UNIVERSITÀ CATTOLICA DEL SACRO CUORE

Interfaculty Economics - Banking, Finance and Insurance

Sciences Master of Science in Statistical and Actuarial Sciences

Data Analytics for Business and Economics

Abstractive Text Summarization

Student: Ogulcan Ertunc

ID: 4811415

Supervisor: Prof. Andrea Belli

Academic Year 2019−2020

Abstract

Text Summarization is the task of extracting important information from the original text document. In this process, the extracted information is produced as a report and presented to the user as a short summary. It is very difficult for people to understand and interpret the content of different types of texts, the language structure used and the subject matter are the most important factors in this. In this thesis, our model is set up over a dataset created from news, by addressing abstractive text summarization methods, which are a state-of-the-art. This article collectively summarizes and decipheres the various methodologies, challenges, and problems of abstractive summarization.

The importance of abstractive summary is to save time, as understanding both the abundance of documents and the required and unnecessary documents in many industries is a huge waste of time.

Here we have done abstract text summarization using the latest advances in the pre-trained NLP model OpenAI GPT-2 to solve this problem. Abstract text summarizing, all abstracts summarize the main topics in a way that imitates a summary task in a document written by human creation. As a result of recent advances in deep learning and especially the great efforts of OpenAI, the applicability of abstractive text summarization models in real situations has increased. However, the biggest problem here is the availability of the necessary data to train the models. A commonly used pre-trained model to

overcome this problem is the use of transfer learning. Our model provides abstract and comprehensive information by correlating the original data and the keywords we create from the original data. Summaries were not permanent, some were poor readability due to some logical errors. Our study can help any industry needed by providing brief summaries of articles that are not yet available. Choosing data sets and fine-tuning more models could lead to more successful results if taken into account in future studies.

Keywords: GPT-2, Abstractive Text Summarization

Table of contents

List of figures

List of tables

Chapter 1: Introduction

1.1 Summarization

Abstract text summarization is an active research area that focuses on decreasing the main text into a structurally smaller text while preserving relevant information. There are two general ways. The first is inferential summarization that aims to extract and consolidate substantial scope of the source text, and it is relatively easy. This is similar to reviewing text.

The second approach creates new meaningful summaries from the main text. It has been shown that the inferential approach that GPT-2 uses to create text maintains reasonable grammar and accuracy. On the contrary, the model is able to preserve the meaning integrity of the main text and then make shorter meaning matching for meaningful new texts, the abstractive approach is much more difficult. However, the model can acquire the ability to use words creatively or infer from the source text, along with keywords.

1.2 Types of Summarizations

1.2.1 Extractive Summarization

In one of the recent studies on NLP, Cheng and Lapata [VII] talked about a neural network-based encoder extractor architecture for generating extractive abstracts. The proposed model obtained state-of-the-art results and explained the strength of deep learning methods for extractive summarization.

Nallapati [VII] introduced an innovative training method for extractive models. The training is finished in an abstractive way, hence only needing human-generated summaries moreover to training abstractive models. Nallapati [XXII] suggested an extractive model leading to state-of-the-art performance.

This research project provides a comprehensive analysis of the use of sentence similarities. Collectively, these subjects outline the probability of working extractive approaches. An important highlight of the extractive approach is that the summaries are accurate in fact typing due to the extractive nature. Nevertheless, abstractive summarization has the characteristic of being easier and smoother to read.

1.2.2 Abstractive Summarization

One of the first deep learning techniques of abstractive summarization was the use of the Attentional Encoder-Decoder Recurrent Neural Network for the text summarizing task, as suggested by Nallapati [XXV]. This model to be built is basically for machine language translation but has been successfully applied to text summarization after fine tuning. Two salient boundaries are that summarizing larger records can result in a lot of constant text and some of the knowledge is mistakenly interpreted.

This can be explained by the fact that certain information is deemed necessary by the model and therefore is repeated over and over again. In many important studies, a solution has been proposed for this limitation by combining extraction and abstractive ways. To deal with the problem of inaccurate information, a pointer mechanism is introduced in addition to the careful array-to-array network. This pointer mechanism is like an extraction method, similar to an extraction method, it provides words in the source text and copies them into the summary. This actually has a great accuracy benefit in reporting. Coverage has also been covered in the network, which has been shown to be very useful in diminishing recurrence.

Zhang [L] combined a NLG model based on BERT.We will try to obtain our summaries by using our model in a similar way by pre-training with the goal of learning a language model.

1.3 A State-of-the-Art Model for Abstractive Text Summarization

1.3.1 Overview of State of the Art

State of the Art simply means that this technology is the latest / most recent version. It can also be referred to as the best technology available, as it has been developed using the most modern techniques and technologies. Humans are among the most important parts of a machine's environment and giving machines the ability to cooperate with humans is one of the most interesting and probably useful challenges for modern engineering. Therefore, the use of machine learning algorithms is increasing day by day. Regularly, new versions of the currently used sensors or new techniques that give better results are presented. Some of these offer minor advances, while others offer progress beyond predictions. There are well-written articles that explain the underlying technique for most techniques, sometimes including code and tutorials. However, there is no general method by which we can measure how much improvement they offer, so evaluating the results based on the task (performance scores) turned out to be the most accurate method. There are a variety of tasks in many areas, but to name the tasks that are common in recent times, Natural Language Processing, Speech Recognition, and Computer Vision.

1.3.2 State of the art in NLP

A common definition for Natural Language Processing is this: NLP is a subfield of AI that gives machines the ability to read, understand, and extract meaning from human languages.

NLP tasks cover a very wide range and differ from each other, and as the definition suggests, they are all about making sense of our language. In other words, it tries to make calculations according to our language and the components of our language. NLP algorithms can be found in many applications and various industries. Translators, grammar checkers, virtual assistants, chatbots and spam filters that we also use in daily life are what NLP offers us.

Major known examples of NLP:

1. Sentiment Analysis
2. Machine Translation
3. Text Classification
4. Question Answering
5. Recommendation Systems
6. Speech Recognition
7. Language Modelling

The state of the art that we took as basis in our project was Language Modelling. Language modelling is the simplest task of predicting the next word (s) based on the existing text and previous words. There have been many successful

studies in this area. The models created by Megatron-Lm and GPT are the most successful.

The model uses the Transformer Network. In his work, a transformer layer consists of a self-attention block and then a two-layer, multi-layer perceptron (MLP). Model parallelism is used in each of the blocks. This type of operations requires a GPU because it brings a heavy load.

1.3.3 State-of-the-Art Model for Abstract Text Summarization

The most efficient way to learn that people have understood a text is often to create a summary about that text. Abstractive text summarization, on the other hand, is one of the most challenging tasks in the NLP field, which includes three very important factors such as text comprehension, information presentation and language creation. Although the initial experiments focused on seq2seq or recurrent neural network solutions, it was realized that this was a problem in long texts, and transformer-encoders / decoders were found to be more effective in modelling the dependencies present in long texts used to summarize.

For example, the BERT, T5, ELECTRA and GPT transformer models have shown that they reach state-of-the-art permissions when fine-tuned as well as a good pre-training.

1.4 Structure of This Work

Both these traditional and reproduced summarization approaches have advanced significantly, thanks to recent advances and the availability of pre-trained (T5, BERT, USE, ConveRT and GPT) NLP models that often take advantage of the attention mechanism. Our study includes GPT-2, the older version of OpenAI GPT-3 that has been introduced more recently than transformers of these models, which is currently being distributed to certain individuals. However, many previously designed methods (T5, Electra, and BERT) fail for abstractive summarization as they are not built to produce meaningful text. However, GPT-2 is trained on all of the many resources on the internet and on a large text data set such as translation, multiple choice question answering, text classification, etc. It performs well in a variety of NLP missions including

In the past few years, array-to-array (seq2seq) models based on transformer decoder architecture have been widely used for abstractive abstraction. Structurally, the encoder reads the given text, establishes a link with keywords, and the encoder takes hidden states and outputs an abstractive summary text. Matching from hidden representation to output text provides the ability to generate architectural language. More recently, it has been a necessity to develop large language models so that text-to-text translation is used in multiple NLP tasks simultaneously. The main idea here is to train a single main model by matching various entered text with the output text. This work uses a similar spirit to fine-tune a pre-trained OpenAIGPT2 to map from a selected keyword to a summary text, thus creating an abstract summary.

1.5 News Summary Dataset

The data set consists of 4515 samples and includes Author_name, Titles, Article Url, Short text, Full Article. Kondalarao Vonteru collected summary news and only compiled news articles from Hindu, Indian Times and the Guardian. The time range varies from February to August 2017.

1.6 The Difficulty of Limited Dataset

An additional challenge to our mission, our model is focused on the requirement that large dataset to create a more efficient results. The CNN / Daily Mail data set, shown as an example in many sources, contains 286 thousand text and summary pairs. However, the data set we use in this project consists of approximately 4.5 thousand texts. This low level of data places a huge strain on fine tuning. However, this framework can be extended further if found useful.

Chapter 2: NLP Applied to Text Summarization

2.1 Auto Text Summarization

When we look at the text summarization process, it is a machine learning approach that can be applied both in a supervised and unsupervised manner. For machine learning, sentences are subjected to a pre-process process, and thanks to this, information such as keywords we know from NLP, sentence analysis or relevance is obtained. Using these properties, it works like a text hash classification algorithm. Keywords in news articles must be tagged "positively" to include them in the summary. Sentences not included in the summary are masked as "negative". The importance of the words in the sentence is found through TF-IDF in line with the features extracted from the sentence. With the use of masking data, the weights of the words in the sentence are learned and it is decided whether to use them in the abstract or not.

For these classification models, models such as Naive Bayes, Decision Trees and Support Vector Machines (SVM) are usually chosen. The model presents a summary after assuming a similarity between sentences.

2.2 Transfer Learning

One of the major difficulties for supervised learning is the trouble of finding appropriate data to train and mask the classifier. In addition, the data used must match the source texts provided for the summarizer. Some methods have been proposed to succeed in this issue. First, semi-supervised learning approaches that require less labelled data can be tried. Second, transfer learning can be implemented using other data. There is a lot of data available as open source for us to train the classification processes. However, while these methods often assume that tagged data and untagged data come from the same distribution, they allow the fields or tasks of the transferred learning to be different. However, many experts claim that the use of transfer learning is not generally approved for applications on NLP, unlike other areas such as computer vision. Therefore, there are two types of tuning techniques preferred for neural networks, fine-tuning, and freezing.

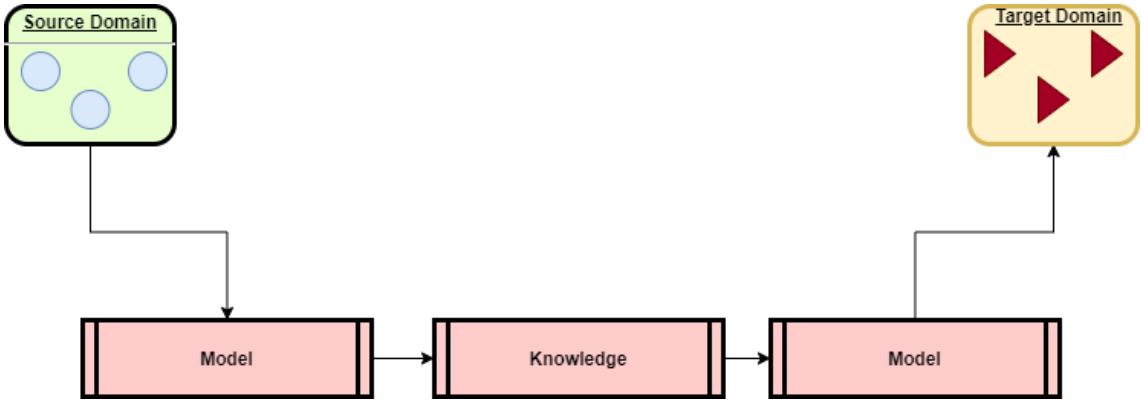

Figure 1: Transfer Learning Scheme

NLP studies are faced with the problem of not having enough captured data which makes transfer learning easier. Those who wanted to implement this

successfully often found that training times and performance were correlated. In addition to the successful results of transfer learning, it also faces overfitting, which is a common problem in deep neural networks. In our study, we tried to prevent overfitting (the text we used directly in the source appeared as output).

2.3 Generative Pre-trained Transformer

2.3.1 What is the OpenAI ?

OpenAI is a non-profit research company that aims to develop and lead artificial intelligence (AI) in a way that benefits humanity. The company was founded in 2015 by Elon Musk and Sam Altman and is headquartered in San Francisco, California.

The company's two founders and other investors founded the company with a $ 1 billion donation. In February 2018, Elon Musk left the company due to a conflict between OpenAI and Nikola Tesla-inspired electronics company Tesla.

OpenAI's first GPT model was mentioned in an article titled "Improving Language Understanding by Generative Pre-training" by Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever.

2.3.2 Brief History and Development of GPT and Transformer

GPT-3 is the newest language model of the OpenAI team. OpenAI released its announcement with detailed information about GPT-3 in May 2020, and in July OpenAI gave several lucky users access to the model to test the beta version via an API. The model was used to write poems, play games, or create simple applications that operate with a few buttons.

In the last decade, deep neural networks (DNN) and Natural Language Processing (NLP) based applications like I mentioned above have become common everywhere. Before deep neural networks, natural language processing solutions were unfortunately not very efficient. For example, when Google Translate first came out, his translations were producing barely consistent sentences, with many glaring errors. In the 2010s, NLP researchers realized the potential found in DNN and fully embraced this area as their main field of study. The combination of NLP and DNN in artificial intelligence applications has become a solution to eliminate bad results.

The first innovation with DNN was the use of neural networks to create word vector models (Word Representations). Rather than using the word itself in a machine learning algorithm, the idea was to first transform words into mathematical vectors, and this was a very accurate approach. Just as the GPT-3 has resonated these days, Word2vec revolutionized when it first came out in 2013. Word vectors had remarkable properties, and the results obtained were quite exciting. For example, when you take the vector of the word Milan in the word vector model, you can get Italy from its approach in language, and when you add France next to these two words in the model, the next word that the

system will give you will be Paris. It became possible to establish a relationship between Word2Vec and many similar words. GloVe technology, which is a similar approach, came out in 2014 and both word vector model approach became very popular, enabling 'state of the art' outputs to be obtained in many NLP applications.

The next major innovation was the use of Recurrent Neural Networks to read sentences and paragraphs. RNN had the advantage of being fed arbitrarily long strings of words and was able to maintain consistency in contexts consisting of many sentences. With the release of Seq2seq on RNN, this approach has become very popular, especially in machine translation. In 2016, Google took advantage of the newest improvements in RNN for NLP tasks and switched to a new Neural Machine Translation (NMT) engine, taking advantage of the previous Statistical Machine Translation (SMT) engine.

Although it produced very successful results, the output was still a little far from the desired level. The output of that period consisted of scattered and inverted sentences that turned out to be a distinct robot. It is good in terms of grammar and predicate, but when the sentences come together, the transitions and the integrity of the sentence could not achieve the desired rate.

In 2017, things started to develop and differentiate in a different direction. A new architecture called 'Transformer' has been introduced by Google. Thanks to this new structure introduced, it was important as it allowed the creation of much deeper and more complex neural networks. Computer Vision studies conducted during this period showed everyone how the performance of DNN systems can

be increased. Now the same energy was ready for NLP researchers. Thanks to the 'Transformer' ability to scale into deeper networks, teams have started releasing even larger models. The stable BERT model published by Google during this period had 340 million parameters. Salesforce's CTRL model is an enormous model with 1.6 billion parameters. In the field of NLP, these models work by choosing a random word when a sentence is given and guessing which word should replace the next word or spaces in the sentence. This approach is well suited to self-regulation. The model does not need any man-made label, so when only the target text is shown unsupervised, it can provide end-to-end self-education and model building. This feature opened the door to training within large databases.

However, producing transformer models comes at a cost. There are so various parameters on so many data that training speed is quite expensive and time-consuming. For this reason, researchers need large amounts of cloud computing power in “state-of-the-art” infrastructures. For this reason, only the world's largest and best funded teams can have the privilege of producing a new model. GPU-based servers with large hardware may be needed even for very small changes and configurations. 10 hours of training in a high-level Azure / AWS virtual machine (cloud computing) can be envisaged for some of the models being worked on. In this example, making the smallest mistake can be very high-priced, and repeating experiences multiple times become high-priced.

Learning with little training data may not seem like a big deal or a difficult thing, but this is one of the biggest active problems in the field of artificial intelligence. People can often learn a new function just by showing them a few times. Despite

the efforts of researchers, the ability of complex functions to learn from only a few sample data, or to learn without data at all, has not been achieved with machines until now, and here is one of the biggest differences between human and machine. Deep neural networks' hunger for data is a major disadvantage, they always need much more data to be better, because data are often difficult to find for many operations and it is very costly to create training sets by tagging new ones. If learning with a few examples of data had worked well, the use of artificial intelligence in many more areas than now would be in our daily lives.

On the GPT-3 side, for example, to understand the results, a human brain has about 100 billion neurons, which creates something at the level of 100 to 500 trillion synaptic connections. If this widely spoken scale provided by GPT-3 is truly a solution for human-like intelligence, then GPT-3 is still 1000 times smaller in capacity than the human brain. This is to assume that synaptic connections roughly match the parameters of the neural network, which, of course, do not match, we have already mentioned that in order to be like a human being, it is also necessary to be approached by different aspects of intuition and emotion. Human neurons are much more complex than any technology that can be provided with software, and they are still full of secrets waiting to be resolved.

Another very interesting result of the GPT-3 is how much a general pattern the approach provides. The traditional approach in the world of machine learning is that a model must be trained for a specific task or field, and it can only fulfil that goal. For example, AlphaGO, the computer that excels the world champion in a complex and difficult game like go, fails to perform in a simple game like tic-tac-toe, which is much simpler than go. GPT-3, unlike AlphaGO, can perform many

different tasks without any additional training. It is a surprisingly excellent language model. Once a news article title and the first sentence are given, it can predict the next word likely to appear, creating complete articles. The abstractive summary creation that we try to do as a project in this thesis is done in this way. After modelling the communication of the words in the text, we plan to create a new text for the summary.

However, GPT-3 models can be used for various purposes. It can translate between languages, so it works better than most of the existing cross-language translation applications. It can produce quite good results in reading comprehension, summarizing, and managing bilateral dialogues. You can also answer exam questions in educational practices and use similar applications in various sectors for many applications specific to language. GPT-3 has received training on so many texts in the internet environment and has so much capacity that we can say that it has memorized many facts about the world.

Surprisingly, the GPT-3 can also do things that its producers didn't think of. After OpenAI began granting selected developers beta access to its API, some showed that it is possible for GPT-3 to generate functional JavaScript code from a native language command prompt. Presumably, some of the web pages used in the training data had code samples and front-end codes. For this purpose, it has been revealed that the system can translate from English to French as well as from English to JavaScript.

2.3.3 What is GPT-2 ?

We used GPT-2 in our thesis as OpenAI only provides the API of GPT-3 to certain developers. Although GPT-2 is not as big as GPT-3 and can be considered as successful enough for us, GPT-2, a giant transformer based on almost 1.5 billion parameter language model, had an effect like GPT-2 when it was launched and for a long time. it remained on the agenda. GPT-2 is trained to predict the next word in a 40 GB internet text. The data set used consisted of 8 million internet pages.

The GPT-2 reflects multiple skills that link the power to produce unprecedented quality text examples where the model is fed with input and forced to form a large length.

Additionally, GPT-2 excels at many language models trained in specific fields such as Wikipedia, news or books. GPT-2 begins to learn these tasks from raw text without using task-specific training datasets in some language tasks, including question answering, reading comprehension, and explanation. In this context, we used GPT-2 in our project, as we think it can adapt to our project.

2.3.4 GPT-3

Productive Pre-Trained Transformer 3 (GPT-3) is the most up-to-date language model used to produce human texts with deep learning. The third generation of GPT, GPT-3, was introduced in May 2020 with 175 billion machine learning parameters in the full version, and then distributed to selected developers in the

beta phase. The closest in number of parameters to it is the Turing NLG created by Microsoft, and it has 17 Billion parameters.

The largest part that makes up the weighted pre-training dataset for GPT-3 comes from the Common Crawl version (about 410 billion pairs of coded tokens), with almost a quarter coming from WebText2. Apart from these, Books1, Books2, and Wikipedia are other great resources that make up the pre-education dataset.

While the GPT-3 was being trained, billions of words were studied. Apart from this, besides Python, which we can call Multiplatform, coding can be done in CSS and JSX languages. Since the training data of GPT-3 includes many things, it does not require pre-training for different language tasks. GPT-3 is one of the biggest developments in the field of NLP, whose API is accessed with a special language in line with its large number of parameters and potentials.

2.3.5 Why We Used GPT-2 Instead of Bert

Both models GPT and BERT are relatively new to the industry, but their cutting-edge performance has made them the most efficient and best among other models in natural language processing. However, GPT-3 trained on 175 billion parameters has a size almost 450 times larger than BERT.

Second, when training the model for certain downstream tasks when using BERT, a more detailed and difficult fine-tuning process is required than GPT's, however, GPT's text input and text output API allows users to reprogram and access it using instructions, recognizes. For many NLP applications such as

sentiment analysis, question-answering tasks, to use BERT, users need to train the model in sentence modelling on a separate layer, and this is an extra challenge for BERT. However, GPT uses a few impulse learning processes in the input token to predict the output result, thanks to the convenience it provides.

For general NLP tasks such as machine translation, answering questions, NLP's complex arithmetic calculations or learning new words, GPT learns in a few steps and works perfectly, conditioned with a few simple examples. Similarly, for text rendering, GPT-3 runs on several command prompts to quickly distribute the associated output with about 52% accuracy. To summarize, OpenAI, which created the GPT, increased the size and training parameters of the model, creating a more powerful model compared to its competitors.

BERT uses the masking system to understand the context of the word, randomly masks one-eighth of the words in each sequence to predict the result, it trained in mask language model tasks. Similarly, for BERT sentence prediction, it is fed with double sentence structure as input and then trained on an additional auxiliary task for prediction. Here, it processes both related sentences to guess the binary label of the sentence prediction. Here, it processes both related sentences to guess the binary label of the sentence prediction.

On the architectural dimension, BERT has been trained on the difficulties of hidden relationships between texts in different contexts, while the GPT-3 training approach is relatively straightforward compared to BERT. Therefore, GPT-3 may be the preferred choice in missions where sufficient data is not available for a wider range of applications. The transformer consists of two separate mechanisms, the encoder and the decoder. The BERT model simply

covers coding mechanisms to create a language model; However, GPT-3 does its job by combining the encoder and decoder to obtain a converter decoder to generate text.

GPT-3 is not currently publicly available (not available via an open source API) but BERT has been an open source model that allows users to fine-tune their needs from the beginning. While GPTs generate one token output each time, BERT on the other hand is not autoregressive, so it uses deep two-way context to predict the result in sentiment analysis and question answering.

Chapter 3: Methodology

3.1 Task Definition

We want to create an abstractive summary model with the News Summary, a public data set available on Kaggle.com. We aim to provide a pre-trained method for fine tuning. Our project consists of two parts. Used as an uncontrolled inference section and a newly abstracted text section.

An uncontrolled inferential hash takes a pre-trained GPT-2 model to perform sentence insertion. It then transforms it into a higher dimensional form. Then, clusters are formed by performing K-medoid analysis on this high dimensional case. The cluster centers representing the text semantic centers are selected as the extracted summary. Compared to inferential summarization, abstractive summarization is trained to create a summary by linking it with pre-created keywords.

Keywords are extracted from the text using a fine-tuned classification tool thanks to NLTK. To establish a semantic link between keywords and text, we have to eliminate certain groups in keywords. Keywords are symbols that fall into three different groups. After the keyword is created, the keywords are matched with the text and passed to the GPT-2 model for training.

After training, summary results are produced using the stochastic sampling method, which performs well in such studies. Results are compared by reading the review.

3.2 Model Architecture

Considering the success of the attention mechanism to convert input strings to output strings, the transformer architecture is used, so most state-of-the-art NLP models are the transformer architecture of choice.

The primary task in our research is to be able to summarize other articles using this transformer. We will do this with 3 sub-tasks, (1). pre-training over the data set, (2). fine-tuning on the same data set and (3) obtaining the summary.

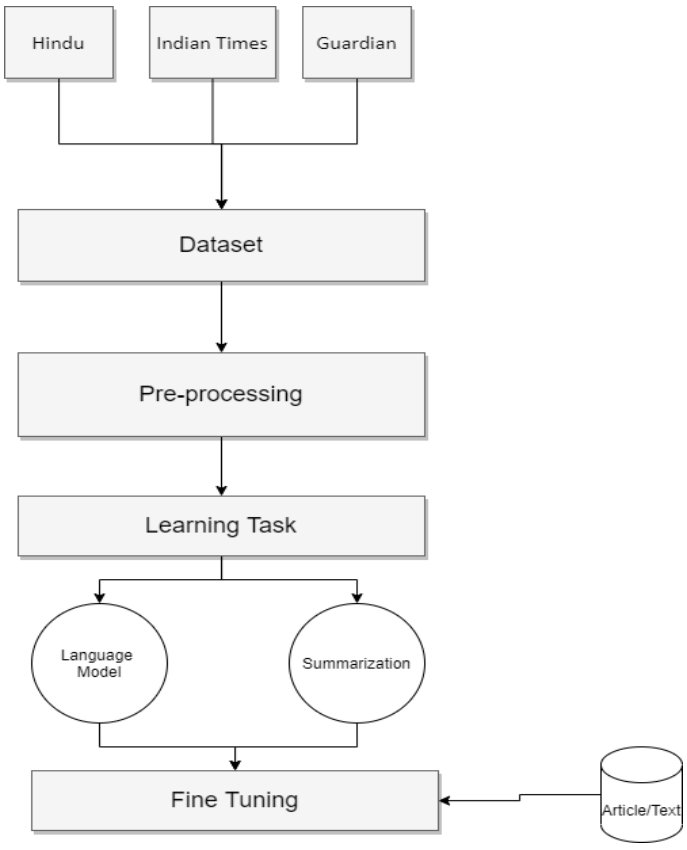

Figure 2: Workflow of tasks

Since we are going to use large data sets as a resource, we need to train models first. However, since the data set, we will use for training is from sources such as Hindu, Indian times and Guardian, certain parts of the data set will fail or the source text will not be consistent from time to time. We have to pre-process them before training and make them ready for training. Some data contain words in local languages rather than English, so we need to delete them completely.

Each training example consistent with the texts in our data set is found in our data, the article itself, and as keywords. Summary, on the other hand, are sentences that will emerge because of the relationship between the keywords and the article itself. The network is trained to predict a partial hash tag that uses three randomly generated split pieces of text that we created from the text as input. As we mentioned earlier, the network news articles we created are pre-trained to create a summary with keywords that match. However, there is no human-written article summary in our data set. When we accepted the article summary as the first paragraph of the article, we discarded it as we encountered many errors. However, there may be some issues with our training set.

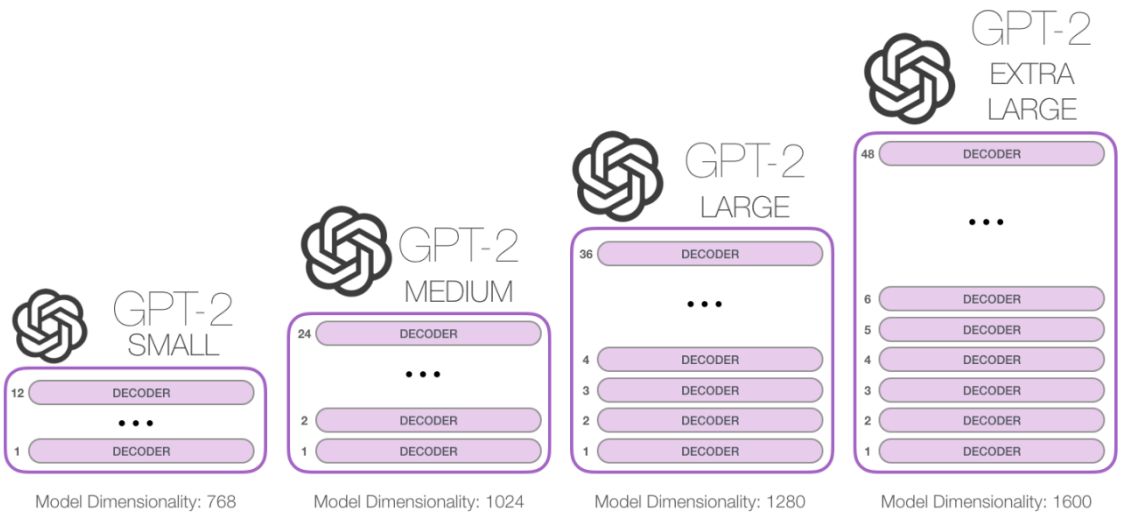

Figure 3: GPT-2 Sizes

(2) Using the pre-trained model in (1), we will fine-tune the network as per our request. We used transformer for this, mostly two types of transformer architectures are emphasized, transformer encoder and transformer decoder. We have considered a pre-trained encoder model of GPT-2 to generate summaries. Since we know the weight of the model to be made and especially the GPT-2 consumes a lot of GPUs, we preferred the length of 1024 coins by using the DistilGPT-2, which is the distil version. With DistilGPT-2, we created our transformer with 12 attention and 6 transformer decoder layers. We use all these from the Huggingface transformer package, the version we preferred was pytorch due to its ease of use.

3.2.1 Pre-Training with Transfer Learning

Pre-training underwent transfer learning. Cite learning attempts to accurately convey information from a news text to a summary.

There are two different areas of transfer learning, (1) source domain Ds and (2) destination domain Dt. A range consists of two parts; A feature space χ and a probability distribution $P(X)$, where X is data from the set

$X = \{x_1, x_1, x_1, x_1, \ldots., x_n\} \in x$. Besides, a domain consists of a task DT with two parts; a tag range Y and a specific data feature vector and a prediction function with a tag (xi, yi) combination input f.

To summarize, transfer learning aims to learn in the form of P (YT | XT) in a predetermined DT domain with information obtained from DS and TS. In our setting, we consider the domains of the model's parameters separately and train them independently. Considering the dataset we use, we are fixed to the DS receiving a resource. Here, the properties, verified words, source and summary domain may differ. While there are general words that should pass at an approximately comparable scale that should be domain-independent, there are also higher domain-specific words that depend on the document type located in such domain. For example, in our inclusive main topic, different words may be common, for example "NLP" or "GPT" will both be found under the natural language processing main topic. Thus, the marginal distribution will be different, so $P(Y_s|X_s), P(Y_t|X_t)$ can also be considered as frequency property deviation.

3.3 Training Strategy for Summary

We basically designed our GPT-2 model to train on 2 main tasks: 1st Language modelling (LM) task, 2nd Multiple choice (MC) prediction task.

Our LM task projects the hidden state onto the word embedding output layer. By applying cross-entropy loss to a text with the corresponding keyword, we get the LM loss value. For training, we label the beginning and end of the text with special markers. We used a special icon "summarize" to separate keywords and text so that the model can understand and do its summarizing task. The entire entry is filled with the fill indicator up to 1024 symbols, as we mentioned in DistilGPT, and all entries longer than 1024 symbols are truncated.

For the MC task, we added the hidden state of the token at the end of the text in the form of "end of text" for the probability score we will use in the future, this was actually done for a simple classification task. As a result of this, the cross-entropy loss is used to gain the MC loss. To generate the training dataset, we randomly selected 3 summary snippets that had nothing to do with keywords, called pseudo-distractors, and matched them with keywords to create a master group. Language modelling training tags are the symbol of the summary scrolled to the right with 1 coin. This is because, as we know from BERT, the GPT-2 also inherently declines automatically, and the nth token output is generated to the left from all previous n-1 coin entries.

The multiple-choice education tag is a tensor of a numeric i that specifies the i^{th} element, the correct keyword hash pair.

3.4 Idea

The idea behind this training strategy is this. Since GPT-2 generally aims to create text, it cannot directly analyze and extract a summary. However, our model was designed to be a self-attention. In other words, our model will evaluate the words before it for the words it will created/predicted, in our application, the words are found as tokens, that is, if it takes the context of the n-1 tokens found before n to predict the nth coin.

However, if we make predictions by constantly going back to the past, we applied the masking process in order to prevent the calculation of the information from the markers to the right of the current location, because more than the summary will be produced again. As we create a masked self-attention mechanism in this way, the sentences to be created/predicted will provide a unique summary.

We used special notation here to define the context "summarize", where the tokens to create will form the created summary. Our GPT-2 model learns this context hint from the data it receives from the fine-tuning.

Multiple loss training is used with the possibility of encouraging matching between the meaning in the keywords specified in our model and the text. Therewithal, the model stores the general contextual knowledge of the keywords so that at the end of the text the model can specify a summary from distractors. This multi-loss education stands out in the final language model training aimed at general language understanding.

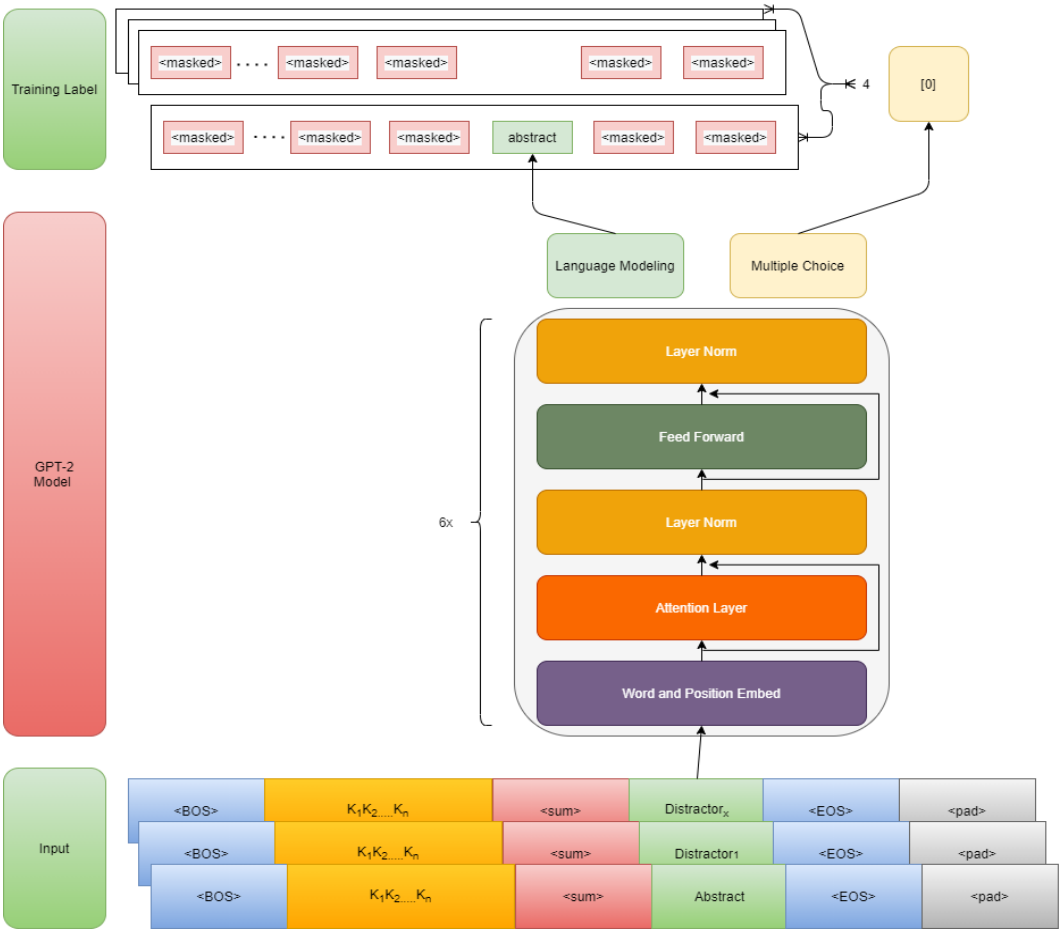

Figure 4: GPT-2's multi-loss training

Chapter 4: Our Experiments and Results

4.1 Model Training

The training of DistilGPT2, which we prefer because it requires less hardware, is carried out in GPU settings above Google COLAB, where the GPU Google offers us is NVIDIA K80. We trained up to 5 epochs in total. The training data set consists of 3246 training samples, each sample has 4 randomly split options.

The validation data set consists of 1572 samples, each with 4 randomly split options. Training parameters include batch size is 1 and 5 epoch gradient accumulation and learning rate $3e^{-5}$.

4.2 Results

Epoch	Trainer Results
Epoch = 1	iteration 100 - LM loss: 17.06 MC loss: 1.48 total loss: 35.61 report time: 360.3 iteration 200 - LM loss: 6.83 MC loss: 1.45 total loss: 15.10 report time: 445.7 iteration 300 - LM loss: 4.05 MC loss: 1.51 total loss: 9.61 report time: 531.0 iteration 400 - LM loss: 3.03 MC loss: 1.47 total loss: 7.53 report time: 616.4 iteration 500 - LM loss: 2.72 MC loss: 1.44 total loss: 6.87 report time: 701.8 iteration 600 - LM loss: 2.52 MC loss: 1.44 total loss: 6.47 report time: 787.3 iteration 700 - LM loss: 2.40 MC loss: 1.42 total loss: 6.23 report time: 872.5 iteration 800 - LM loss: 2.32 MC loss: 1.39 total loss: 6.03 report time: 957.8 iteration 900 - LM loss: 2.28 MC loss: 1.42 total loss: 5.98 report time: 1043.1 iteration 1000 - LM loss: 2.20 MC loss: 1.40 total loss: 5.79 report time: 1128.3 iteration 1100 - LM loss: 2.14 MC loss: 1.34 total loss: 5.61 report time: 1213.5 iteration 1200 - LM loss: 2.11 MC loss: 1.36 total loss: 5.58 report time: 1298.8 iteration 1300 - LM loss: 2.10 MC loss: 1.34 total loss: 5.54 report time: 1384.0 iteration 1400 - LM loss: 2.04 MC loss: 1.30 total loss: 5.39 report time: 1469.3 iteration 1500 - LM loss: 1.94 MC loss: 1.21 total loss: 5.10 report time: 1554.6 iteration 1600 - LM loss: 2.01 MC loss: 1.21 total loss: 5.22 report time: 1639.8 iteration 1700 - LM loss: 2.02 MC loss: 1.18 total loss: 5.22 report time: 1725.1 iteration 1800 - LM loss: 1.94 MC loss: 1.14 total loss: 5.02 report time: 1810.4 iteration 1900 - LM loss: 1.99 MC loss: 1.03 total loss: 5.01 report time: 1895.9 iteration 2000 - LM loss: 1.96 MC loss: 0.86 total loss: 4.79 report time: 1981.4 iteration 2100 - LM loss: 2.00 MC loss: 0.76 total loss: 4.76 report time: 2066.9 iteration 2200 - LM loss: 2.03 MC loss: 0.50 total loss: 4.56 report time: 2152.3 iteration 2300 - LM loss: 1.89 MC loss: 0.35 total loss: 4.14 report time: 2237.6 iteration 2400 - LM loss: 1.97 MC loss: 0.20 total loss: 4.14 report time: 2322.8 iteration 2500 - LM loss: 2.03 MC loss: 0.14 total loss: 4.19 report time: 2408.2 iteration 2600 - LM loss: 2.01 MC loss: 0.09 total loss: 4.11 report time: 2493.6 iteration 2700 - LM loss: 1.90 MC loss: 0.07 total loss: 3.87 report time: 2578.9 iteration 2800 - LM loss: 1.95 MC loss: 0.06 total loss: 3.96 report time: 2664.2 iteration 2900 - LM loss: 1.93 MC loss: 0.04 total loss: 3.90 report time: 2749.7 iteration 3000 - LM loss: 1.93 MC loss: 0.05 total loss: 3.92 report time: 2835.0 iteration 3100 - LM loss: 1.93 MC loss: 0.07 total loss: 3.94 report time: 2920.3 Epoch 1 - Avg lm_loss: 1.73 Avg mc_loss: 0.03 Avg total_loss: 3.48

Epoch =5	Trainer Results - iteration 100 - LM loss: 1.86 MC loss: 0.05 total loss: 3.76 report time: 5254.0 Trainer Results - iteration 200 - LM loss: 1.99 MC loss: 0.03 total loss: 4.01 report time: 5339.2 Trainer Results - iteration 300 - LM loss: 1.96 MC loss: 0.06 total loss: 3.97 report time: 5424.6 Trainer Results - iteration 400 - LM loss: 1.96 MC loss: 0.04 total loss: 3.96 report time: 5509.9 Trainer Results - iteration 500 - LM loss: 1.91 MC loss: 0.05 total loss: 3.87 report time: 5595.3 Trainer Results - iteration 600 - LM loss: 1.89 MC loss: 0.08 total loss: 3.86 report time: 5680.5 Trainer Results - iteration 700 - LM loss: 1.90 MC loss: 0.03 total loss: 3.83 report time: 5765.8 Trainer Results - iteration 800 - LM loss: 1.95 MC loss: 0.07 total loss: 3.96 report time: 5851.0 Trainer Results - iteration 900 - LM loss: 2.50 MC loss: 0.06 total loss: 5.07 report time: 5936.2 Trainer Results - iteration 1000 - LM loss: 1.97 MC loss: 0.04 total loss: 3.98 report time: 6021.4 Trainer Results - iteration 1100 - LM loss: 1.91 MC loss: 0.06 total loss: 3.87 report time: 6106.7 Trainer Results - iteration 1200 - LM loss: 1.89 MC loss: 0.03 total loss: 3.81 report time: 6192.0 Trainer Results - iteration 1300 - LM loss: 1.98 MC loss: 0.04 total loss: 4.00 report time: 6277.2 Trainer Results - iteration 1400 - LM loss: 1.92 MC loss: 0.05 total loss: 3.89 report time: 6362.5 Trainer Results - iteration 1500 - LM loss: 1.92 MC loss: 0.03 total loss: 3.86 report time: 6447.8 Trainer Results - iteration 1600 - LM loss: 1.91 MC loss: 0.03 total loss: 3.85 report time: 6533.1 Trainer Results - iteration 1700 - LM loss: 1.90 MC loss: 0.04 total loss: 3.85 report time: 6618.3 Trainer Results - iteration 1800 - LM loss: 1.90 MC loss: 0.05 total loss: 3.86 report time: 6703.7 Trainer Results - iteration 1900 - LM loss: 1.93 MC loss: 0.04 total loss: 3.89 report time: 6789.0

	Trainer Results - iteration 2000 - LM loss: 1.88 MC loss: 0.04 total loss: 3.80 report time: 6874.2 Trainer Results - iteration 2100 - LM loss: 1.87 MC loss: 0.04 total loss: 3.78 report time: 6959.4 Trainer Results - iteration 2200 - LM loss: 1.91 MC loss: 0.08 total loss: 3.90 report time: 7044.5 Trainer Results - iteration 2300 - LM loss: 1.90 MC loss: 0.06 total loss: 3.86 report time: 7129.8 Trainer Results - iteration 2400 - LM loss: 1.90 MC loss: 0.04 total loss: 3.84 report time: 7215.2 Trainer Results - iteration 2500 - LM loss: 1.96 MC loss: 0.05 total loss: 3.96 report time: 7300.4 Trainer Results - iteration 2600 - LM loss: 1.97 MC loss: 0.06 total loss: 4.00 report time: 7385.6 Trainer Results - iteration 2700 - LM loss: 1.93 MC loss: 0.06 total loss: 3.92 report time: 7471.0 Trainer Results - iteration 2800 - LM loss: 1.94 MC loss: 0.04 total loss: 3.92 report time: 7556.5 Trainer Results - iteration 2900 - LM loss: 1.87 MC loss: 0.03 total loss: 3.78 report time: 7641.8 Trainer Results - iteration 3000 - LM loss: 1.89 MC loss: 0.03 total loss: 3.81 report time: 7727.1 Trainer Results - iteration 3100 - LM loss: 1.92 MC loss: 0.04 total loss: 3.89 report time: 7812.5 Validation Results - Epoch 1 - Avg lm_loss: 1.74 Avg mc_loss: 0.03 Avg total_loss: 3.52 Trainer Results - iteration 3200 - LM loss: 1.85 MC loss: 0.05 total loss: 3.76 report time: 9494.8 Trainer Results - iteration 3300 - LM loss: 1.91 MC loss: 0.04 total loss: 3.86 report time: 9580.3 Trainer Results - iteration 3400 - LM loss: 1.95 MC loss: 0.04 total loss: 3.94 report time: 9665.6 Trainer Results - iteration 3500 - LM loss: 2.00 MC loss: 0.06 total loss: 4.06 report time: 9751.0 Trainer Results - iteration 3600 - LM loss: 1.98 MC loss: 0.06 total loss: 4.02 report time: 9836.4 Trainer Results - iteration 3700 - LM loss: 1.95 MC loss: 0.04 total loss: 3.95 report time: 9921.6

	Trainer Results - iteration 3800 - LM loss: 1.90 MC loss: 0.04 total loss: 3.84 report time: 10006.8 Trainer Results - iteration 3900 - LM loss: 2.00 MC loss: 0.04 total loss: 4.03 report time: 10092.1 Trainer Results - iteration 4000 - LM loss: 1.90 MC loss: 0.04 total loss: 3.84 report time: 10177.4 Trainer Results - iteration 4100 - LM loss: 1.90 MC loss: 0.03 total loss: 3.83 report time: 10262.9 Trainer Results - iteration 4200 - LM loss: 1.93 MC loss: 0.04 total loss: 3.91 report time: 10348.3 Trainer Results - iteration 4300 - LM loss: 1.96 MC loss: 0.05 total loss: 3.97 report time: 10433.8 Trainer Results - iteration 4400 - LM loss: 1.93 MC loss: 0.05 total loss: 3.92 report time: 10519.1 Trainer Results - iteration 4500 - LM loss: 1.93 MC loss: 0.05 total loss: 3.91 report time: 10604.4 Trainer Results - iteration 4600 - LM loss: 1.89 MC loss: 0.03 total loss: 3.81 report time: 10689.7 Trainer Results - iteration 4700 - LM loss: 1.96 MC loss: 0.04 total loss: 3.95 report time: 10775.2 Trainer Results - iteration 4800 - LM loss: 1.95 MC loss: 0.03 total loss: 3.93 report time: 10860.6 Trainer Results - iteration 4900 - LM loss: 1.91 MC loss: 0.04 total loss: 3.86 report time: 10946.1 Trainer Results - iteration 5000 - LM loss: 1.97 MC loss: 0.04 total loss: 3.98 report time: 11031.6 Trainer Results - iteration 5100 - LM loss: 1.92 MC loss: 0.05 total loss: 3.89 report time: 11116.9 Trainer Results - iteration 5200 - LM loss: 1.92 MC loss: 0.04 total loss: 3.89 report time: 11202.2 Trainer Results - iteration 5300 - LM loss: 1.92 MC loss: 0.04 total loss: 3.89 report time: 11287.6 Trainer Results - iteration 5400 - LM loss: 1.93 MC loss: 0.04 total loss: 3.90 report time: 11373.0 Trainer Results - iteration 5500 - LM loss: 1.91 MC loss: 0.04 total loss: 3.86 report time: 11458.4 Trainer Results - iteration 5600 - LM loss: 1.97 MC loss: 0.06 total loss: 4.01 report time: 11543.6

	Trainer Results - iteration 5700 - LM loss: 1.94 MC loss: 0.04 total loss: 3.91 report time: 11628.9 Trainer Results - iteration 5800 - LM loss: 1.94 MC loss: 0.04 total loss: 3.92 report time: 11714.2 Trainer Results - iteration 5900 - LM loss: 1.91 MC loss: 0.06 total loss: 3.88 report time: 11799.4 Trainer Results - iteration 6000 - LM loss: 1.93 MC loss: 0.04 total loss: 3.90 report time: 11884.7 Trainer Results - iteration 6100 - LM loss: 1.88 MC loss: 0.04 total loss: 3.81 report time: 11970.0 Trainer Results - iteration 6200 - LM loss: 1.93 MC loss: 0.04 total loss: 3.90 report time: 12055.4 Validation Results - Epoch 2 - Avg lm_loss: 1.71 Avg mc_loss: 0.02 Avg total_loss: 3.44 Trainer Results - iteration 6300 - LM loss: 1.78 MC loss: 0.02 total loss: 3.58 report time: 13736.2 Trainer Results - iteration 6400 - LM loss: 1.90 MC loss: 0.06 total loss: 3.86 report time: 13821.3 Trainer Results - iteration 6500 - LM loss: 1.90 MC loss: 0.04 total loss: 3.84 report time: 13906.6 Trainer Results - iteration 6600 - LM loss: 1.90 MC loss: 0.05 total loss: 3.85 report time: 13991.9 Trainer Results - iteration 6700 - LM loss: 1.92 MC loss: 0.04 total loss: 3.88 report time: 14077.2 Trainer Results - iteration 6800 - LM loss: 1.92 MC loss: 0.04 total loss: 3.88 report time: 14162.4 Trainer Results - iteration 6900 - LM loss: 1.93 MC loss: 0.05 total loss: 3.91 report time: 14247.5 Trainer Results - iteration 7000 - LM loss: 1.91 MC loss: 0.04 total loss: 3.86 report time: 14332.6 Trainer Results - iteration 7100 - LM loss: 1.91 MC loss: 0.06 total loss: 3.89 report time: 14418.0 Trainer Results - iteration 7200 - LM loss: 1.93 MC loss: 0.04 total loss: 3.90 report time: 14503.2 Trainer Results - iteration 7300 - LM loss: 1.96 MC loss: 0.04 total loss: 3.95 report time: 14588.4 Trainer Results - iteration 7400 - LM loss: 1.92 MC loss: 0.03 total loss: 3.88 report time: 14673.6

	Trainer Results - iteration 7500 - LM loss: 1.94 MC loss: 0.03 total loss: 3.91 report time: 14758.8 Trainer Results - iteration 7600 - LM loss: 1.95 MC loss: 0.05 total loss: 3.96 report time: 14843.8 Trainer Results - iteration 7700 - LM loss: 1.83 MC loss: 0.04 total loss: 3.70 report time: 14929.0 Trainer Results - iteration 7800 - LM loss: 1.96 MC loss: 0.05 total loss: 3.97 report time: 15014.2 Trainer Results - iteration 7900 - LM loss: 2.34 MC loss: 0.05 total loss: 4.74 report time: 15099.6 Trainer Results - iteration 8000 - LM loss: 1.96 MC loss: 0.04 total loss: 3.96 report time: 15184.7 Trainer Results - iteration 8100 - LM loss: 1.94 MC loss: 0.04 total loss: 3.92 report time: 15269.9 Trainer Results - iteration 8200 - LM loss: 1.97 MC loss: 0.03 total loss: 3.97 report time: 15355.3 Trainer Results - iteration 8300 - LM loss: 1.95 MC loss: 0.03 total loss: 3.93 report time: 15440.6 Trainer Results - iteration 8400 - LM loss: 1.84 MC loss: 0.05 total loss: 3.73 report time: 15525.8 Trainer Results - iteration 8500 - LM loss: 1.85 MC loss: 0.05 total loss: 3.76 report time: 15611.0 Trainer Results - iteration 8600 - LM loss: 1.95 MC loss: 0.06 total loss: 3.96 report time: 15696.1 Trainer Results - iteration 8700 - LM loss: 1.94 MC loss: 0.03 total loss: 3.91 report time: 15781.4 Trainer Results - iteration 8800 - LM loss: 1.93 MC loss: 0.05 total loss: 3.91 report time: 15866.6 Trainer Results - iteration 8900 - LM loss: 1.92 MC loss: 0.08 total loss: 3.93 report time: 15951.9 Trainer Results - iteration 9000 - LM loss: 1.94 MC loss: 0.04 total loss: 3.92 report time: 16037.3 Trainer Results - iteration 9100 - LM loss: 1.90 MC loss: 0.04 total loss: 3.84 report time: 16122.4 Trainer Results - iteration 9200 - LM loss: 1.99 MC loss: 0.06 total loss: 4.05 report time: 16207.6 Trainer Results - iteration 9300 - LM loss: 1.91 MC loss: 0.04 total loss: 3.86 report time: 16292.6

	Trainer Results - iteration 9400 - LM loss: 1.95 MC loss: 0.03 total loss: 3.92 report time: 16378.0 Validation Results - Epoch 3 - Avg lm_loss: 1.72 Avg mc_loss: 0.05 Avg total_loss: 3.50

Table 1 : Model Training Table

4.3 Sequence Generation

The output generated by modelling GPT2 is a tensor, containing line length and word size. This is the tensor of a probability distribution that contains all the words before SoftMax. To create a text string from this output, we sample words from this distribution by word. To derive the nth word, we consider the conditional probability of n-1 words found before it.

$$P(X) = \prod_{i=1} P(X_i | x_1, x_2, x_3 \ldots . x_{i-1})$$

First, before sampling, we can apply a scaling factor called temperature (t), the probability of reshaping the skew probability distribution before softmax (u).

$$P(X) = \frac{\exp\left({}^{u}/_{t}\right)}{\sum l^{*} \exp\left({}^{u_l^*}/_{t}\right)}$$

The high temperature caused by low probability words tends to distort the dispersion; it tends to dissipate with low temperature, leaning towards high probability words. The result is a controversy between opting for manufacturing accuracy at the expense of word diversity. Second, to select previous words in

the array, we need to make a conditional selection, we use a stochastic sampling method called top-p sampling in the probability distribution of these words.

The smallest set of candidate words to be considered by the rule of top-p sampling is that it is greater than the cumulative conditional probability.

$$\sum_{X} P(x|x_{1:i-1}) \geq P$$

In addition, by considering the individual words in the texts, we took these words into consideration and did not want the model to concentrate too much on low probability words and planned to avoid them. Therefore, we had to limit the number of candidate words to be considered, and we decided to limit it to k words. As a result of our experiments, we tested some of the sampling parameters and determined certain values. We found a value of 1 for temperature, 50 for word count, and p = 0.8 giving a reasonable result.

Gold Summary	**Keywords**	**Generated Summary**
The German Johannes Gutenberg introduced printing in Europe. His invention had a decisive contribution in spread of mass-learning and in	['A text about Johannes Gutenberg, johannes, gutenberg, german, goldsmith, publisher, introduced, printing, europe, introduction, mechanical, movable, type, printing, europe, started, printing,	Johannes Gutenberg german publisher introduced printing europe and introduced prints of europe in the first half of his life, and rewritten his reprinted and reprinted. comprepared for printing the european revolution. widely regarded by the important event of modern

building the basis of the modern society. **Gutenberg major invention was a practical system permitting the mass production of printed books. The printed books allowed open circulation of information, and prepared the evolution of society from to the contemporary knowledge-based economy.**	revolution, widely, regarded, important, event, modern, period, played, key, role, scientific, revolution, laid, basis, modern, economy, spread, learning, masses, gutenberg, many, contributions, printing, invention, process, movable, type, use, ink, printing, books, adjustable, molds, use, wooden, printing, press, truly, epochal, invention, combination, elements, practical, system, allowed, mass, production, printed, books, economically, viable, printers, readers, renaissance, europe, arrival, mechanical, movable, type, printing, introduced, era, mass, communication, permanently, altered, structure, society, relatively, unrestricted, circulation, revolutionary, borders, captured, masses, reformation, sharp, increase, literacy, broke, monopoly, literate, elite, education, learning,	period played by key role scientific revolution and laid basis modern economy spread,all of the contribution actions, printed in the printing of europes,started printing revolution widely and regarded the important.

	bolstered, emerging, middle,class']	
		Johannes Gutenberg introduced the printing press in europe. the invention of the mass spread of learning and made a decisive contribution in building the foundations of modern society. gutenberg's great invention was a practical system that allows the mass production of printed books. printed books permitted the free circulation of knowledge and prepared the evolution of society for the contemporary knowledge-based economy.

Table 2: Generated Summary

Chapter 5: Analysis

ROGUE Score

ROGUE stands for Recall Focused Understudy for Gisting Assessment. It is a measure for measuring the quality of a summary produced by the summarizer we have produced for this thesis against a man-made summary. There are several types of ROUGE and the most common of these are listed below:

1. **ROUGE-n**: This metric is a recall-based measure and is based on an n-gram comparison. A set of n-gram reference summaries, which are two and three of which are often preferred as n, and the summary automatically generated from the candidate summary. "P" refers to the common n-gram number between the candidate and the reference summary, and "q" refers to the n-gram number extracted from the reference summary only. The score is calculated as follows:

$$ROUGE - n = p/q$$

2. **ROUGE-L**: This measure uses the concept that is the longest common substring between two text strings (LCS). The hunch is that the longer the LCS between two summary sentences, the more similar they are. While this metric is more flexible than the previous one, it has a number of drawbacks as all n-grams must be structurally sequential.

3. **ROUGE-SU**: This metric jump is called bi-gram and uni-gram ROUGE, and it considers bi-grams and uni-grams. This measure allows words to be added between the first and last words of bi-grams, so they do not have to be sequential word strings.

TEXT	**Rouge 1**			**Rouge 2**			**Rouge L**		
	Precision	Recall	Fmeasure	Precision	Recall	Fmeasure	Precision	Recall	Fmeasure
Gutenberg Epoch = 1	0.40	0.46	0.43	0.071	0.080	0.075	0.25	0.28	0.26
Gutenberg Epoch = 5	0.82	0.84	0.83	0.55	0.56	0.56	0.68	0.69	0.69

Table 3: ROUGE score for Gutenberg Text

TEXT	**Rouge 1**			**Rouge 2**			**Rouge L**		
	Precision	Recall	Fmeasure	Precision	Recall	Fmeasure	Precision	Recall	Fmeasure
CNN News Epoch = 1	0.47	0.34	0.39	0.21	0.15	0.17	0.23	0.17	0.19

CNN News Epoch = 5	0.81	0.74	0.77	0.5	0.46	0.47	0.62	0.57	0.6

Table 4: ROUGE score for CNN News Text

Based on the ROUGE score, we see the accuracy of using a small but news-based dataset in our gpt training. Due to the much smaller size compared to many text data, pre-training required much less iteration and as a result, it received much faster training. In addition, thanks to the diversity of the news in the text creation part, it was possible to create a summary with appropriate patterns without history or story. Some academic studies have stated that the more relevant those used as resources are with the task, the easier it is to learn transfer. For example: If we wanted to create an academic essay summarizer, it would be more correct for us to use a source of academic essays rather than news, but we chose the news dataset because we wanted an above average, readable and useful summarizer in a simple framework. In addition, it may be possible to gradually increase the relationship between training time and performance to more data at a certain point, so that a richer summary extraction process can be made with growing and diversifying news. However, since we have not yet conducted an experiment to test this hypothesis, this statement should be interpreted with caution for the time being and should be treated with caution.

Chapter 6: Conclusion and Future Steps

Abstractive summarization is still a major challenge for natural language processing. Even more, when this task is applied to a site-specific corpus that is different from pre-training, which is highly technical or contains little training material.

The News Dataset Challenge exemplifies all the challenges mentioned above. However, we have used a successful pre-trained model such as GPT-2, demonstrating that summarizing with text-to-text, multiple-loss training strategy can be done on this model, but our result is interpretable and logical, although not close to the level of human-level performance we want. First of all, we think that as we increase the content of our data set with current news, we can take steps to increase our success in education, and we think that we can make better use of our model by comparing with new summaries to be created and working on fine-tuning. Thus, it should make it more accurate in terms of the idea (using keywords) that we used in our model and the ability to summarize successfully.

Looking back, we think the keyword-building phrase is flexible and can be improved significantly without much investment. We think that if we concentrate the keywords on verbs, nouns, and adjectives, we could be closer to success. If we could add more information to the keyword, such as the adjective part, we could also investigate how much more accuracy could be gained. Apart from

this, it can be done by clustering similar news, adding random common words to keyword sets, or more data increase. This will generate many more keyword digest pairs than exist. Finally, we think another strategy can be tried to extract keywords. For example, a token classification model can be fine-tuned to extract keywords specific to the type of news, such as political/political, economic, from the topics of the news.

There is a great deficiency in the evaluation process of abstract summaries, and besides the development period of this summarizing idea, evaluation methods should be developed / discovered in this regard. In the field of summarization, ROUGE scores are generally preferred, but this scoring system represents words that overlap, rather than the semantic integrity or readability of the summaries. The abstractive models we have made are from inferential models, so they will score low.

The well-readable summaries that GPT has created with limited information are not reflected by the ROUGE scores, clearly demonstrating that other relevant scores or assessment methods are required in these aspects in the future. One possible idea would be to combine these two ideas to reward semantically similarity, punishing if the word pairs present in the abstract occur directly in the source text.

Finally, we can consider resource assets. We think that the success of the model can be increased if there are resources that we can use more for more intense computing processes. Since our processing power is limited, we had to keep some values small. For example, with the Tesla K-80 GPU that COLAB has presented to us, training can only be done in the DistilGPT2 version with batch

size = 1 due to the GPU's memory limit. If the available computing power allows, the result is likely to benefit greatly from using the larger GPT-2 version and more training.

Considering what we have done because of the project, we hope to have a successful text summarization approach in the future. Like the dataset we use, it can help the community of many different topics and needs keep pace with the rapidly growing literature and can benefit people for their own business or time.

We think that making GPT-3 available for everyone will be a great benefit in this regard. For example, with the special use permission Microsoft has received for GPT-3, we can see innovations in this area in a short time.

Chapter 7: Acknowledgements

In particular, I would like to thank my supervisor Professor Andrea Belli, who is Head of R&D at expert.ai, whom I met during the Text Mining lesson at Università Cattolica Del Sacro Cuore, for valuable feedback and guidance during the dissertation period. I am especially grateful for the time and commitment spent in supervising me and my thesis through valuable meetings. I am also grateful for the flexibility and freedom given to me in the research as it brought me different perspectives and adapted to the challenges of the Pandemic time.

Chapter 8: Bibliography

I. "All You Need Is LSD." *All You Need Is LSD*, 2018, doi:10.5040/9781350101272.00000005.

II. Allahyari, Mehdi, et al. "Text Summarization Techniques: A Brief Survey." *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, 2017, doi:10.14569/ijacsa.2017.081052.

III. Alshaina, S, et al. "Multi-Document Abstractive Summarization Based on Predicate Argument Structure." *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2017, doi:10.1109/spices.2017.8091339.

IV. Anh, Dang Trung, and Nguyen Thi Thu Trang. "Abstractive Text Summarization Using Pointer-Generator Networks With Pre-Trained Word Embedding." *Proceedings of the Tenth International Symposium on Information and Communication Technology - SoICT 2019*, 2019, doi:10.1145/3368926.3369728.

V. "Bidirectional Molecule Generation with Recurrent Neural Networks." doi:10.1021/acs.jcim.9b00943.s001.

VI. Chen, Yen-Chun, and Mohit Bansal. "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, doi:10.18653/v1/p18-1063.

VII. Cheng, Jianpeng, and Mirella Lapata. "Neural Summarization by Extracting Sentences and Words." *Proceedings of the 54th Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), 2016, doi:10.18653/v1/p16-1046.

VIII. Chopra, Sumit, et al. “Abstractive Sentence Summarization with Attentive Recurrent Neural Networks.” *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, doi:10.18653/v1/n16-1012.

IX. Cintas, Celia, et al. “Towards Neural Abstractive Clinical Trial Text Summarization with Sequence to Sequence Models.” *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, 2019, doi:10.1109/ichi.2019.8904526.

X. Egonmwan, Elozino, and Yllias Chali. “Transformer-Based Model for Single Documents Neural Summarization.” *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 2019, doi:10.18653/v1/d19-5607.

XI. Gehrmann, Sebastian, et al. “Bottom-Up Abstractive Summarization.” *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, doi:10.18653/v1/d18-1443.

XII. Ghalehtaki, Razieh Abbasi, et al. “Evaluating Preprocessing by Turing Machine in Text Categorization.” *2014 Iranian Conference on Intelligent Systems (ICIS)*, 2014, doi:10.1109/iraniancis.2014.6802540.

XIII. *Introduction to Algorithms*. MIT Press, 2001.

XIV. *Introduction to Statistical Relational Learning*. MIT Press, 2019.

XV. Karakoc, Enise, and Burcu Yilmaz. “Deep Learning Based Abstractive Turkish News Summarization.” *2019 27th Signal Processing and Communications Applications Conference (SIU)*, 2019, doi:10.1109/siu.2019.8806510.

XVI. Kim, Seungwon. “Using Pre-Trained Transformer for Better Lay Summarization.” *Proceedings of the First Workshop on Scholarly Document Processing*, 2020, doi:10.18653/v1/2020.sdp-1.38.

XVII. Klein, Tassilo, and Moin Nabi. “Attention Is (Not) All You Need for Commonsense Reasoning.” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, doi:10.18653/v1/p19-1477.

XVIII. Koehn, Philipp, and Rebecca Knowles. “Six Challenges for Neural Machine Translation.” *Proceedings of the First Workshop on Neural Machine Translation*, 2017, doi:10.18653/v1/w17-3204.

XIX. Kozma, Robert, et al. *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Academic Press, an Imprint of Elsevier, 2019.

XX. Lim, Hyun-Il. “A Study on Dropout Techniques to Reduce Overfitting in Deep Neural Networks.” *Lecture Notes in Electrical Engineering Advanced Multimedia and Ubiquitous Engineering*, 2020, pp. 133–139., doi:10.1007/978-981-15-9309-3_20.

XXI. Liu, Shuang. “Fourier Neural Network for Machine Learning.” *2013 International Conference on Machine Learning and Cybernetics*, 2013, doi:10.1109/icmlc.2013.6890482.

XXII. Mann, William C. “Text Generation: The Problem of Text Structure.” *Natural Language Generation Systems*, 1988, pp. 47–68., doi:10.1007/978-1-4612-3846-1_2.

XXIII. Melamud, Oren, et al. “context2vec: Learning Generic Context Embedding with Bidirectional LSTM.” *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, doi:10.18653/v1/k16-1006.

XXIV. Mosa, Mohamed Atef. “Data Text Mining Based on Swarm Intelligence Techniques.” *Trends and Applications of Text Summarization Techniques Advances in Data Mining and Database Management*, 2020, pp. 88–124., doi:10.4018/978-1-5225-9373-7.ch004.

XXV. Nallapati, Ramesh, et al. “Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond.” *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016, doi:10.18653/v1/k16-1028.

XXVI. "Neural Information Processing Systems." *The Deep Learning Revolution*, 2018, doi:10.7551/mitpress/11474.003.0014.

XXVII. OpenAI. "About OpenAI." *OpenAI*, OpenAI, 2 Sept. 2020, openai.com/about/.

XXVIII. *OpenNMT*, opennmt.net/.

XXIX. "Overview." *CoreNLP*, stanfordnlp.github.io/CoreNLP.

XXX. Pattnaik, Sagarika, and Ajit Kumar Nayak. "Summarization of Odia Text Document Using Cosine Similarity and Clustering." *2019 International Conference on Applied Machine Learning (ICAML)*, 2019, doi:10.1109/icaml48257.2019.00035.

XXXI. "Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies." 2016, doi:10.18653/v1/n16-1.

XXXII. Qu, Yuanbin, et al. "A Text Generation and Prediction System: Pre-Training on New Corpora Using BERT and GPT-2." *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2020, doi:10.1109/iceiec49280.2020.9152352.

XXXIII. Radev, Dragomir, and James Pustejovsky. *Puzzles in Logic, Languages and Computation The Red Book*. Springer Berlin, 2015.

XXXIV. "Related Work in Text Generation." *Systemic Text Generation as Problem Solving*, 1988, pp. 133–145., doi:10.1017/cbo9780511665646.009.

XXXV. Rush, Alexander M., et al. "A Neural Attention Model for Abstractive Sentence Summarization." *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, doi:10.18653/v1/d15-1044.

XXXVI. Saito, K., and R. Nakano. "Extracting Characteristic Words of Text Using Neural Networks." *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, doi:10.1109/ijcnn.2004.1380154.

XXXVII. Sak, Haşim, et al. “Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping.” *Interspeech 2017*, 2017, doi:10.21437/interspeech.2017-1705.

XXXVIII. See, Abigail, et al. “Get To The Point: Summarization with Pointer-Generator Networks.” *Proceedings of the 55th Annual Meeting of the Association For Computational Linguistics (Volume 1: Long Papers)*, 2017, doi:10.18653/v1/p17-1099.

XXXIX. “Single Document Text Summarization Using Random Indexing And Neural Networks.” *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, 2010, doi:10.5220/0003066601710176.

XL. Solaiman, Irene. “GPT-2: 1.5B Release.” *OpenAI*, OpenAI, 4 Sept. 2020, openai.com/blog/gpt-2-1-5b-release/.

XLI. Song, Shengli, et al. “Abstractive Text Summarization Using LSTM-CNN Based Deep Learning.” *Multimedia Tools and Applications*, vol. 78, no. 1, 2018, pp. 857–875., doi:10.1007/s11042-018-5749-3.

XLII. “Text Generation.” *Text Knowledge and Object Knowledge*, doi:10.5040/9781474285407.ch-003.

XLIII. Torres-Moreno, Juan-Manuel. “Automatic Text Summarization: Some Important Concepts.” *Automatic Text Summarization*, 2014, pp. 23–52., doi:10.1002/9781119004752.ch2.

XLIV. Torres-Moreno, Juan-Manuel. “Evaluating Document Summaries.” *Automatic Text Summarization*, 2014, pp. 243–273., doi:10.1002/9781119004752.ch8.

XLV. Vig, Jesse. “A Multiscale Visualization of Attention in the Transformer Model.” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2019, doi:10.18653/v1/p19-3007.

XLVI. Weiss, Karl, et al. “A Survey of Transfer Learning.” *Journal of Big Data*, vol. 3, no. 1, 2016, doi:10.1186/s40537-016-0043-6.

XLVII. Wong, Kam-Fai, et al. “Extractive Summarization Using Supervised and Semi-Supervised Learning.” *Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08*, 2008, doi:10.3115/1599081.1599205.

XLVIII. Yang, Yinfei, et al. “Multilingual Universal Sentence Encoder for Semantic Retrieval.” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2020, doi:10.18653/v1/2020.acl-demos.12.

XLIX. Zhang, Haoyu, et al. “Pretraining-Based Natural Language Generation for Text Summarization.” *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019, doi:10.18653/v1/k19-1074.

L. Zhang, Jiajun, et al. “Abstractive Cross-Language Summarization via Translation Model Enhanced Predicate Argument Structure Fusing.” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 10, 2016, pp. 1842–1853., doi:10.1109/taslp.2016.2586608.

LI. Zhang, Xingxing, et al. “Neural Latent Extractive Document Summarization.” *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, doi:10.18653/v1/d18-1088.

LII. *Deeplearning.ai*, https://www.deeplearning.ai/program/natural-language-processing-specialization/

Appendix B: Generated Summary Samples

Sample 1

	Text	Johannes Gutenberg (1398 – 1468) was a German goldsmith and publisher who introduced printing to Europe. His introduction of mechanical movable type printing to Europe started the Printing Revolution and is widely regarded as the most important event of the modern period. It played a key role in the scientific revolution and laid the basis for the modern knowledge-based economy and the spread of learning to the masses. Gutenberg many contributions to printing are: the invention of a process for mass-producing movable type, the use of oil-based ink for printing books, adjustable molds, and the use of a wooden printing press. His truly epochal invention was the combination of these elements into a practical system that allowed the mass production of printed books and was economically viable for printers and readers alike. In Renaissance Europe, the arrival of mechanical movable type printing introduced the era of mass communication which permanently altered the structure of society. The relatively unrestricted circulation of information—including revolutionary ideas—transcended borders, and captured the masses in the Reformation. The sharp increase in literacy broke the monopoly of the literate elite on education and learning and bolstered the emerging middle class. From: http://explainwell.org/index.php/table-of-contents-synthesize-text/examples-of-summaries/

	Summary of the Text (Gold)	The German Johannes Gutenberg introduced printing in Europe. His invention had a decisive contribution in spread of mass-learning and in building the basis of the modern society. Gutenberg major invention was a practical system permitting the mass production of printed books. The printed books allowed open circulation of information, and prepared the evolution of society from to the contemporary knowledge-based economy.
Epoch =1	**Abstractive Summary**	Johannes Gutenberg german publisher introduced printing europe and introduced prints of europe in the first half of his life, and rewritten his reprinted and reprinted. comprepared for printing the european revolution. widely regarded by the important event of modern period played by key role scientific revolution and laid basis modern economy spread,all of the contribution actions, printed in the printing of europes,started printing revolution widely and regarded the important.
Epoch = 5	**Abstractive Summary**	Johannes Gutenberg introduced the printing press in europe. the invention of the mass spread of learning and made a decisive contribution in building the foundations of modern society. gutenberg's great invention was a practical system that allows the mass production of printed books. printed books permitted the free circulation of knowledge and prepared the evolution of society for the contemporary knowledge-based economy.

Sample 2

	Text	Hong Kong six people were hurt after an explosion at a controversial chemical plant in china 's southeastern fujian province sparked a huge fire, provincial authorities told state media . the plant, located in zhangzhou city , produces paraxylene -lrbpx-rrb- , a reportedly carcinogenic chemical used in the production of polyester films and fabrics .the blast occurred at an oil storage facility monday night after an oil leak , though local media has not reported any toxic chemical spill. From: CNN Dataset
Epoch = 1	**Summary of the Text (Gold)**	a blast rocks a chemical plant in china 's southeastern fujian province for the second time in two years .six were injured after the explosion and are being hospitalized. the explosion was triggered by an oil leak , though local media has not reported any toxic chemical spills.
	Abstractive Summary	an explosion occurred at an oil storage facility on monday night. six people were injured after the explosion at the chemical factory in fujian province. provincial officials say the explosion caused a huge fire..
Epoch = 5	**Abstractive Summary**	for the second time, an explosion occurred at a chemical factory in china's fujian province. six people were injured and hospitalized after the blast. the explosion was triggered by an oil spill, although local media did not report any toxic chemical spills.