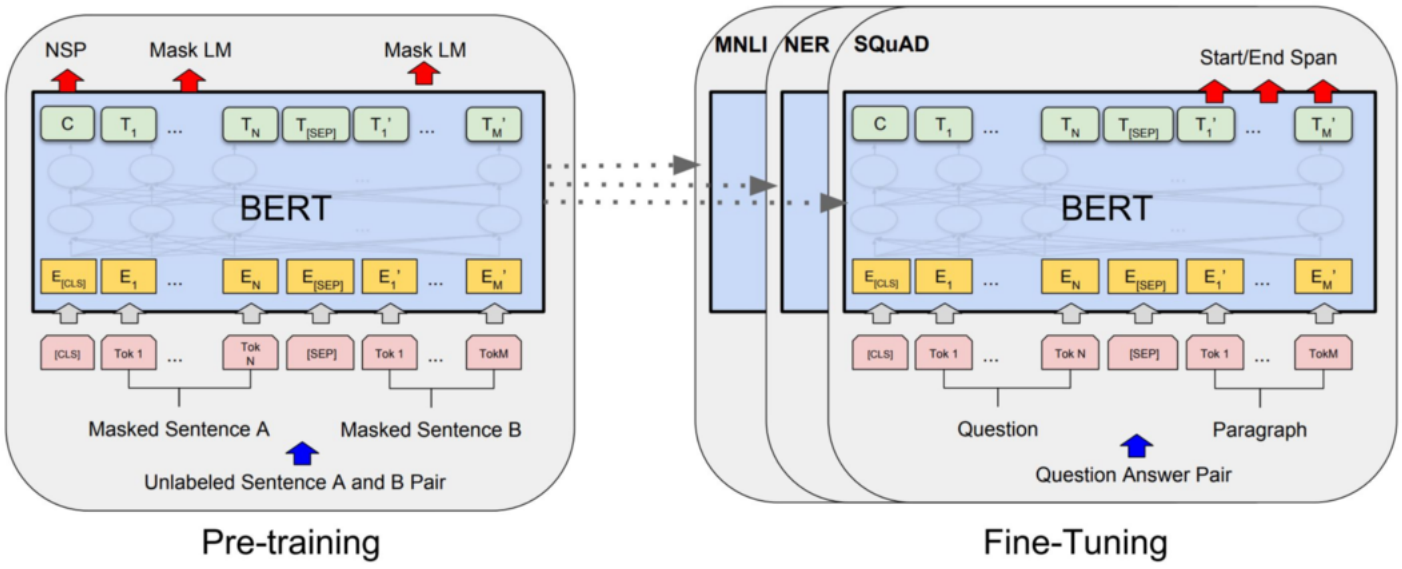

Figure 3.2: BERT fine tuning (source: Devlin et al., 2018)

3.1.3 NeuSummaDev

To develop NeuSummaDev, we stack 5 LSTM layers on top of each-others, using the keras.Sequential() model proposed by the Keras library. We further add a dense layer with a sigmoid activation. Lastly, we implement a regularisation technique by adding a Dropout Keras layer to avoid the problem of over fitting (see the model summary in figure below . .)

LSTMs have 4 important components. The forget gate, which consists of a sigmoid layer which looks at the former hidden state h_{t-1} and the current input x_t to decide whether to forget or retain the input in the previous cell state C_{t-1}. It is mathematically defined as per the below equation.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{3.2}$$

Next, the input gate decide what new information is stored in the cell state. It is a two step gate. First, a sigmoid layer decides which values to update, with its equation below.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{3.3}$$

Thereafter, a tanh layer creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state. It is defined in the below equiation.

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3.4}$$

The cell states builds on the forget gate and the input gate outputs. As such, it calculates what is left from the previous cell state, instructed by the forget gate and calculated as follows: $f_t * C_{t-1}$. It further computes what needs to be added to the cell state, instructed by the input gate, and calculated as follows: $i_t * \tilde{C}_t$. The sum of these two values make the new cell C_t, defined below.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.5}$$

Lastly, the output gate filters the the cell state to decide what to output to the next timestep. It does so by applying a sigmoid function defined below.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{3.6}$$

Further, the cell state is passed through a tanh activation. the activated cell state and the output gate values are then multiplied to get h_t. This step is mathematically defined below.

$$h_t = o_t * tanh(C_t) \tag{3.7}$$

3.1.4 TextRank

TextRank is a graph-based ranking model unsupervised method for sentence extraction, introduced by Mihalcea and Tarau (2004). It is based on a voting mechanism, where the higher the number of votes are for a node, the more important it is, as the below equation defines it, where we have, for $G = (V, E)$ a directed graph: V the nodes, E the set of edges - a subset of VxV - $In(V_i)$ a set of vertices (nodes) that point to its predecessors and $Out(V_i)$ the set of vertices that vertex V_i points to its successors. d is assigned with a probability value which allows the model to jump randomly from vertex to vertex.

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{3.8}$$

TextRank for sentence extraction represents each sentence as a node, and assigns each node with a ranking depending on their similarity to other sentences. Similarity is calculated as follows:

$$Similarity(S_i, S_j) = \frac{|(w_k | w_k \in S_i and w_k \in S_j)|}{log(|S_i|) + log(|S_j|)} \tag{3.9}$$

Where, S_i and S_j are two sentences, with N_i the number of words in the sentence $S_i = w_1^i, w_2^i, ..., w_{N_i}^i$.

Once the graph built, TextRank applies the ranking algorithm, based on PageRank(Brin and Page, 1998), on the graph and selects the top ranked sentence for the summary.

3.1.5 LexRank

LexRank is a graph-based method model for text summarisation, developed by Erkan and Radev (2004). LexRank uses eigenvector centrality to find out the most important sentences in a document. It is calculated as per the below equation.

$$p(u) = \sum_{v \in adj[u]} \frac{p(v)}{deg(v)} \tag{3.10}$$

Where $p(u)$ is the centrality of the node u, $adj[u]$ the set of adjacent nodes to u and $deg(v)$ the degree of the node v.

The LexRank algorithm, also based on PageRank (Brin and Page, 1998), then assigns scores to the nodes based notably on their eigenvector centrality.

3.1.6 Baseline model

To complete the methodology, we run a pre-trained BertForSequenceClassifcation model with no training specific to our data in order to assess the impact of fine-tuning. Further, we make predictions on the test-data with well established extractive summarisation models to compare their performances on development project papers against our models' performances.

3.1.7 Implementation details

In this section, we detail the necessary steps carried to train our models and predict summaries. We also describe the methodology chosen to allow a good evaluation of the models' performances. Below is a figure summarising the main steps of this process.

Figure 3.3: Methodology framework

Data Collection

The development project papers' description were collected from the International Aid Transparency Initiative (IATI) website, which gathers all data from development actors who inform their projects' detail to the IATI standards. Precisely, the IATI publishers include a broad range of organisations, from donor governments, development finance institutions and UN agencies to non-governmental organisations, foundations, and private sector organisations. We use their Datastore query builder to collect a total of 908,196 rows of development projects, informed with 97 columns of features ranging from the name of the organization, the geographical region, the scope of the project or the budget involved.

Data Cleaning

The unnecessary columns from the IATI data set were removed, resulting in keeping only the project title and the description columns. Further, only the descriptions written in english were

retained. After writing a function to count the number of words per descriptions, the 5,000 descriptions with the longest descriptions were kept. Lastly, after manually investigating some project descriptions, unwanted characters were removed from the text. We then exported the file to excel for the next step, which is to manually annotating the summaries.

Data Annotating

The dataset did not contain any 'gold' summaries – a human written text summarizing the content of the description, necessary for the supervised training involved in the fine-tuning of the models and their evaluation. For this reason, the gold summaries were manually extracted from the selected descriptions. Due to a lack of time resource 3,000 descriptions out of the 908,196 were retained for which summaries were manually extracted. It is important to stress that the summaries were extracted from the original text and not written. In other terms, the summaries were one or two sentences that were in the original text and that best summarised the text to the judgement of the author. Although this was a time-consuming task, it was necessary for both fine-tuning the main models and to later evaluate their performance.

Data Wrangling and splitting

In this section, we implement data wrangling to later ease the step of pre-processing the text. Precisely, we define a function to assign a label of 1 to every sentence that is present in the summary and a label of 0 to the sentences that are not. To do so, we first tokenize each sentence using the 'en-core-web-lg' tokenizer from the spacy library. We then feed these to a list, and then embed each sentence from the documents and from the summaries using the 'distilbert-base-nli-mean-tokens' model from the Sentence Transformer huggingface library. With these embeddings, we assign the index of the document embeddings that have a high cosine similarity with the embeddings of the summary to a new array (e.g. if the second and the fifth sentences in the document are sentences that are in the summary, the array will be [1,4]). This then allows to create an array with a label of 0 if the sentence is not in the description, and 1 if the sentence is in the summary, with the length of the array naturally corresponding to the number of sentences in the original document. In addition, since the model will later classify the sentences on a sentence level, we add a document label to each sentence in the documents to later know what original text the predicted summary belongs to.

For instance, if the third description contains 10 sentences, it will be associated with the following array [2, 2, 2, 2, 2, 2, 2, 2, 2, 2]. This prepares for the next step which is to create a new data-frame with every sentence being a row, associated with its label and its document label (see figure 3.4). After these steps, we split the data-set into a training data-set and a test data-set using the skicit-learn train-test-split function. We choose a 0.8 split for the training data-set and 0.2 for the test data-set.

	sentence	label	document_label
0	The development objective of the Ecosystem Con...	1	656
1	The project comprises of four components.	0	656
2	The first component, pilot landscape planning ...	0	656
3	The second component, sustainable use of natur...	0	656
4	It consists of two sub-components: (i) sustain...	0	656

Figure 3.4: Training data point examples

Data Exploration

The next important step is to explore the data. Namely, we look at both the descriptions and the summaries' statistics, such as their mean length and the sentences length distribution. These statistics are shown in the below tables 3.1 and 3.2.

Type of text	**Sentence length average**
Document	13.7
Summary	1.3

Table 3.1: Average length of documents and summaries

Percentile	Document number of sentences	Summary number of sentences
0.00	2.0	1.0
0.25	9.0	1.0
0.50	11.0	1.0
0.75	15.0	1.0
0.95	30.0	3.0
0.99	53.6	4.0
1.00	173.0	9.0

Table 3.2: Sentences length distribution

Furthermore, the most frequent words appearing in the summaries are looked at to gain understanding of the data that is treated. Figure 3.5 showcases such words.

Figure 3.5: Most frequent words in gold summaries

We note here that we see key words such as development, objective, aims, contribute or increase which are words used to explain how one project brings aid and value to a specific area.

Pre-processing the text with BERT pre-trained language model

For the purpose of pre-processing, only the encoder part of BERT was used. This produces an output which is then used as input in both the BERT fine-tuned and the LSTM fine-tuned models for extractive summarisation. More precisely, encoding text in BERT specific input format is obtained by applying the following steps:

1. Split the sentence into tokens As for all NLP tasks, it is crucial to split each sentence into word tokens, to later enable word embedding representations. We do so by using the BertTokenizer function from the Huggingface transformer's library.

2. Add special tokens to each sentence For the task of classification, we must append a special token [CLS] at the beginning of each sentence. This allows the hidden state of the first token to be taken to represent the whole sentence. Furthermore, we append a special token [SEP] at the end of each sentence to separate each of them.

3. Map the tokens to their input IDs Each token is mapped to a singular ID which locates the word in the vocabulary. This can be done manually where we assign a new ID to each new word the training corpus contains. However, since we use the pre-trained Huggingface encoder, it is done based on the vast vocabulary BERT was initially trained on. Therefore, when a word in our text exists in the corpus BERT was trained on, it will be assigned the ID of that word.

4. Pad all sentences to the same length For BERT to manage the different lengths of each sentence, we must pad all sentences so that the input ID vectors are of the same length. To do so, we compute the largest sentence's length in the training set and add as much [PAD] tokens as needed for other sentences to match that maximum sentence length.

5. Create an attention mask which differentiate real tokens from padded tokens The attention mask is simply an array of 1s and 0s where 0s indicate when the token is a [PAD] token, and 1s when it is a normal token. This mask tells the Self-Attention mechanism in BERT not to take these PAD tokens into account for its interpretation of the sentence.

All these steps were attained by using the encoder-plus method proposed by huggingface for BERT tokenization.

Importantly, we note that no stopwords removal nor lemmatization or other NLP common text cleaning steps were implemented since BERT is best implemented on raw text, compared to other word embedders used in the field.

Fine tuning models

Fine Tuning BERT

We used Google Colab's free GPU to fine tune BERT to our task of classifying development sentences on wether they belong to a summary or not.

Precisely, we loaded the bert-base-uncased model from the BertForSequenceClassification pre-trained model proposed by huggingface. This model is the normal BERT with an added single linear layer for the task of classification. We train it with an ADAM learning rate of 2e-5, a batch size of 32 on 4 epochs to start with.

Figure 3.6: BERT 4 epochs training and validation loss

We see on the above figure 3.6 that 4 epochs is causing over-fitting issues with the validation loss being higher than with less epochs and the training loss being at its lowest. We therefore decide to train the model with the same hyper-parameters but on 2 epochs instead to avoid over-fitting issues.

Fine Tuning the stacked LSTM

Google Colab's free GPU is also used to train the stacked LSTM.

To fine-tune it, we compile it choosing the ADAM optimization algorithm for the optimizer, a log loss function given the binary nature of this classification task and a batch size of 50, on 25 epochs.

We note that the required inputs for this model are different than for the BERT classifier. Indeed, we only plug the input IDS as inputs. Furthermore, the model expects an input shape and dimension and an output dimension. For the input dimensions, we use the size of BERT's vocabulary. Because this model is meant to be generalised, we chose a large value of 30,200 so that the vast majority of the existing vocabulary in BERT can be read. We chose an output dimension of 64. For the input shape, we plug the maximum sentence's length. Again, we choose a longer sequence than the longest sentence's length in the training data for generalisation purposes.

Making predictions with main models and baseline models

Predictions with Baseline models

We use the Sumy library for both TextRank and LexRank. Since these are not machine learning classifiers, we do not run them on BERT input. Instead, the following text pre-processing steps are carried. Firstly, since models do not behave like classifiers, we gather all separated sentences by their document labels so that each row corresponds to the entire descriptions. Thereafter, we tokenize the text, to then parse it. Parsing text is the process of analysing sequential tokens to determine its grammatical structure. We then feed this into the loaded TextRank or LexRank summariser for the models to generate predictions of the parsed texts. We create a function to carry these steps, and then apply it to all the descriptions in the test data-set. We note that, we generate one sentence These generated summaries are then appended to a list, which is later added as a column to the test data-set, which we rename df-summaries.

Predictions with main models

With regards to BertSummaDev, we repeat the pre-processing steps on the test data-set to feed the trained model. We then put the model in eval mode for it to generate predictions on the test

data-set. We output the results as logits, which renders a two column array per sentence, which maps the probability of a sentence belonging to the summary or not. Specifically, the first column gives the probability score (from -1 to 1) for a sentence to not be part of the summary, and the second column the probability score for the sentence to be part of the summary. We then assign the sentences with the label 0 if the first column of the logits array has a higher score, and 1 if the second column has a higher score. We add this list of predictions to the test data-set, which we rename df-summaries, and group all the dataframe by the document label, such that each row represents a single original document, alongside the list of the reference labels when the sentence was in the original summary, and a list of prediction labels. We write a function to generate the predicted summaries by extracting the sentences that correspond to the index values that were labelled with a 1 in the prediction list. For example, if a description has 10 sentences, and the prediction list is as followed: (0,0,1,0,0,0,0,0,0,0), then the sentence with an index of 2 in the description will be extracted as the machine generated summary. We append these generated summaries to a list and add them to the df-summaries dataframe. With NeuSummaDev, the approach is the same except that we feed in only the input IDs as X-test. Furthermore, the model generates probabilities from 0 to 1, in contrast with logits which gives scores to the probability of a sentence to be classified as 0 or 1. Therefore, we write a function to create a list that scores 0 if the probability for a sentence is less than 0.5 and 1 if that probability is above 0.5. Once we have this list of predictions, we extract the generated summaries in the same way the summaries were extracted for BertSummaDev.

Evaluating models performance with ROUGE scores

The last step of the methodology was to evaluate each model performance with the ROUGE score metrics, standing for Recall-Oriented Understudy for Gisting Evaluation. ROUGE scores consist of a set of metrics that measure accuracy for language translation and summarisation. ROUGE-N, a metric of the set, measures the number of matching n-grams (a group of tokens) between the generated summaries and the reference summaries. N represents the n-grams that is used to evaluate the match-rate between the predicted and the reference summaries, with ROUGE-1 measuring the match-rate of unigrams and ROUGE-2 of match-rate of the bigrams. Furthermore, it proposes 3 metrics, the recall, the precision and the fmeasure. The recall is a ratio of the number of n-gram ground in the generated summary and in the reference summary out of the n-grams found in the

reference summary. The precision is a ratio of the n-grams found in the generated summaries and in the reference summaries out of the generated summaries. Lastly, the f1 score combines both recall and precision in its computation. We will report all three metrics for ROUGE-1 scores in this paper. In addition to the ROUGE-N scores, we also report the ROUGE-L and ROUGE-Lsum scores. L stands for Longest Common Subsequence based statistics. This metric calculates theses statistics between the reference and the predicted summary. In other words, it gives more weight to a predicted summary that has the longest sequential similarity with a reference summary. ROUGE-L, in contrast with ROUGE-N, has the benefit of taking into consideration the sequence of words and not just looking at whether a word in the predicted summary is also in the reference summary. We calculated these scores using huggingface's 'rouge-score' metric, which was applied by inputting the generated summaries along with the reference summaries. In this paper, we report the high recall ROUGE scores only. The other scores can be found in the notebooks.

Public Application deployment

Lastly, we deploy the summariser on an application for public usability. To do so, we push the BertSummaDev model to the huggingface hub, which allows to load the model from any machine since it is hosted on their github repository. Once this model pushed, we create a web application using streamlit, which turns python scripts onto shareable apps. We create a github repository with the appropriate requirement.txt file along with the source code which pre-processes input text and extracts summaries with the BertSummaDev model hosted on huggingface's hub. Unfortunately, NeuSummaDev was trained using Keras library, which does not allow to save a trained model on a public hub. Therefore, we could only use BertSummaDev for this application.

3.2 Results

In the below table 3.3 are displayed the ROUGE scores for both our baseline model, the well-established extractive summarisation models, and our models.

ROUGE score	Baseline model	LexRank	TextRank	BertSummaDev	NeuSummaDev
R1	43.3	78.0	76.4	**91.9**	**96.8**
R2	27.0	65.1	57.9	**89.1**	**95.1**
RL	37.6	72.6	68.5	**91.1**	**96.0**
RLsum	37.9	72.6	68.6	**91.1**	**96.2**

Table 3.3: ROUGE scores

We note that BertSummaDev's recall ROUGE scores are almost always above 90, with the highest being close to 92. NeuSummaDev reaches scores above 95 in every case. In contrast, we note poor yet expected random scores from the Bert For Sequence Classification model, and the underperformances of both LexRank and TextRank. We will discuss these results in more depth in the next chapter.

4 Discussion and Critical Evaluation

4.1 Findings

4.1.1 Introducing SummaDevDocs, a dataset for development project summarisation

One important contribution of this paper reside in the constitution of a new summarisation data-set, gathering 3,000 descriptions of development projects, associated with purely extractive summaries. There is room to think that this methodology enables more accurate fine tuning of a task, as the results from this paper show. Evidence outlined by earlier work (Cohan and Goharian, 2016) show that human-written summaries can affect the reliability of summarisation evaluation methods such as ROUGE (Lin, 2004). The below table 4.1 compares characteristics of our constructed SummaDevDocs against other existing datasets made for automatic summarisation present in the literature.

From this comparative table, we note several points. First, the size of the data-set is part of the smallest, although comparable in size to other datasets in the literature. This is justified as we fine tune pre-trained models that were already trained on vast amounts of text, avoiding the need for extensive training to obtain satisfying results, as ours show. The length of each document, however, is the shortest of all the displayed datasets. Further, its compression ratio, which is the ratio of the number of words in the summaries against the number of words in the document, is among the lowest. However, this is also due to the nature of the development project papers, as we saw earlier that only one sentence was extracted per document for most of the project descriptions. Lastly, we note that the percentage of novel words in the SummaDevDocs is the only to equal 0 percent, which is interesting as it gives material for comparison as to what is the best annotating method to perform great results.

Dataset	Nb. of documents	Avg. words in document	Avg. words in summary	Compression ratio	% novel words
DUC (Over, 2003)	624	441	11	40.1	30.0
NYTimes (Sandhaus, 2008)	655K	549	40	13.7	20.1
DailyMail (Hermann et al., 2015)	220K	653	55	11.9	17.0
XSUM (Narayan et al., 2018)	93K	760	46	16.5	16.8
Newsroom (Grusky et al., 2018)	1.32M	659	27	24.4	26.0
BigPatent (Sharma et al., 2019)	1.34M	3.6K	117	30.5	13.6
SciSummNet (Yasunaga et al., 2019)	1K	4.7K	150	31.2	7.4
TalkSumm (Lev et al., 2019)	1.7K	4.8K	965	5.0	16.5
SciTldr (Cachola et al., 2020)	3.2K	5K	21	238.1	15.2
SummaDevDocs (Ours)	**3K**	**314**	**39**	**8.2**	**0**

Table 4.1: Datasets comparison

4.1.2 NeuSummaDev and BertSummaDev outperform by a large margin other extractive models

As hypothesised, the fine tuned models outperformed by a large margin the baseline and the established models. This is extremely encouraging as it proves that we have, through this paper, built a model that summarises development paper projects better than any automatic summariser in the literature to the best of our knowledge. We remark the strong added value of fine tuning the models when compared to the performance of the Bert For Sequence Classifier, which classifies randomly.

More encouraging, is the outperformance of both our models compared to LexRank and TextRank. We note however, that LexRank and TextRank extracted two sentences per summary, which highly affects their precision, and therefore their f-measure scores since the average number of sentences in a gold summary was equal to 1 approximately. However, the recall score should not be affected, rather it should be more likely to be high, give it measures the percentage of the amount of words in the model and reference summary that is in the reference summary. By extracting two sentences, both TextRank and LexRank augment their chance to have a high recall score. Yet, we observe that none of their recall scores are close to either BertSummaDev or NeuSummaDev's scores, confirming the better performance of our fine-tuned models. Surprisingly, we obtain better results with the NeuSummaDev than with BertSummaDev, which goes against Liu's findings (2019) on the CNN/Daily mail dataset as the BERT classifier obtained a better ROUGE scores than. This adds to research by contradicting previous findings. In fact, NeuSummaDev is consistently and strongly better than BertSummaDev, with in average 8 point more of fmeasure in all ROUGE scores.

4.2 Practical implications

Other than the academic contributions stated above, this present work shows practical implications. With the results of the models being satisfying, a public application, SummaDev, was deployed in order for the developed models to have a direct real-life application. This can be used for Development Organisations' employees who will now not to read the entirety of a project descriptions to understand the purpose of one project, thereby enabling an increase in productivity. Furthermore, it has the potential to facilitate the understanding from local people of a project happening on their lands, as the application is easy to use with an access to the internet.

4.3 Limitations

One limitation that this paper presents resides in the data-set itself. Indeed, since all the descriptions were from the IATI, all the descriptions followed similar standards for informing their projects. Consequently, this might limit the generalisability of the models on development project papers that do not meet the IATI's requirements. Although, the application was tested with random devel-

opment project descriptions, and appeared to extract good summaries.

Another limitation resulting from the size of the data-set might be the number of data points the models were trained on. Indeed, it is less than the average training points researchers usually train their models on. However, as already mentioned, this was limited by training pre-trained models, which had already seen extensive amounts of data. Furthermore, the results did not appear to be affected by such sizes, although there is always room for improvement.

Lastly, the fact that we could not host NeuSummaDev on a public hub, which renders significantly better results than BertSummaDev, represents a limitation for this project since it is not used for public usability.

4.4 Future work

A various number of potential future research could be associated to this present paper. Namely, to enhance the applicability of the interface, one could add a translator from english to local languages. This would enable local inhabitants to enhance their understanding on the actions of development banks and their stakeholders. Furthermore, to facilitate even more this latter point, abstractive summarisation could be developed to simplify the vocabulary used in development project papers. Academically, it would be interesting to compare the performance of abstractive summarisation on the SummaDevDocs compared to the extractive methods presented in this paper. This would require human evaluation, or novel annotation to evaluate with metrics such as ROUGE. Another point of extensive work to this paper would be to augment the size of the dataset, with a different source of development papers than the IATI to augment the generalisability of the models. Lastly, NeuSummaDev, with its configurations saved on our github repository, could be hosted publicly and therefore loaded from any machine in order for it to be used on SummaDev, our public summarisation application.

5 Conclusion

In this paper, we have built BertSummaDev and NeuSummaDev, two extractive summarising models built on top of BERT pre-processed text. These models were built by replicating methodologies introduced by Liu (2019), which presented state-of-the art ROUGE score results. These models were compared to a pre-trained Bert For Sequence Classification as a baseline. Such comparison showed that the process of fine-tuning enabled a significant rise in performance. Furthermore, the performance of BertSummaDev and NeuSummaDev were tested against two well-established extractive summarisers, LexRank and TextRank, and demonstrated to outperform them by a large margin when looking at their ROUGE scores on the same test data-set. We surprisingly find that NeuSummaDev outperforms BertSummaDev, which goes in contradiction with Liu's findings, who found that adding a simple classifier on top of BERT resulted in better performance than a LSTM on top of BERT output.

To the best of our knowledge, no models in the literature have proven to perform better results in the task of summarising development paper projects than the models developed in this paper. As a result, we deployed a public application, SummaDev, to summarise development project descriptions with BertSummaDev.

In addition to building BertSummaDev and NeuSummaDev, we introduce SummaDevDocs, a new data-set for development papers summarisation, incentivising future research around development project papers' summarisation. It consists of 3,000 descriptions associated with purely extractive gold summaries. This data-set leaves room for improvement in developing better extractive summarisation models for development papers, or to build abstractive summarisation models.

This work has real-life application, since it can be used by employees from development organisations as a tool to summarise longer descriptions, hence resulting in productivity gains. Furthermore, it can be used by local populations for them to enhance their understanding of the various development projects that are implemented in their region. To augment its applicability to local populations, we suggest future research focuses on adding a translator on top of the model and

integrating it to the application.

6 References

Alcántara-Ayala, I. (2002). 'Geomorphology, natural hazards, vulnerability and prevention of natural disasters in developing countries.' *Geomorphology*, 47(2-4), pp.107-124.

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., D., E., B., J. and Kochut, K. (2017). 'Text Summarization Techniques: A Brief Survey', *International Journal of Advanced Computer Science and Applications*, 8(10).

Aone, C., Okurowski, M. E., Gorlinsky, J. (1998). 'Trainable, scalable summarization using robust NLP and machine learning', *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 1, pp. 62-66.

Bahdanau, D., Cho, K., Bengio, Y. (2014). 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473.*

Barnard, E., Davel, M. H. Van Huyssteen, G. B. (2010). 'Speech technology for information access: A South African case study', *Proceedings of the AAAI Spring Symposium: Artificial Intelligence for Development*, pp.13–15.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C. (2003). 'A neural probabilistic language model', *The journal of machine learning research*, 3, pp. 1137-1155.

Brin, S., Page, L. (1998). 'The anatomy of a large-scale hypertextual web search engine.' *Computer networks and ISDN systems*, 30(1-7), pp.107-117.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G. (2005). 'Learning to rank using gradient descent', *Proceedings of the 22nd international conference on Machine learning*, pp. 89-96.

Cachola, I., Lo, K., Cohan, A., Weld, D. S. (2020). 'TLDR: Extreme summarization of scientific documents', *arXiv preprint arXiv:2004.15011.*

Cantú, F., Saiegh, S. M. (2011). 'Fraudulent democracy? An analysis of Argentina's infamous decade using supervised machine learning', *Political Analysis*, 19(4), pp.409-433.

Chen, F. and Neill, D. (2015). 'Human Rights Event Detection from Heterogeneous Social

Media Graphs', *Big Data*, 3(1), pp.34-40.

Chen, S., Zhang, F., Sone, K., Roth, D. (2021). 'Improving faithfulness in abstractive summarization with contrast candidate generation and selection', *arXiv preprint arXiv:2104.09061.*

Cheng, J., Lapata, M. (2016). 'Neural summarization by extracting sentences and words', *arXiv preprint arXiv:1603.07252*

Collobert, R., Weston, J. (2008). 'A unified architecture for natural language processing: Deep neural networks with multitask learning', *Proceedings of the 25th international conference on Machine learning*, pp. 160-167.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. (2011). 'Natural language processing (almost) from scratch', *Journal of machine learning research*, 12, pp.2493-2537.

Conroy, J. M., O'leary, D. P. (2001). 'Text summarization via hidden markov models', *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 406-407.

Dai, A. M., Le, Q. V. (2015). 'Semi-supervised sequence learning', *Advances in neural information processing systems*, 28, pp.3079-3087.

De-Arteaga, M., Herlands, W., Neill, D. B., Dubrawski, A. (2018). 'Machine learning for the developing world', *ACM Transactions on Management Information Systems (TMIS)*, 9(2), pp.1-14.

Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805.*

Erkan, G. Radev, D. (2004). 'LexRank: Graph-based Lexical Centrality as Salience in Text Summarization', *Journal of Artificial Intelligence Research*, 22, pp.457-479.

Ermon, S., Xue, Y., Toth, R., Dilkina, B., Bernstein, R., Damoulas, T., ... Gomes, C. P. (2015). 'Learning large-scale dynamic discrete choice models of spatio-temporal preferences with application to migratory pastoralism in East Africa', *Twenty-Ninth AAAI Conference on Artificial Intelligence.*

Ferris, E. (2010). 'Natural disasters, conflict, and human rights: tracing the connections', *The Brookings Institution.*

Fuentes Fort, M., Alfonseca, E., Rodríguez Hontoria, H. (2007). 'Support vector machines for query-focused summarization trained and evaluated on pyramid data'.

Gong, Y., Liu, X. (2001). 'Generic text summarization using relevance measure and latent semantic analysis', *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.19-25.

Grace, E., Rai, A., Redmiles, E., Ghani, R. (2016). 'Detecting fraud, corruption, and collusion in international development contracts: The design of a proof-of-concept automated system', *IEEE International Conference on Big Data*, pp.1444-1453.

Grusky, M., Naaman, M., Artzi, Y. (2018). 'Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies', *arXiv preprint arXiv:1804.11283.*

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P. (2015). 'Teaching machines to read and comprehend', A*dvances in neural information processing systems*, 28, pp.1693-1701.

Howard, J., Ruder, S. (2018). 'Universal language model fine-tuning for text classification', *arXiv preprint arXiv:1801.06146.*

Jean, N., Burke, M., Xie, M., Davis, W. M., Lobell, D. B., Ermon, S. (2016). 'Combining satellite imagery and machine learning to predict poverty.', *Science*, 353(6301), pp.790-794.

Jia, X., Khandelwal, A., Nayak, G., Gerber, J., Carlson, K., West, P., Kumar, V. (2017). 'Predict land covers with transition modeling and incremental learning', *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp.171-179.

Kågebäck, M., Mogren, O., Tahmasebi, N., Dubhashi, D. (2014). 'Extractive summarization using continuous vector space models', *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pp. 31-39.

Kapoor, A., Eagle, N., Horvitz, E. (2010). 'People, quakes, and communications: Inferences from call dynamics about a seismic event and its influences on a population', *AAAI Spring Symposium Series.*

Kupiec, J., Pedersen, J., Chen, F. (1995). 'A trainable document summarizer', *18th annual international ACM SIGIR conference on Research and development in information retrieval*, pp.68–73.

Lev, G., Shmueli-Scheuer, M., Herzig, J., Jerbi, A., Konopnicki, D. (2019). 'Talksumm: A dataset and scalable annotation method for scientific paper summarization based on conference talks',*arXiv preprint arXiv:1906.01351.*

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2019). 'Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension', *arXiv preprint arXiv:1910.13461.*

Lin, C. Y., Hovy, E. (2002).'Automated multi-document summarization in neats', *Proceedings of the Human Language Technology Conference*, pp.23-27.

Liu, X., Zhou, Y., Zheng, R. (2007). 'Sentence similarity based on dynamic time warping', *International Conference on Semantic Computing (ICSC 2007)*, pp. 250-256.

Liu, Y. (2019). 'Fine-tune BERT for extractive summarization', *arXiv preprint arXiv:1903.10318.*

Liu, Y., Lapata, M. (2019). 'Text summarization with pretrained encoders', *arXiv preprint arXiv:1908.08345.*

Lloret, E., Palomar, M. (2009). 'A gradual combination of features for building automatic summarisation systems', *International Conference on Text, Speech and Dialogue*, pp. 16-23.

Lloret, E., Palomar, M. (2012). 'Text summarisation in progress: a literature review.' *Artificial Intelligence Review*, 37(1), pp.1-41

Luhn, H. P. (1958). 'The automatic creation of literature abstracts', *IBM Journal of research and development*, 2(2), pp.159-165.

McCargar, V. (2004). 'Statistical approaches to automatic text summarization', *Bulletin of the american society for information science and technology*, 30(4), pp.21-25.

McKeown, K. R., Barzilay, R., Evans, D., Hatzivassiloglou, V., Klavans, J. L., Nenkova, A., ... Sigelman, S. (2002). 'Tracking and summarizing news on a daily basis with Columbia's Newsblaster', *Proceedings of the Human Language Technology Conference*, pp.280-285.

Mihalcea, R., Tarau, P. (2004). 'Textrank: Bringing order into text', *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404-411.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J.(2013). 'Distributed representations of words and phrases and their compositionality', *Advances in neural information processing systems*, pp. 3111-3119.

Murray, G., Renals, S., Carletta, J. (2005). 'Extractive summarization of meeting recordings'.

Nallapati, R., Zhai, F., Zhou, B. (2017). 'Summarunner: A recurrent neural network based sequence model for extractive summarization of documents', *Thirty-First AAAI Conference on Artificial Intelligence*.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B. (2016). 'Abstractive text summarization using sequence-to-sequence rnns and beyond', *arXiv preprint arXiv:1602.06023.*

Narayan, S., Cohen, S. B., Lapata, M. (2018). 'Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization', *arXiv preprint arXiv:1808.08745.*

Narayan, S., Cohen, S. B., Lapata, M. (2018). 'Ranking sentences for extractive summarization with reinforcement learning', *arXiv preprint arXiv:1802.08636.*

Over, P. (2003). 'An introduction to DUC 2003: Intrinsic evaluation of generic news text summarization systems', *Proceedings of Document Understanding Conference 2003.*

Ozsoy, M. G., Alpaslan, F. N., Cicekli, I. (2011). 'Text summarization using latent semantic analysis', *Journal of Information Science*, 37(4), pp.405-417.

Parvez, M. R., Mosharraf, T., Ali, M. E. (2016). 'A novel approach to identify spatio-temporal crime pattern in dhaka city', *Proceedings of the eighth international conference on information and communication technologies and development*, pp. 1-4.

Pasunuru, R., Bansal, M. (2018). 'Multi-reward reinforced summarization with saliency and entailment', *arXiv preprint arXiv:1804.06451.*

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018). 'Deep contextualized word representations.' *arXiv preprint arXiv:1802.05365.*

Peters, M. E., Neumann, M., Logan IV, R. L., Schwartz, R., Joshi, V., Singh, S., Smith, N. A. (2019). 'Knowledge enhanced contextual word representations.' *arXiv preprint arXiv:1909.04164.*

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018). 'Improving language understanding by generative pre-training.'

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). 'Language models are unsupervised multitask learners.' *OpenAI blog*, 1(8), pp.9.

Rubio, A., Frias-Martinez, V., Frias-Martinez, E., Oliver, N. (2010). 'Human mobility in advanced and developing economies: A comparative analysis', *AAAI Spring Symposium Series*.

Sandhaus, E. (2008). 'The new york times annotated corpus', *Linguistic Data Consortium.*

See, A., Liu, P. J., Manning, C. D. (2017). 'Get to the point: Summarization with pointer-generator networks', *arXiv preprint arXiv:1704.04368.*

Sharma, E., Li, C., Wang, L. (2019). 'Bigpatent: A large-scale dataset for abstractive and

coherent summarization', *arXiv preprint arXiv:1906.03741.*

Sparck, J. (1993). 'Discourse Modeling for Automatic Text Summarizing' *Technical Report.*

Steinberger, J., Jezek, K. (2004). 'Using latent semantic analysis in text summarization and summary evaluation', *Proc. ISIM*, 4, pp.93-100.

Sutskever, I., Vinyals, O., Le, Q. V. (2014). 'Sequence to sequence learning with neural networks', *Advances in neural information processing systems*, pp.3104-3112.

Svore, K., Vanderwende, L., Burges, C. (2007). 'Enhancing single-document summarization by combining RankNet and third-party sources', *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 448-457.

Tehrany, M. S., Pradhan, B., Jebur, M. N. (2014). 'Flood susceptibility mapping using a novel ensemble weights-of-evidence and support vector machine models in GIS', *Journal of hydrology*, 512, pp.332-343.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). 'Attention is all you need', *Advances in neural information processing systems*, pp.5998-6008.

Wang, D., Liu, P., Zheng, Y., Qiu, X., Huang, X. (2020). 'Heterogeneous graph neural networks for extractive document summarization', *arXiv preprint arXiv:2004.12393.*

Wang, D., Liu, P., Zhong, M., Fu, J., Qiu, X., Huang, X. (2019). 'Exploring domain shift in extractive text summarization', *arXiv preprint arXiv:1908.11664.*

Wesolowski, A., Eagle, N. (2010). 'Parameterizing the dynamics of slums', *AAAI Spring Symposium Series.*

Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., Affandy, A. (2020). 'Review of automatic text summarization techniques methods', *Journal of King Saud University-Computer and Information Sciences.*

Yasunaga, M., Kasai, J., Zhang, R., Fabbri, A. R., Li, I., Friedman, D., Radev, D. R. (2019). 'Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks', Proceedings *of the AAAI Conference on Artificial Intelligence.*

Yin, W., Pei, Y. (2015). 'Optimizing sentence modeling and selection for document summarization', *Twenty-Fourth International Joint Conference on Artificial Intelligence.*

Zemel, R., Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. (2013). 'Learning fair representations.' *Proceedings of the 30th International Conference on Machine Learning (PMLR).*

Zhang, J., Zhao, Y., Saleh, M., Liu, P. (2020, November). 'Pegasus: Pre-training with extracted gap-sentences for abstractive summarization', *International Conference on Machine Learning*, pp.11328-11339.

Zheng, H., Lapata, M. (2019). 'Sentence centrality revisited for unsupervised summarization', *arXiv preprint arXiv:1906.03508.*

Zhong, M., Liu, P., Chen, Y., Wang, D., Qiu, X., Huang, X. (2020). 'Extractive summarization as text matching', *arXiv preprint arXiv:2004.08795.*

Zhou, Q., Yang, N., Wei, F., Huang, S., Zhou, M., Zhao, T. (2018). N'eural document summarization by jointly learning to score and select sentences', *arXiv preprint arXiv:1807.02305*