Human-Centered Financial Summarization

Author: Tatiana PASSALI  
Supervisor: Dr. Grigorios TSOUMAKAS

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Science

March 22, 2021
Abstract

Automatic summarization has seen a rapid growth in recent years, allowing for efficient handling and processing of the huge amount of documents, which are available on the Web. There is also an increasing growth of domain-specific summarization models which use knowledge of the particular domain terminology e.g. medical reports, scientific publications, financial and company filings etc. Despite the success of these models, the majority of them are isolated from the users, not allowing any kind of useful feedback from them during the summarization process as well as not providing any meaningful explanations about the final predictions to them. In this thesis, we focus on financial summarization and we contribute a new dataset, which consists explicitly of financial newswire articles from economic and business categories via Bloomberg’s website. We also propose a financial human-centered summarization model, utilizing the implementation of an existing state-of-the-art transformer model: PEGASUS. We approach the task of summarization from a human-centered aspect, providing an interpretation analysis of the complex structure of the existing neural model, especially by examining the transparency of a widely used attention mechanism, namely self-attention. The user has an active role in our summarization system by selecting among different diverse hypotheses during inference, improving this way the quality of our collected dataset with human-approved summaries. We significantly improve the performance of the model after the fine-tuning by 10 points in terms of ROUGE-1 score. At the same time, we provide some useful insights about the causative factors which drive the behavior of the model, allowing for better understanding of the final predictions. We combine all the above aspects into a simple human-centered summarization interface. Our analysis sets the ground for the development of sophisticated human-centered summarization techniques, opening the way for prospective research directions.
To my mother.
Acknowledgements

First and foremost, I would like to thank my professor Dr. Grigoris Tsoumakas for the continuous support and invaluable guidance that gave me at every stage of this thesis. His passion and enthusiasm have deeply inspired me to push further my limits and try out new ideas.

A special thanks goes to all my professors from the master program for their inspiring lectures and the knowledge shared with us.

Also, I would like to express my appreciation to PhD student Alexios Gidiotis for his creative suggestions and comments as well as the guidance that provided to me whenever I needed it.

My special appreciation also goes to Efstathios Chatzykiriakidis from Medoid AI, who significantly helped to implement our idea into a novel interface.

Last but not least, I would like to thank my family who encouraged me throughout this thesis with plenty of love and unwavering support.
Περίληψη στα Ελληνικά

Η αυτόματη παραγωγή περίληψης γνωρίζει ραγδαία ανάπτυξη τα τελευταία χρόνια καθώς επιτρέπει την γρήγορη και αποτελεσματική επεξεργασία του ολόσωνα και συζητούμενου όγχου εγγράφων που είναι διαθέσιμοι στον παγκόσμιο ιστό. Επιπλέον, τα τελευταία χρόνια παρατηρείται μια συζητούμενη ανάπτυξη μοντέλων περίληψης, με ιδιαίτερη έμφαση στα μοντέλα βαθιάς μάθησης, τα οποία εξειδικεύονται σε συγκεκριμένη θεματολογία εγγράφων π.χ. ιατρικές εκδόσεις, επιστημονικές δημοσιεύσεις, χρηματοοικονομικά αρχεία κ.λπ. Παρά την αδιαμφισβήτητη επιτυχία αυτών των μοντέλων, συνήθως είναι απομονωμένα από τους χρήστες και δεν επιτρέπουν οποιαδήποτε χρήση ανατροφοδότηση από αυτούς, καθώς επίσης δεν παρέχουν ουσιαστικά κάποια εξήγηση σχετικά με την διαδικασία παραγωγής των τελικών προβλέψεων. Στην παρούσα διπλωματική εργασία, εστιάζουμε στην παραγωγή περίληψης από κείμενα οικονομικού χαρακτήρα και συνεισφέρουμε με την δημιουργία ενός νέου συνόλου δεδομένων που αποτελείται αποκλειστικά από χρηματοοικονομικά άρθρα. Επιπλέον, προσέγγιζουμε το ζήτημα της αυτόματης περίληψης μέσα από μια ανθρωποκεντρική δοκιμή δημιουργώντας ένα σύστημα στο οποίο ο χρήστης έχει ενεργό ρόλο δίνοντας του επιπλέον την δυνατότητα να επιλέξει μεταξύ διαφορετικών προτάσεων όσον αφορά την τελική περίληψη. Τέλος, αναλύουμε την ερμηνευσμόστη του βασικού μηχανισμού της σύνθετης δομής του μοντέλου με βάση τελικής προβλέψεως με στόχο την καλύτερη ερμηνεία των τελικών προβλέψεων. Ειδικότερα, παρέχουμε εμπλοκαρικά χρήση στοιχεία σχετικά με τους αιτιατούς παράγοντες του δικτύου της συμπεριφορά του μοντέλου, επιτρέποντας την καλύτερη κατανόηση των τελικών προβλέψεων με στόχο να ενισχύσουμε την εμπιστοσύνη του χρήστη με το προτεινόμενο σύστημα. Ολες οι παραπάνω πτυχές συνδυάζονται σε μια ανθρωποκεντρική διεπαφή η οποία μπορεί να χρησιμοποιηθεί εύκολα και γρήγορα από τον χρήστη. Η ανάλυσή μας θέτει τις βάσεις για την ανάπτυξη εξελιγμένων ανθρωποκεντρικών μοντέλων αυτόματης περίληψης, αναλύοντας το δρόμο για μελλοντικές ερευνητικές κατευθύνσεις.
Contents

Abstract iii

Acknowledgements vii

Περιγραφή στα Ελληνικά ix

1 Introduction 1

1.1 Contributions ................................. 2
1.2 Thesis Overview ................................. 4

2 Neural Summarization 5

2.1 Categories of Neural Summarization ............... 6

2.1.1 Based on the Input .......................... 7
2.1.2 Based on the Output .......................... 7
2.1.3 Based on the Purpose ........................ 7

2.2 Neural Summarization Datasets ..................... 8

2.2.1 CNN/Daily Mail .............................. 9
2.2.2 Gigaword ................................... 9
2.2.3 NEWSROOM ................................. 9
2.2.4 Extreme Summarization Dataset ............... 10
2.2.5 Document Understanding Conference .......... 10

2.3 Evaluation of Neural Summarization ............... 11

2.3.1 BLEU ........................................ 12
2.3.2 ROUGE ..................................... 13
2.3.3 METEOR ..................................... 14
2.3.4 Sentence Mover’s Similarity .................. 15

2.4 Neural Summarization Models ...................... 16

2.4.1 Sequence to Sequence Models ................ 17
Embeddings .................................... 17
Encoding and Decoding .......................... 18
Attention Mechanism ............................. 21
2.4.2 Transformers ........................................ 21
    Encoding and Decoding ............................... 23
    Multi-Head Attention ............................... 23
    Position Feed-Forward Layers ...................... 23
2.4.3 Pre-trained Encoders from Transformers ....... 25
2.4.4 PEGASUS for Abstractive Summarization ....... 27
2.5 Text Generation - Inference ....................... 28
    2.5.1 Greedy Search ................................ 29
    2.5.2 Random Sampling - Top-S Sampling .......... 31
    2.5.3 Beam Search .................................. 31
    2.5.4 Diverse Beam Search ......................... 33

3 Sources of Financial Documents ..................... 35
    3.1 Newswire Sources ................................ 36
    3.1.1 Bloomberg .................................... 37
    3.1.2 Financial Times ............................... 38
    3.1.3 Morgan Stanley ............................... 38
    3.2 Company filings .................................. 39
    3.2.1 SEC Filings .................................. 39
    3.2.2 FNS Workshop ................................. 41
    3.3 Our dataset ...................................... 42
    3.3.1 Data Collection ................................ 43
    3.3.2 Preprocessing ................................ 43

4 Human-Centered Financial Summarization .......... 45
    4.1 Experimental Setup ................................ 46
    4.1.1 Downstream Datasets .......................... 46
    4.1.2 Input Source Size ............................. 47
    4.2 Human-centric Approach ......................... 48
    4.2.1 Interpretation of Attention Weights ......... 48
    4.2.2 Diverse Hypotheses During Inference ...... 50
    4.3 Results ........................................... 51
    4.4 Interface ......................................... 53

5 Conclusions and Future Work ....................... 57
    5.1 Conclusions ...................................... 58
    5.2 Future Work ..................................... 58
List of Figures

2.1 Three major automatic summarization categories: based on the input, based on the output and based on the purpose. .............................................. 6

2.2 Example of an encoder-decoder architecture, which is used in sequence to sequence models. The encoder accepts a sequence of tokens $x = (x^{(1)}, x^{(2)}, x^{(...)}, x^{(n_x)})$ and encodes them to a fixed size context vector $c$. The decoder receives this context vector in order to generate the output, which in this case is a sequence of tokens $y = (y^{(1)}, y^{(2)}, y^{(...)}, y^{(n_y)})$. Figure from: (Goodfellow et al. 2016) .............................................. 20

2.3 (a) Example of an encoder-decoder architecture. The encoder receives the input sequence $x = (x_1, x_2, x_3)$ and encodes it to hidden states $h_1, h_2, h_3$. The decoder receives the last hidden states and generates the output summary $y = (y_1, y_2)$. (b) Example of an encoder-decoder architecture with attention. The context vector is the weighted sum of the hidden states with their attention weights. Figure from: (Nguyen et al. 2018) .... 22

2.4 Example of transformers architecture. Figure from: (Vaswani et al. 2017) .............................................. 24

2.5 Example of multi-head attention. The attention layers are running in parallel and then concatenated together. Figure from: (Vaswani et al. 2017) ................. 25
2.6 BERT Architecture. The left figure represents the original BERT model. The right figure represents the modified BERT for the task of summarization. The BERT for summarization uses the [CLS] tokens at the start of each sentence in order to capture the contextual sentence representation and the [SEP] tokens as boundaries to separate each sentence from the other. Figure from: (Yang Liu and Lapata 2019) ............................................. 27

2.7 PEGASUS Architecture: An encoder-decoder Transformer which uses two self-supervised objectives: GSG and MML. Figure from: (Zhang et al. 2019) ............... 29

2.8 An example of the greedy search algorithm. The algorithm always chooses the path with the higher conditional probability. The selection at each timestep is represented with the red line. Figure from: https://huggingface.co/blog/how-to-generate .......... 30

2.9 An example of Beam Search Algorithm with beam width=2. At each timestep, the algorithm expands the top probable beams based on beam width. The red and red dotted lines represent the two possible paths for the final generation. Figure from: https://huggingface.co/blog/how-to-generate .................. 32

2.10 A comparison between beam search and diverse beam search for image captioning. The generated beams of beam search are almost identical in contrast with diverse beam search that produce a variety of different captions which are also more close to the ground truth examples from humans. Figure from: (Vijayakumar et al. 2016) .................. 33

3.1 Example of Bloomberg article about financial news from the category “Markets”. The two bullet points inside the red frame represent the abstract of the article. Available at: https://www.bloomberg.com/news/articles/2019-03-03/peso-beats-asian-peers-in-surprise-win-as-philippine-cpi-eases .......................... 36
3.2 Example of Morgan Stanley article about Coronavirus in the category "Tales from the emerging world". In this example, the blue-letter lead paragraph inside the red frame serves as an opening paragraph to the article. Available at: https://www.morganstanley.com/im/en-gr/institutional-investor/insights/articles/the-era-after-the-coronavirus.html .......................... 37

3.3 Example of Morgan Stanley article about Coronavirus in the category "Tales from the emerging world". In this example, the blue-letter lead paragraph inside the red frame may represent a short summary of the key ideas of the article. Available at: https://www.morganstanley.com/im/en-gr/institutional-investor/insights/macro-insights/differentiation-over-duration.html ........................................ 39

3.4 Example of 10-Q filing of JPMorgan Chase & Co. The bullet points inside the red frame concern the key highlights of the full content of the report. Available at: https://www.jpmorganchase.com/corporate/investor-relations/document/988066cc-7900-48d3-a71b-8b2693695741.pdf ................................. 40

3.5 Example of 10-Q filing of News Corporation. The bullet points inside the red frame concern the key highlights of the full content of the report. Available at: https://newscorpcom.files.wordpress.com/2020/05/q3-2020-earnings-press-release_final_05.07.2020.pdf .............................. 41

4.1 Example of encoder attention weights of the Cnn/DailyMail model with generated summary: Government considers again delaying increase in pension contributions. Unions say pay rises since the last delay have been poor. .................................................. 50

4.2 Example of different attention weights among different levels of encoder layers. The three pictures represent the first, second and last layer, respectively. ...................... 51

4.3 Screenshot of the proposed summarization interface. . . 55
List of Tables

3.1 Data Sources: Pros and Cons .......................... 36
3.2 Dataset Statistics ........................................ 44

4.1 Comparison of input size of the original document based on Rouge scores. R-1, R-2, R-L and R-S stands for the f-measure of Rouge-1, Rouge-2, Rouge-L and Rouge-S, respectively. ........................................ 47
4.2 Example of the different summaries generated from beam search and diverse beam search algorithm. ........ 52
4.3 Results before and after finetuning based on Rouge scores. R-1, R-2, R-L and R-S stands for the f-measure of Rouge-1, Rouge-2, Rouge-L and Rouge-S, respectively. The best Rouge results are bolded. .................. 53
4.4 Example of generated summaries before and after fine-tuning for CNN/Daily Mail and XSum .................. 54
Chapter 1

Introduction

The explosive growth of the available information on the World Wide Web during the last decades, has led to an exponential increase in the amount of documents that are available online, such as blog posts and news stories. Handling and processing these huge amounts of documents requires efficient and effective information retrieval and natural language understanding tools. Text summarization is a critical aspect in this process, since it allows for condensing large documents into smaller pieces, keeping only their basic concepts, without losing any important information. In this way, users can much more efficiently identify relevant documents and mine useful information from them.

There is a rich literature on summarization methods that have been developed to address these needs. Summarization methods can be categorized into three distinct categories based on the output of the generated summary: a) extractive (Aliguliyev 2009; Ferreira et al. 2013), b) abstractive (Li et al. 2017; L. Liu et al. 2017; Nallapati et al. 2016), and c) hybrid approaches (Bhat et al. 2018; Wang et al. 2017). Extractive models work by learning how to select and copy words, phrases or whole sentences from the initial source document in order to create the summary. On the other hand, abstractive summarization models aim to generate novel summaries without being restricted to using phrases and/or sentences from the initial documents. Hybrid summarization methods combine aspects of both of these approaches to further improve their performance. Abstractive summarization is much more closer to the way humans perform summarization of documents, but it is also a significantly more challenging task that requires deeper knowledge and understanding of the linguistic aspects of the text.

These challenges led to the employment of Deep Learning (DL)
methods providing models that can capture the semantic content of a document creating well-structured summaries. These neural approaches mainly concern sequence to sequence architectures, which are typically implemented with Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) networks and Gated Recurrent Unit (GRU) (Cho et al. 2014) networks, or with more recently proposed architectures, such as transformers (Vaswani et al. 2017) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2018).

Sequence to sequence architectures are utilized for a wide range of applications, including machine translation (Bahdanau, Cho, et al. 2014; Klein et al. 2017; Luong et al. 2015), summarization and speech recognition (Bahdanau, Chorowski, et al. 2016). Variants of these approaches include mechanisms that are able to cope with some basic limitations, such as overcoming the fixed length vector representation and proposing coverage mechanisms to avoid repetition of the same words in the generated summaries. Even though LSTMs and GRUs aim to capture long-term dependencies between words in the documents, it was demonstrated that they weren’t particularly efficient on this task (Vaswani et al. 2017). To overcome this limitation, as well as address the difficulty of parallelizing sequence to sequence architectures, transformers were introduced. Transformers’ architecture consists of stacks of encoder and decoder layers, each containing a self-attention and a feed-forward sub-layer, avoiding this way entirely the need of RNNs. Other models such as PEGASUS (Zhang et al. 2019) propose already trained models with self-supervised objectives, which are fine-tuned further on downstream datasets.

1.1 Contributions

The success of neural summarization models has led to the rise of a lot of summarization datasets, which are mainly created from newswires’ articles (Grusky et al. 2018; Hermann et al. 2015; Narayan et al. 2018; Rush et al. 2015) or from scientific and academic publications (Cohan et al. 2018) and have been widely used in literature. However, to the
best of our knowledge, there is no existing dataset consisting exclusively of financial articles from newswires. Financial summarization requires deeper knowledge of the financial terminology, which may differ from the semantics of the available existing newswire datasets.

In this thesis, we tackle the issue of abstractive financial summarization by creating our own dataset consisting exclusively from financial newswire articles, fine-tuning on this dataset one of the most popular summarization models - PEGASUS - and experimenting with different decoding strategies, such as beam search and diverse beam search. Our financial-focused dataset consists of 2096 document-target examples and contains only economic articles, which are collected from Bloomberg’s website.

The fine-tuned results in this dataset indeed demonstrate the value of proper fine-tuning with appropriate data, which in our case contain specific technical terms of the financial domain. We notice that the model adapts very fast to the new dataset and after only a few training steps is capable of generating financially-focused summaries.

Despite the success of DL models, the existing approaches are isolated from the users and do not provide any way of incorporating any kind of feedback from them, like, for example, relevance feedback methods that can lead to more accurate retrieval results. As a result, existing neural summarization models just produce static summaries, which cannot be customized in order to fit the user’s needs.

In this thesis, we provide some useful insights on the self-attention mechanisms that are widely used in state-of-the-art (SOTA) encoder-decoder methodologies, in order to better understand the summarization process, attempting to interpret intuitively their decisions during inference. We also propose diverse hypotheses to the users so that they interactively participate in the summarization process by examining the generated summaries and accepting or rejecting them.

We create a simple yet novel human-centered interface allowing the users to interact with the proposed system by selecting the best summary based on their unique preferences from a number of different diverse hypotheses. The users can also submit their own suggestion for the final summary, given an input document. We can easily use these suggestions to fine-tune further our model, aiming to achieve higher performance and additional user-targeted summaries.
Chapter 1. Introduction

1.2 Thesis Overview

The rest of this thesis is organised as follows. In Chapter 2, we provide an in-depth analysis of the neural summarization categories, the existing datasets, the current DL models in the literature, as well as the available evaluation metrics, in order to provide the basic backgrounds of neural summarization. We also present the summarization model that we will use for our experiments.

In Chapter 3, we examine the available financial data sources, which can be categorized into newswire articles and company filings. We also present the source that we selected (Bloomberg’s website) and describe the collection and the final preprocessing of the data.

In Chapter 4, we describe the conducted experiments using the Bloomberg dataset. We also give insights about the mechanisms of self-attention encoder layers and examine the impact of diverse beam search in the results of the final model. The generated summaries confirm the effectiveness of the fine-tuning on our dataset. We also explain the functionality of the proposed interface providing some usage instructions.

Finally, in Chapter 5 we conclude our work and give future research directions.
Chapter 2

Neural Summarization

Neural summarization approaches range from sequence to sequence architectures such as Recurrent Neural Networks (RNN) with attention mechanism (Nallapati et al. 2016; Rush et al. 2015; See et al. 2017) to more contemporary methodologies such as Transformers (Zhang et al. 2019) and Bidirectional Encoder Representations from Transformers (BERT) (Yang Liu and Lapata 2019).

We can handle the problem of automatic summarization either as a supervised or an unsupervised machine learning task. The supervised learning approach usually requires a large number of training examples as well as a target summary, while in the unsupervised approach, we can generate the summaries without the requirement of a training corpus (Gong and X. Liu 2001; Mihalcea and Tarau 2004). We can further categorize summarization taking into consideration three different perspectives as shown in Figure 2.1. More precisely, we can categorize the employed methods based on the input: as single document or multi-document approaches, based on the output: as extractive, abstractive or hybrid approaches and based on the purpose: as generic, domain-specific or query-based methods. In this thesis, we focus on supervised single document abstractive methods aiming to extract knowledge from the financial domain.

Another concern about automatic summarization is the quality of the generated summaries. The evaluation of a summary can be conducted either by a human or by an automated system. Human evaluation, except from that it requires a lot of effort, can also be subjective due to the human factor, hence it can vary among experts. As we have to handle a large number of generated summaries, an automated evaluation metric is critically important. Methods such as Recall-Oriented
Chapter 2. Neural Summarization

2.1 Categories of Neural Summarization

As we mentioned earlier, automatic summarization can be categorized according to three basic aspects: input, output and purpose. In this section, we will describe the aforementioned categories.

Understudy for Gisting Evaluation (ROUGE) (Lin 2004), which are mainly based on the overlap of unigrams or bigrams, have been proposed in the literature.

In this chapter, we will explain the three aforementioned summarization categories focusing on neural methods. We will also examine the available summarization datasets as well as the evaluation methods, along with their strengths and weaknesses. In the two last sections of this chapter we will discuss the current models existing in the literature as well as some of the most widely used algorithms for decoding during inference.
2.1. Categories of Neural Summarization

2.1.1 Based on the Input

Taking into consideration the number of input documents, summarization methods can be divided into single document and multi-document ones. In single document summarization (Joshi et al. 2018; Litvak and Last 2008; Sarkar et al. 2013), one document per time is used as an input in order to generate the summary, while in a multi-document approach (Goldstein et al. 2000; Gupta and Siddiqui 2012), we use a collection of multiple similar-content documents. This task is more difficult due to the repetition of similar information in different documents. In this thesis, we focus on the single document category.

2.1.2 Based on the Output

We can also categorize summarization methods based on the output summary as extractive, abstractive and hybrid approaches. Extractive models (Aliguliyev 2009; Ferreira et al. 2013) produce the output by copying sentences from the input, hence the generated summary consists only of the most important existing sentences. Extractive summaries are usually more readable and cohesive, as they are extracted unaltered from the original document.

Contrary to that, abstractive methods (Li et al. 2017; L. Liu et al. 2017; Nallapati et al. 2016) generate new words and sentences to summarize the key elements of a document. The aim of abstractive summarization is to emphasize the basic points without just copying sentences, but rather by generating new words and phrases to produce the output summary. In this way, it is more close to human writing, which requires a combination of skills as far as the text structure, spelling, cohesiveness and readability concerns.

Hybrid approaches (Bhat et al. 2018; Wang et al. 2017), which are based both on abstractive and extractive methods, have also been proposed. In this thesis, we focus on abstractive summarization.

2.1.3 Based on the Purpose

The approaches of text summarization can also be categorized according to their purpose as generic, domain-specific and query-based. Generic summarization models (Gong and X. Liu 2001; Kruengkrai
and Jaruskulchai 2003) choose the highlights of the document without any previous knowledge about its domain. More specifically, the output summary is generated only by the general key elements of the document without any particular domain focus.

On the other hand, domain-specific (Kan et al. 2001; Reeve et al. 2007) approaches focus their attention on the domain terminology and can accurately generate a summary utilizing knowledge from the particular field. In our case, we focus on financial documents, which contain both economic and business terms.

Last but not least, query-based models (Kao et al. 2010; Mohamed and Rajasekaran 2006) receive an input query about a specific topic and then summarize the most crucial points in order to fully answer it. In case we have to deal with multiple documents, a summarization query-based model should be able to select only those documents that are associated with the query and then to extract the basic points and information from them.

### 2.2 Neural Summarization Datasets

A wide number of datasets has been created for the task of automatic summarization in order to train and evaluate the different models and compare the derived results. The existing datasets mainly consist of newswire articles or academic publications. Both extractive and abstractive datasets have been proposed, which can be used according to the aim of the proposed model. Existing datasets generally consist of documents that contain textual information along with a target summary. They can therefore be used in order to evaluate the generated summaries from a model, utilising the metrics that we will describe in the last section of this chapter. In this section, we will examine the most popular existing datasets that have been proposed in the literature, exploring their structure and their characteristics.
2.2.1 CNN/Daily Mail

The CNN/Daily Mail dataset (Hermann et al. 2015) has been created by combining 93K newswire articles from Cables News Network (CNN)\(^1\) and 220K articles from the Daily Mail\(^2\). The target summaries consist of bullet points as written by the authors of the articles. These bullet points highlight the key elements of the main body of the article and have abstractive character. The articles contain approximately 780 tokens, while the multi-sentence summaries, which are generally short in size, contain approximately 55 tokens. This dataset was first created for the task of question-answering, but it is also widely used for summarization (Nallapati et al. 2016; Paulus et al. 2017).

2.2.2 Gigaword

Gigaword (Rush et al. 2015) consists of 10 million English documents from seven international online data sources, including, among others, Associated Press Worldstream\(^3\), Washington Post Newswire Service\(^4\), and New York Times Newswire Service\(^5\). Gigaword automatically generates the syntactic structure annotation providing tokenized and segmented sentences, treebank-style and syntactic dependency trees, name entities as well as coreference chains inside the document. At first, the annotated summaries were not provided. However, later a Gigaword headline-document dataset (Graff et al. 2003) was proposed, which consists of approximately 3 million pairs. The headlines of the articles, which are considered as summaries, are very short in size, containing 7-8 words on average.

2.2.3 NEWSROOM

NEWSROOM (Grusky et al. 2018) is another large-scale summarization dataset, which consists of 1.3 million online human written articles and summaries from 38 different online news publications. The articles were collected throughout two decades and more specifically

\(^1\)https://edition.cnn.com/
\(^2\)https://www.dailymail.co.uk/home/index.html
\(^3\)https://apnews.com
\(^4\)https://www.washingtonpost.com
\(^5\)https://www.nytimes.com/
between 1998 and 2017, by extracting social media and search engine metadata. The large period of time between the extracted data diversifies the writing style between the authors and editors, providing unique opportunities to capture the different aspects of the narrative style. The mean article length is 659 words, while a target summary contains approximately 27 words on average. It has also to be noted that the target summaries are associated with both extractive and abstractive strategies as they utilize mechanics for copying words and expressions from the document at different frequencies, as well as generating novel words, respectively.

2.2.4 Extreme Summarization Dataset

Extreme Summarization Dataset (XSum) (Narayan et al. 2018) is a large-scale dataset that consists of approximately 226K articles from the time period of 2010 to 2017, providing a wide variety of topics ranging from politics, economy, business, technology, to entertainment, art and family. The articles were extracted online from the British Broadcasting Corporation (BBC), along with human-written single sentence summaries. These summaries were usually written by the author or the editor of the article. The average length of a document is 226 words and the target summary contains 23 words. The target sentences were not extracted from the original document, hence the dataset aims to demonstrate abstractive summarization strategies.

2.2.5 Document Understanding Conference

The Document Understanding Conference (DUC) dataset collection consists of a number of datasets with multiple human-written summaries in order to evaluate different summarization models. The proposed multiple reference summaries for each document can be used both for supervised or unsupervised models. It has also to be noted that the ROUGE (Lin 2004) metric, which is a very common evaluation metric, was designed for multiple references summaries, giving a great advantage to the DUC datasets. We will extensively examine the ROUGE metric in the next section of this chapter.

---

6https://www.bbc.com/
DUC concerns a question-focused task, which can easily be applied to the task of automatic summarization. The aim of these datasets is to produce a answer that summarizes the basic points to a complicated question from single or multiple documents. Due to the limited number of documents, the datasets generally can not be used on their own for training, hence they are usually combined with other already mentioned datasets such as Gigaword, CNN/DailyMail, etc.

2.3 Evaluation of Neural Summarization

A major issue to be considered is the evaluation of the generated summaries. The evaluation of a summary can either be conducted by humans or by particular automation systems.

Experts can judge the quality of a summary based on different aspects such as readability, conciseness, coherence, grammar and spelling as well as the relevance of the basic points from the original document. Even though automated systems usually lack the ability to capture all the aforementioned aspects of a text, human evaluation has two significant drawbacks. First of all, evaluating a large corpora of summaries is both time-consuming and expensive. Second, the evaluation of the generated summaries may differ between experts, due to the subjective point of view among them. Hence, the development of automated metrics is significantly critical.

On the other hand, a wide variety of automated evaluation metrics such as ROUGE (ibid.) and Bilingual Evaluation Understudy (BLEU) (Papineni et al. 2002), have been proposed in the literature in order to judge the quality of the output summaries. However, the aforementioned metrics are based mainly on word-pairs occurrences and overlapping of output tokens, ignoring the semantic information of the words. To address this issue, other metrics such as Metric for Evaluation of Translation with Explicit ORDERing (METEOR) (Banerjee and Lavie 2005) and Sentence MOVER's Similarity (Clark et al. 2019), have been suggested. In this section, we will examine these different metrics and their employed methodology.
2.3.1 BLEU

The BLEU metric was first introduced for evaluating machine translation tasks, but is also widely applied in automatic summarization. BLEU is a precision-based metric which counts the average of n-gram between the target and the predicted summary and is calculated as follows:

\[
\text{BLEU} = \text{BP} \cdot \exp \left( \sum_{n=1}^{N} w_n \log p_n \right)
\]  

(2.1)

where \( p_n \) is the modified n-gram precision, \( w_n \) is a weight between 0 and 1, \( N \) is the length of the n-grams that are used, while \( \text{BP} \) is the brevity penalty. The default value of \( w_n \) is \( \frac{1}{N} \) with \( N = 4 \).

Brevity penalty is used in order to penalize short candidate sequences and is computed as follows:

\[
\text{BP} = \begin{cases} 
1 & \text{if } c > r \\
\exp(1 - r/c) & \text{if } c \leq r
\end{cases}
\]  

(2.2)

where \( c \) is the target length and \( r \) the reference length.

The modified n-gram precision is calculated as follows:

\[
p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n-gram \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n-gram' \in C'} \text{Count}(n\text{-gram}')} 
\]  

(2.3)

where \( \text{count} \) is the maximum number of times that a n-gram occurs in a reference sequence and \( \text{Count}_{\text{clip}} = \min(\text{count}, \text{max_ref_count}) \) with \( \text{max_ref_count} \) to represent the maximum n-gram occurrences in any reference count. This ensures that the counts of any n-gram will not exceed the highest count for this n-gram in any reference summary.

Even though BLEU is an easy and comprehensible method for evaluation, it does not take into account the semantic meaning of the text, as well as the sentence structure. Thus, it fails to capture the rich morphological characteristics of natural language. In order to address some of these problems, the ROUGE family of metrics is proposed.
2.3.2 ROUGE

**ROUGE** is a package of automated evaluation metrics, which is widely used in automatic summarization and machine translation tasks. It contains several metrics including, among others, **ROUGE-N**, **ROUGE-L** and **ROUGE-S**. These metrics are based on overlapping units (unigram, bigram, etc.), word sequence co-occurrences as well as longest common subsequences, between the generated and the target summary.

**ROUGE-N** or *N-gram Co-Occurrence Statistics*, measures the n-gram recall between the target and generated summaries. The parameter $N$ defines the number of tokens in the sequence hence, ROUGE-1 and ROUGE-2, count the co-occurrence of the unigrams and bigrams, respectively. ROUGE-N is calculated as follows:

$$\text{ROUGE-N} = \frac{\sum_{S \in R \text{ summaries}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match} (\text{gram}_n)}}{\sum_{S \in R \text{ summaries}} \sum_{\text{gram}_n \in S} \text{Count} (\text{gram}_n)}$$ \hspace{1cm} (2.4)

where $\text{Count}_{\text{match} (\text{gram}_n)}$ and $\text{Count} (\text{gram}_n)$ represents the count of n-gram co-occurrences and the total number of n-grams, respectively. The numerator counts all the overlapping n-grams between the reference and the target summary for all the reference summaries, while the denominator counts all the n-grams in the reference summaries. Taking into consideration that there is more than one acceptable summary, the number of the denominator increases while adding more target summaries. Hence, if an n-gram appears in several target summaries then it is automatically assigned with more weight.

**ROUGE-L** or *Longest Common Subsequence*, evaluates the quality of a generated summary by counting the maximum-length co-occurring subsequence. We can define the LCS as the maximum length of an overlapping sequence between $X$ and $Y$, which represent the reference and the candidate summary, respectively. Assuming that a summary $X$ has length $m$ and the candidate summary $Y$ has length $n$, we can calculate ROUGE-L - a F-measure metric which is based on LCS - as follows:

$$R_{\text{LCS}} = \frac{\text{LCS}(X, Y)}{m}$$ \hspace{1cm} (2.5)
Chapter 2. Neural Summarization

\[ P_{lcs} = \frac{LCS(X, Y)}{n} \]  
\[ F_{lcs} = \frac{(1 + \beta^2)R_{LCS}P_{lcs}}{R_{lcs} + \beta^2P_{lcs}} \]

where \( LCS \) indicates the longest common subsequence of the sequences \( X \) and \( Y \), with \( \beta = \frac{P_{lcs}}{R_{lcs}} \). It has also to be noted that when the two sequences are identical, i.e. \( X = Y \), the \( LCS(X, Y) = m \) or \( LCS(X, Y) = n \), hence the ROUGE-L score is 1. In case of two sequences \( X \) and \( Y \), which do not share any common subsequence, the ROUGE-L score is 0, considering \( LCS(X, Y) = 0 \). Since, LCS is calculated automatically, there is no need for defining the n-gram length. ROUGE-L aims to capture sentence structure, since it is interested in in-sequence matches and not merely in n-gram consecutive co-occurrences of words.

**ROUGE-S** or **Skip-Bigram Co-Occurrence Statistics**, counts the co-occurrence of skip-bigrams, between the reference and the generated summary. Any possible combination of pairs of words within a sentence can be considered as a skip-bigram. ROUGE-S is also an F-measure metric and calculated as follows:

\[ R_{skip2} = \frac{SKIP2(X, Y)}{C(m, 2)} \]  
\[ P_{skip2} = \frac{SKIP2(X, Y)}{C(n, 2)} \]  
\[ F_{skip2} = \frac{(1 + \beta^2)R_{skip2}P_{skip2}}{R_{skip2} + \beta^2P_{skip2}} \]

where \( SKIP2 \) represents the total count of skip-bigram overlap of the sequences \( X \) and \( Y \), with \( \beta = \frac{P_{lcs}}{R_{lcs}} \). When \( \frac{\partial F_{lcs}}{\partial R_{lcs}} = \frac{\partial F_{lcs}}{\partial P_{lcs}} \), and \( C \) indicates the combination function.

### 2.3.3 METEOR

**THE METEOR** metric creates a word alignment between a pair of words. Given two sequences, METEOR maps every word from the first sequence to explicitly one (or zero) word from the other sequence. In order to calculate this word alignment, three basic modules have
been proposed: the exact, the Porter stem and the WN synonymy. The exact module maps a word to another only if they are identical, while in the Porter stem module words are mapped together only if they have derived from the same stem. In WN synonym, the words must be synonyms according to the WordNet lexical database. The final METEOR score is calculated as follows:

\[
\text{METEOR} = (1 - P_{en}) \cdot F_{mean} \tag{2.11}
\]

where \( P_{en} \) represents the penalty for the proposed alignment, while \( F \) indicates the parameterized harmonic mean, based on precision and recall. The penalty is computed with the following formula:

\[
P_{en} = \gamma \cdot \text{frag}^\beta \tag{2.12}
\]

\[
\text{frag} = \frac{\text{ch}}{m} \tag{2.13}
\]

where \( \gamma \) is a parameter between 0 and 1, which indicates the maximum penalty while \( \text{frag} \) is the fragmentation function with \( \text{ch} \) and \( m \) representing the number of chunks and matches of the alignment, respectively. The relation between the penalty and the fragmentation is denoted by \( \beta \).

This metric intends to capture the semantic aspect of a generated text, since it takes into consideration the occurrence of equivalent words with the same meaning. Experiments have shown that METEOR can enhance the correlation with human evaluation (Banerjee and Lavie 2005; Denkowski and Lavie 2014).

2.3.4 Sentence Mover’s Similarity

**Sentence Mover’s Similarity** is a similarity metric for multi-sentences text which can be successfully applied in automatic summarization. Sentence Mover’s Similarity, uses both semantic and syntactic structure in order to evaluate the generated summary, claiming that the similarity between documents is highly associated with the distance of their sentence embeddings.

Two evaluation metrics have been proposed: *Sentence Mover’s Similarity (SMS)* and *(S+WMS) Sentence and Word Mover’s Similarity*. The
first one is based only on sentence embeddings, while the other one uses both sentence and word embeddings.

In SMS, a model of a bag of sentences is created to represent a multi-sentence sequence of text. In order to create the document representation, the sentence embeddings of a document are combined together as an average. In order to calculate the weight of a sentence embedding, we count the number of words that contains. In particular, the weight of a sentence $i$ in a document $A$ is calculated as follows:

$$d_{A,i} = |i|/|A|$$  \hspace{1cm} (2.14)

where $|i|$ and $|A|$ represent the number of the sentences and words, respectively.

In S+WMS, both word and sentence embeddings are used in order to represent a document. Similar to SMS, a bag of words, weighted by their frequency, and sentences, weighted by their length, is created. The weight of a sentence $i$ in the document $|A|$ is calculated as follows:

$$d_{A,i} = \begin{cases} \text{count}(i)/2|A| & \text{if } i \text{ is a word} \\ i/2|A| & \text{if } i \text{ is a sentence} \end{cases}$$  \hspace{1cm} (2.15)

### 2.4 Neural Summarization Models

The significant growth of neural summarization has brought in a great amount of different neural approaches including, among others, sequence to sequence architectures, which are based on encoders and decoders with attention mechanisms as well as transformers and self-supervised pre-trained models, such as PEGASUS (Zhang et al. 2019). Other methods have been proposed to handle the long-input documents, which are based on divide and conquer algorithms (Gidiotis and Tsoumakas 2020) or structure-based models (Gidiotis and Tsoumakas 2019). In this section, the most basic and fundamental methods proposed in the literature are discussed and categorized according to their employed methodology.
2.4. Neural Summarization Models

2.4.1 Sequence to Sequence Models

Sequence to Sequence models are widely applied to language generation tasks such as machine translation (Bahdanau, Cho, et al. 2014; Klein et al. 2017; Luong et al. 2015) and automatic summarization (Nallapati et al. 2016; See et al. 2017), particularly due to the different and non fixed length of the input/output. Their architecture consists of the following components: an encoder, a decoder and most of the times an attention mechanism. Generally, variants of RNNs such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) are used for implementing the encoder and the decoder due to the vanishing gradient problem of RNNs. The weakness of the traditional RNNs to capture and memorize long input text has led to the development and use of such methods, which can handle efficiently long-term dependencies. An attention mechanism is usually added in these approaches to help with memorizing long inputs, by giving special “attention” to the most important parts of the document.

Different variants of sequence to sequence models have been proposed in the literature. For example, Pointer Generator Networks (See et al. 2017) uses a similar encoder-decoder architecture which combines both copying words from input and generating new words from an existing vocabulary. A coverage mechanism has been added to avoid repetition in the output. In this subsection, we will examine the basic components of a sequence to sequence model and more specifically the encoding and decoding phase as well as the attention mechanism.

Embeddings

There are two types of embeddings that are widely used in neural language models: word embeddings, which are also known as static embeddings, and contextual embeddings. Static embeddings have the same vector representation regardless the context of the text, while contextual embeddings are dynamically changed based on the context.

Word embeddings are used in order to represent the meaning of a word as a multi-dimensional vector. This kind of representation can be a useful tool for many NLP tasks allowing for better generalization to unknown words. At the same time, word embeddings can capture
word semantics with words with close meaning having similar vector representation.

Word embeddings can be divided into two distinct categories: pre-trained embeddings and one-hot vector embeddings. One-hot vector embeddings are added as an extra layer to the neural model and are first initialized with random values. Each word is represented as one hot-vector of length $|V|$, where $V$ represents the model’s vocabulary. All the values of the vector are set to 0 except of the dimension which indicates the word’s index in the vocabulary, which is set to 1. The embeddings are trained in parallel with the neural model before getting their final representations. Pre-trained Embeddings like seq2sec (Mikolov et al. 2013) or GloVe (Pennington et al. 2014) are embeddings that have already been trained on large datasets in order to learn the semantics of a word. We can use these pre-trained embeddings in any NLP tasks without further training.

Contextual Embeddings like BERT (Devlin et al. 2018) are used in recently neural models such as Transformers. These embeddings are not static and generated based on the other words of the text i.e. the context of the text.

**Encoding and Decoding**

The two major phases of a sequence to sequence model is the encoding and the decoding phase, as shown in Figure 2.2. Let’s denote the document as $x = (x_1, x_2, \ldots, x_J)$ and the output summary as $y = (y_1, y_2, \ldots, y_T)$, where $J$ and $T$ is the number of words, known as document length of the initial input document and the output summary, respectively. Given a sequence of fixed length input tokens $x$, which here is represented using one-hot vector, the aim of a sequence to sequence model is to generate a summary $y$. The encoder which is denoted with $e$, receives the tokens one by one and encodes them in a unique inner representation with hidden states $h = (h_1^e, h_2^e, \ldots, h_J^e)$, also known as context vector. The hidden states are passed to the decoder which is represented with $d$ in order to generate the output summary $y$. When using LSTM units, the input sequences of tokens are
2.4. Neural Summarization Models

encoded to hidden states using the following formulas at time step $t$:

$$i_t = \sigma(W_{ii}E_{xt-1} + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \quad (2.16)$$

$$f_t = \sigma(W_{if}E_{xt-1} + b_{if} + W_{hf}h_{t-1} + b_{hf}) \quad (2.17)$$

$$o_t = \sigma(W_{io}E_{xt-1} + b_{io} + W_{ho}h_{t-1} + b_{ho}) \quad (2.18)$$

$$g_t = \tanh(W_{ig}E_{xt-1} + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \quad (2.19)$$

$$c_t = f_t c_{t-1} + i_t g_t \quad (2.20)$$

$$h_t = o_t \tanh(c_t) \quad (2.21)$$

The $i_t$, $f_t$ and $o_t$ represent the input, forget and output gate, respectively while the $E_{xt-1}$ represents the embedding of the input token for the previous time step from the embedding matrix $E$. $W$, which indicates the weight matrices and $b$, the bias parameters, are learnable parameters.

As we mentioned earlier, an encoder can be implemented by an LSTM, GRU etc. Recent approaches use bidirectional LSTMs, which are able to capture more aspects of the input and as a result to lead to a better representation. In this case, the input sequence is transformed into hidden states $\overrightarrow{h}$ and $\overleftarrow{h}$, where the arrows represent the encoding direction.

Then, the decoder receives these hidden states and update them as follows:

$$h^d_t = LSTM(h^d_{t-1}, E_{\hat{y}_{t-1}}, c) \quad (2.22)$$

where $E_{\hat{y}_{t-1}}$ is the embedding from the softmax output at the previous timestep.

Finally, in order to generate a word for the output summary, the probability distribution is calculated using softmax, with the following formula:

$$P_{\text{Vocab}, t} = \text{Softmax}(W_d h^d_t + b_d) \quad (2.23)$$

At every timestep, $P_{\text{Vocab}, t}$, which represents the vocabulary, gives a probability distribution for all output words in our dictionary.
FIGURE 2.2: Example of an encoder-decoder architecture, which is used in sequence to sequence models. The encoder accepts a sequence of tokens $x = (x^{(1)}, x^{(2)}, \ldots, x^{(n)})$ and encodes them to a fixed size context vector $c$. The decoder receives this context vector in order to generate the output, which in this case is a sequence of tokens $y = (y^{(1)}, y^{(2)}, \ldots, y^{(n)})$. Figure from: (Goodfellow et al. 2016)
2.4. Neural Summarization Models

Attention Mechanism

A major limitation of the encoder-decoder approaches is that all the information from a given document must be compressed into a fixed length vector. This approach is indeed functional for short documents, however it may cause severe information loss for larger documents. In order to overcome this limitation, (Bahdanau, Cho, et al. 2014) has led to the development of the attention mechanism, as introduced by (ibid.). The attention mechanism serves as an intervening layer between an encoder and a decoder as shown in Figure 2.3. In this case, the context vector is the weighted sum of the hidden states with their attention weights. More specifically, at each timestep, the decoder not only takes the encoded hidden states of the input sequence but also gives more attention to the most relevant parts. The attention distribution is computed as follows:

$$a_{ij} = \text{Softmax}(s(h_{ej}, h_{dt}))$$ (2.24)

where $s(h_{ej}, h_{dt})$ can be calculated by a content-based function with the following three alternatives (LeCun et al. 2015): dot, general and concat.

$$s(h_{ej}, h_{dt}) = \begin{cases} h_{ej}^c \top h_{dt}^d & \text{(dot)} \\ h_{ej}^c \top W_{\text{align}} h_{dt}^d & \text{(general)} \\ v_{\text{align}} \top \tanh(W_{\text{align}} (h_{ej}^c \oplus h_{dt}^d + b_{\text{align}})) & \text{(concat)} \end{cases}$$ (2.25)

It has also to be noted that both the general and concat functions are mainly used for abstractive summarization. (Paulus et al. 2017; See et al. 2017)

2.4.2 Transformers

Transformers’ architecture is similar to the architecture of an encoder-decoder, which we described in the previous section. In this case, stacked self-attention fully-connected layers are used, as shown in Figure 2.4.
Chapter 2. Neural Summarization

Figure 2.3: (a) Example of an encoder-decoder architecture. The encoder receives the input sequence $x = (x_1, x_2, x_3)$ and encodes it to hidden states $h_1, h_2, h_3$. The decoder receives the last hidden states and generates the output summary $y = (y_1, y_2)$. (b) Example of an encoder-decoder architecture with attention. The context vector is the weighted sum of the hidden states with their attention weights. Figure from: (Nguyen et al. 2018)
Encoding and Decoding

The encoder consists of $N$ layers with two sublayers: one feed-forward fully-connected layer and one multi-head attention mechanism.

The decoder consists of the same encoder layers, as well as a third sublayer with a masked multi-head attention mechanism. Residual connections are applied over the sublayers of the encoder and decoder, followed by a normalization layer.

Unlike RNNs which capture the order of the input sequence, Transformers do not have such means in order to understand the position of each token. In order to alleviate this problem, positional encoding is used. Positional encoding is added to the bottom of the input and output embedding of the encoder and decoder, respectively, as shown in Figure 2.4. The positional encoding and the embeddings are summed in order that the model takes into account the position of each token.

Multi-Head Attention

Transformers use a multi-head attention mechanism instead of single attention, which was described in the previous section. The use of multi-head attention aims to capture as much information as possible, among the individual subspaces over all the possible representations at every position. The formula of this mechanism is calculated as follows:

$$ MultiHead_{(Q,K,V)} = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O $$

$$ \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, V^W_i) $$

where $QW_i^Q$, $KW_i^K$ and $V^W_i$ are parameter matrices learned from linear projections in dimensions $d_q$, $d_k$ and $d_v$, respectively. An example of the multi-head attention mechanism is shown in Figure 2.5

Position Feed-Forward Layers

The encoder and decoder comprise a fully connected feed-forward network, which is positioned in each layer. This network consists of two linear transformations and uses the activation function ReLU as
Chapter 2. Neural Summarization

Figure 2.4: Example of transformers architecture. Figure from: (Vaswani et al. 2017)
2.4. Neural Summarization Models

![Multi-Head Attention Diagram](image)

**Figure 2.5**: Example of multi-head attention. The attention layers are running in parallel and then concatenated together. Figure from: (Vaswani et al. 2017)

follows:

\[
FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.28}
\]

where the feed forward network is denoted by \( FFN \).

### 2.4.3 Pre-trained Encoders from Transformers

The development of transformers has led to the rise of different language representation models, during the last years. Hence, a wide range of pretrained-encoders has been developed, including, among others, BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2018), RoBERTa (Yinhan Liu et al. 2019) and ALBERT (Lan et al. 2019). These models have been trained in large corpuses and they can be easily used with some modifications for different NLP tasks. Since there is an extensive amount of particular models, we will briefly examine one of them in order to explain the bottom line of the employed methodology. More specifically, we will discuss the BERT architecture which indicates the benchmark for the subsequent models and can be used also for the task of summarization.

In BERT, the input sequence is given in the model in a special format, using two distinct tokens: the [CLS] and the [SEP] token. The
[CLS] indicates the beginning of a text sequence, while the [SEP] token is used for sentence separation and it is positioned after the end of each sentence. In order to provide rich linguistic representation for each token, BERT uses the three following embeddings as far as the tokens, segments and position, concerns. The token embeddings represent the semantic definition of each word, the segmentation embeddings serve as boundaries between the sentences and the position embeddings indicate the distinct position of each token in the sequence.

The BERT architecture is shown in Figure 2.6. The left part of the picture indicates the original BERT model, while the right part represents the modified BERT for the task of summarization (Yang Liu and Lapata 2019). BERT for summarization uses the [CLS] tokens at the start of each sentence, in order to capture the contextual sentence representation. The [SEP] tokens are used as boundaries to separate each sentence from the other.

Then, the final vector, which is the sum of the aforementioned embeddings, passes to a multi-layer Bidirectional Transformer as follows:

$$\tilde{h}^l = \text{LN}(h^l-1 + \text{MHA}(h^l-1))$$ (2.29)

$$h^l = \text{LN}(\tilde{h}^l + \text{FFN}(\tilde{h}^l))$$ (2.30)

The $h^0$ denotes the initial input vector while LN and MHA represent the layer normalization and the multi-head attention, respectively. The superscript $l$ represents the depth of the transformer layer. Finally, for each token a unique output vector representation is generated.

BERT uses in parallel two different pre-training self-supervised objectives: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP). In MLM, a number of random tokens is removed from the original sequence and replaced with a [MASK] token. A bi-directional transformer, based on the neighbor tokens from both left and right, attempts to predict a word for the [MASK] token. In NSP, given two sentences, the model is trained to predict if the last sentence was following the first sentence in the initial document.
2.4. Neural Summarization Models

2.4.4 PEGASUS for Abstractive Summarization

PEGASUS (Zhang et al. 2019) for Abstractive Summarization is an encoder-decoder pre-trained transformer, which we will use extensively in our work. PEGASUS proposes a new self-supervised objective: Extracted Gap Sentences Generation (GSG), aiming to create a new sophisticated abstractive summarization model with the assumption that the pre-training objective to the downstream task is highly associated with easiest and fastest fine-tuning. In GSG, existing sentences from the initial document are removed and masked with [MASK] tokens. Then, the model is trained to predict these whole sentences in parallel with the MLM objective, which has already been discussed. The architecture of PEGASUS is shown in Figure 2.7.

PEGASUS selects the masked sentences with the three following strategies: random, lead and principal. The random strategy selects randomly some sentences from the document, while the lead strategy selects only a predefined number of the first sentences. The principal strategy aims to choose the most important sentences of the original document, based on ROUGE metrics. The ROUGE score is calculated
Chapter 2. Neural Summarization

among the candidate sentence and the remaining sentences of the document as follows:

\[ s_i = \text{ROUGE}(x_i, D \setminus \{x_i\}), \forall i \]  

(2.31)

where \( D \) and \( x \), represent the document and its sentences, respectively.

PEGASUS is pre-trained on two large corpora: C4 and HugeNews. C4 (Colossal and Cleaned version of CommonCrawl) contains crawling data of approximately 350 millions web pages (Raffel et al. 2019), while HugeNews consists of 1.5 billions news articles from websites. After the pre-training, the model is fine-tuned on 12 downstream datasets including CNN/DailyMail, XSum, Multi-News, GigaWord and other common summarization datasets which have already been discussed.

PEGASUS model provides two variants of the same architecture: PEGASUS\text{Base} and PEGASUS\text{Large}. The base model consists of 12 layers for both the encoder and the decoder with 12 attention heads for each of them, while the Large model contains 16 layers of encoder and decoder with 16 attention heads. The batch size of the proposed models is 256 and 8192, for the base and the large model, respectively. The final model, which is also uploaded in GitHub\footnote{https://github.com/google-research/pegasus} is the PEGASUS\text{Large} model using beam search decoding strategy with a length penalty.

2.5 Text Generation - Inference

Despite the great success of the most recently proposed DL methodologies of neural summarization models, the generation of a salient summary is still not a straightforward task. Summarization methodologies are based on the hypothesis that given an input sequence, the generated output sequence is strongly associated with the original input, enhancing this correlation by using different mechanisms, such as attention and self-attention. In order to generate a summary, the model follows an iterative process, choosing at each timestep the next token over a probability distribution of the existing vocabulary. If an attention mechanism is utilized, the model also takes into account the
current sequence for the generation of the next word. This process, which is called inference, is the most important phase of summarization systems, allowing for the evaluation of the model’s performance with the appropriate metrics, such as ROUGE. Differences at the decoding mechanisms can lead to improvement in performance of the same trained model. For this purpose, there have been proposed various decoding strategies for language modeling - with Beam Search algorithm being the most popular one. In this section, we will discuss some of these decoding methods along with the advantages and disadvantages each one presents.

2.5.1 Greedy Search

The greedy search algorithm is one of the most common categories of heuristic algorithms that produce suboptimal solutions to a wide variety of problems. Greedy search, which is also known as the best-first algorithm, does not guarantee an optimal solution as it always chooses greedily the local optimum at each step. Correspondingly, in the case of summarization language models, the final sequence of the generated output tokens is not necessarily the optimal one, as the
model chooses the best token at each time step, namely the one with the max probability as shown in Figure 2.8. Hence, the token with the highest conditional probability may not lead to the optimal sequence of tokens.

One common issue which arises from the nature of this algorithm is that after some number of timesteps, it tends to repeat the same tokens again and again. This behaviour has also been noticed in the beam search algorithm, which will be discussed in the next subsection.

Even though choosing the best token at each step can produce a good selection of the final sequence, it is possible for the algorithm to ignore other likely high score tokens, which may lead to a better final path for the final sequence. For this reason, greedy search is not usually preferred for language modeling tasks. This problem has partially been addressed with the introduction of the beam search algorithm. (Shao et al. 2017; Vijayakumar et al. 2016)
2.5.2 Random Sampling - Top-S Sampling

Random sampling selects randomly a token from the probability distribution, at each timestep. This approach is usually combined with a parameter of temperature that takes into account the distribution’s entropy in order to generate more coherent and rational results (Holtzman et al. 2019).

It has been observed that the higher the temperature, the higher the randomness of the generated tokens in the final sequence. It has also to be noted that setting the temperature to 0 will produce similar results to the greedy search algorithm. Another variation of random sampling is top-S sampling (Fan et al. 2018), which limits the possible choices at each time step only to the top S tokens over the vocabulary distribution.

2.5.3 Beam Search

Beam search is another heuristic search algorithm that acts similarly to greedy search. In this case, instead of selecting only the single token with the highest conditional probability at each time step, we also keep a number of other possible high probability tokens, based on a fixed parameter, which is called beam width. In fact, when the beam width is set to 1, we are using the beam search exactly as the greedy search algorithm.

The process is repeated until the final token is generated, keeping at each step as many tokens as the number of the beam width, concluding with as many final paths as the selected beams as shown in Figure 2.9.

Despite the successful and widespread use of beam search as a decoding strategy for many NLP tasks the algorithm has some limitations. First, the same repetition issue with greedy search has been noticed also in beam search, which has partially been addressed with adding n-gram blocking and penalty parameters (Wu et al. 2016). Also, it suffers from lack of diversity between the final top hypotheses. In other words, the final beams are almost identical to each other. To address this issue, a simple modification of the beam search algorithm, diverse beam search, has been proposed (Vijayakumar et al. 2016).
Figure 2.9: An example of Beam Search Algorithm with beam width=2. At each timestep, the algorithm expands the top probable beams based on beam width. The red and red dotted lines represent the two possible paths for the final generation. Figure from: https://huggingface.co/blog/how-to-generate
2.5.4 Diverse Beam Search

Diverse beam search can alleviate the problem of identical beams of the beam search algorithm by adding a diversity term in the objective in order to take into consideration the sequence dissimilarity without sacrificing quality. In other words, the model is encouraged to generate different beams, which can finally lead to different suggestions for the final summary. An example of the different produced output sequences between the two algorithms is shown in Figure 2.10.

A diversity between the final summaries can be useful in the case of human-centric summarization systems, allowing the users to actively participate in the process by selecting the summary which fits best in their preferences.
Chapter 3

Sources of Financial Documents

As long as the task of summarization is handled as a supervised machine learning problem, it requires a large number of input training examples to provide an accurate prediction. Hence, a great amount of datasets has been created for this purpose, including, among others, cnn/daily mail, newsroom, multinews and gigaword, as we discussed in the previous chapter. The aforementioned datasets have been created mainly from newswires and have been widely used in the literature for evaluating different summarization models.

Apart from newswire articles, we can also tackle the problem of financial automatic summarization from another perspective taking into consideration that financial documents can include company economic reports, known also as company filings. These reports usually consist of company financial information for a specific period of time such as business earnings and losses and are usually much longer than economic articles.

Recently, a workshop introduced a Financial Narrative Summarization (FNS 2020) task, which concerns company annual reports. However, collecting this type of reports is not a straightforward task, since the majority of company filings does not have an explicit structure.

In this chapter, we examine the FNS 2020 workshop and present some of the available sources for collecting financial documents, either from company filings or from newswires, such as Bloomberg and Financial Times. In addition, we discuss their advantages and disadvantages, which are summarized in Table 3.1.

In the last section of this chapter, we present the financial source that we selected for this thesis, along with a description of the data collection process.
3.1 Newswire Sources

Newswires are services that allow real-time continuous transmission of news through the internet, on a daily basis. As we are interested in financial articles, we focus on financial newswire services such as Financial Times and Bloomberg or even on companies that provide similar financial services, such as Morgan Stanley. In this section, we will discuss the available ways of collecting data from the aforementioned websites as well as the emerging challenges each one presents.

<table>
<thead>
<tr>
<th>Sources</th>
<th>Financial Times</th>
<th>Bloomberg</th>
<th>Morgan Stanley</th>
<th>FNS</th>
<th>Filings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Times</td>
<td>✔ API</td>
<td>✔ API</td>
<td>✔ Short articles</td>
<td>✔ Annotated dataset</td>
<td>✔ Novel task</td>
</tr>
<tr>
<td>Paid license</td>
<td>✔ Short articles</td>
<td>✔ Available Abstract</td>
<td>✔ No API</td>
<td>✔ Large Documents</td>
<td>✔ No structure</td>
</tr>
<tr>
<td>API</td>
<td>✔ Short articles</td>
<td>✔ Paid subscription</td>
<td>✔ No structure</td>
<td>✔ Large documents</td>
<td>✔ Large documents</td>
</tr>
</tbody>
</table>
3.1. Newswire Sources

3.1.1 Bloomberg

Bloomberg offers financial and business news about global markets, stock prices and economic current affairs from all over the world. It provides a wide variety of topics including, among others, markets, technology, politics and environmental issues. The financial articles from Bloomberg can be found in specific categories such as stocks, markets and rate, and are usually short in size (approximately 400-600 words). In most cases, these articles are accompanied with a small abstract of 2-3 bullet points at the beginning, as shown in Figure 3.1.

Bloomberg provides an open API for free in four programming languages: C++, C#(.NET), Java and Python, along with an extensive documentation of them. However, in order to have access to the full content of the Bloomberg website you need to connect from Bloomberg Terminal, which requires a paid subscription.
3.1.2 Financial Times

Financial Times is a global newswire service about financial and business news, which provides articles about international economic current affairs along with market data for companies and businesses. Even though some articles are accessible from their website, a paid subscription licence is necessary to access the full content. Financial Times provides two APIs: the headlines API and the data mining API. The headlines API requires the headlines license, which is free, but can only give access to the headlines of the news. The data mining license, which is a paid license, can give access to the full content of the website. The documentation of the available APIs is clear and easy to use.

3.1.3 Morgan Stanley

Morgan Stanley is an international investment bank located in New York that also supports financial and business services. The website of this company provides articles and podcasts concerning economic and business issues of international current affairs. The articles, which are accessible through the website, are short in size and usually begin with an opening paragraph in blue font as shown in Figures 3.2 and 3.3.

Even though in some cases this blue paragraph could represent a short summary of the article, most of the time it is just a simple lead paragraph which serves as an introduction to the topic that will be discussed. Moreover, there is a limited number of available articles on the website, since Morgan Stanley is not a digital newspaper such as Bloomberg and Financial Times. Additionally, there is not an available API to extract these articles. Taking into consideration the limited number and the non-explicit structure of the articles as well as the lack of an API, we can conclude that Morgan Stanley is not the first option to collect data for the requirements of this thesis.
3.2 Company filings

Companies are obligated to complete and deliver forms at regular intervals. These company forms are called filings and contain information about earnings, risk factors or other economic data. As we mentioned above, financial company reports have a very different structure from the newswires articles, both in size and structure. In this section, we will examine the filings data sources, as well as their strengths and benefits and we will discuss how they differentiate from newswires.

3.2.1 SEC Filings

SEC filings are financial documents or statements that a company is required to submit to the U.S. Securities and Exchange Commission. They are generally more extensive than a newswire article and contain both narrative and numerical information. There exist a large number of filings types including, among others, 10-K, 10-Q, 8-K and 6-K,
which contain information about risk factors, business and management analysis, earning releases, etc. The majority of them are accessible through the public EDGAR database¹.

More specifically, 10-K are extensive annual reports about financial analysis, 10-Q are quarterly earnings reports associated with the earning releases of the company, while 6-K and 8-K refer to unexpected events that took place between deliveries of 10-Q forms. We noticed that 10-Q, 6-K and 8-K filings usually start with some bullet points which are the highlights of the full report content, as shown in Figures 3.5 and 3.4. However, these filings are in most cases really large in size (usually more than 100 pages) and contain more tabular information than narrative. Moreover, even though SEC filings are accessible from the EDGAR database and there are some options of available API’s,

¹https://www.sec.gov/edgar/searchedgar/companysearch.html
3.2. Company filings

![News Corp](image)

**Figure 3.5:** Example of 10-Q filing of News Corporation. The bullet points inside the red frame concern the key highlights of the full content of the report. Available at: [https://newscorpcom.files.wordpress.com/2020/05/q3-2020-earnings-press-release_final_05.07.2020.pdf](https://newscorpcom.files.wordpress.com/2020/05/q3-2020-earnings-press-release_final_05.07.2020.pdf)

the text is not available through them. In order to collect the body of the report which is generally given as PDF format, you need to extract it from HTML. Hence, as there is not an explicit structure, it is difficult to collect this type of data. In conclusion, all the above restrictions complicate the procedure of data collection.

3.2.2 FNS Workshop

*Financial Narrative Summarization (FNS 2020)* (El-Haj et al. 2020) is a workshop that concerns the task of automatic summarization of company financial reports. This workshop provides approximately 3,800 annual reports, along with their gold-standard summaries from annual filings of U.K. companies. The dataset is split into training, validation and test set. At first, only the training and validation sets are
provided from the workshop. The gold-standard summaries, as well
as the full text are extracted from PDF documents. However, these ex-
tracted summaries are part of the annual reports. More specifically, it is
clear that the summaries have not been created independently but they
belong to the report as a section. Therefore, experts have been asked
to define which section of the report summarizes the basic points.

According to the FNS 2020 workshop, this task can be handled ei-
ther as an abstractive or extractive summarization problem, depend-
ing on the model that will be used. However, taking into considera-
tion the extractive character of the training summaries, it is likely for
the model to learn copying sentences instead of creating new ones.
From this perspective, this challenge is not compatible with the re-
quirements of this thesis, as we aim to create an abstractive summa-
ization model.

3.3 Our dataset

The existence of a complete dataset has a great impact in training and
evaluating a summarization model. As has already been mentioned
in chapter 2, a considerable amount of datasets, covering a wide va-
riety of topics, has been created for this purpose. To the best of our
knowledge, financial summarization methodologies have not yet been
studied enough, creating an imperative need for building a financial-
focused dataset.

In the previous section, we examined carefully some of the avail-
able financial data sources, accessing the strengths and the limitations,
which each one presents. Taking all the relevant parameters into con-
sideration, we conclude to choose Bloomberg for the three following
reasons: a) the existence of an API, which makes the collection process
fast and straightforward, b) the presence of some key bullet-points in
the beginning of the document which can serve as a summary and c)
the manageable size of the articles. The handling of company filings,
which require more complex methodologies, has to be left for future
research.
3.3. Our dataset

In this thesis, we propose a new summarization dataset which consists of 2,096 financial articles from Bloomberg along with human written bullet points, which briefly summarize the key-elements. In this section, we will describe the data collection process and the preprocessing steps we followed.

3.3.1 Data Collection

Our dataset contains 2,096 Bloomberg articles of financial interest accompanied with human-written abstractive single or multi-sentence summaries. As it is stated in the previous chapter, even though Bloomberg provides an open API, a paid subscription to Bloomberg products is required, in order to have access to the full content. To overcome this issue, we use the Bloomberg Market and Financial News API\(^2\), which is provided freely from RapidAPI\(^3\).

RapidAPI is a huge API’s marketplace which allows users to connect and subscribe to different API’s from various sources, in order to ease data collection and testing processes. It allows 500 requests per month, giving access to the content of full articles which deal with financial and economic issues. We collect articles from different financial categories including, among others, stock, markets, currency, rate, cryptocurrencies and industries.

The response of a request is a simple structured JSON file, which consists of various field about article content and metadata. As we are interested only in the full text of the article as well as the abstract we collect these two fields. Some articles do not contain abstract bullet points, hence we don’t include them in our dataset.

3.3.2 Preprocessing

Unlike other NLP tasks, which require demanding preprocessing e.g., removing stopwords, numbers, and special characters, lower casing etc., summarization usually does not require such processes. Summarization models learn to generate phrases and sentences by exploiting all the linguistic knowledge of the text including word sequence,

\(^2\)https://rapidapi.com/apidojo/api/bloomberg-market-and-financial-news
\(^3\)https://rapidapi.com
numbers, upper or lower casing, etc. For this reason, the preprocessing pipeline that we follow is restricted to the removal of the noisy examples.

Table 3.2: Dataset Statistics

<table>
<thead>
<tr>
<th></th>
<th>Initial</th>
<th>Preprocessed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Document Length</td>
<td>676</td>
<td>669</td>
</tr>
<tr>
<td>Average Target Length</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Minimum Document Length</td>
<td>20</td>
<td>79</td>
</tr>
<tr>
<td>Maximum Document Length</td>
<td>3758</td>
<td>2537</td>
</tr>
<tr>
<td>Number of One-Sentence Target</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Total Documents</td>
<td>2120</td>
<td>2096</td>
</tr>
</tbody>
</table>

As we have already discussed, our datasets consists of 2,096 articles, with average input document size 669 tokens and corresponding average target summary 23 tokens, specifically one or two sentences.

In order to reduce noisy data and keep only same-structured examples, we remove some outliers of the dataset, namely extremely small (under 70 tokens) or large (over 3000 tokens) input size documents. Furthermore, we noticed that the target examples with only one sentence are very limited. For this reason, we also remove one-sentence target examples. Some statistics of our final dataset are shown in Table 3.2.
Chapter 4

Human-Centered Financial
Summarization

In this chapter, we provide a description of the experiments that we conducted on the new dataset of Bloomberg articles and we explain the proposed interface functionality. We experiment with different model setups, trying out various parameters to provide an analysis into different levels. We combined all the above aspects into a simple interface in order to demonstrate the value of human-centered summarization.

In order to carry out the experiments, we use one of the already existing in the literature summarization model: PEGASUS. As we mentioned in Chapter 2, PEGASUS is based on the architecture of Transformers. Our experiments follow three major directions: a) we aim to optimize the model parameters in order to acquire the best summaries for our dataset as far as quality and structure are concerned, b) we aim to understand the model’s decisions allowing for better explainability by visualizing attention weights for the encoder layers, and c) we attempt to provide different suggestions of the same summary in order to better cover the preferences of the user exploiting the properties of the diverse beam search algorithm.

In this chapter, we describe briefly the experimental setup of the model we used, the different conducted experiments and we go one step further by analyzing the explainability of the output summaries as well as examining the user interactivity in the summarization process, allowing for a human-centered summarization approach. Finally, we describe the interface functionality.
4.1 Experimental Setup

To conduct our experiments we use PEGASUS, which consists of 16 layers for both the encoder and decoder layers with 16 attention heads and 1024 hidden size. We experiment with different input token sizes, ranging from 256 to 516 tokens, keeping the output length of tokens constant to 32. We assume that the generated summaries will certainly not exceed the limit of the 32 tokens, as the target summaries of the available training examples consist of 24 tokens, on average.

PEGASUS uses a subword tokenization algorithm in order to tokenize the initial sentences of the document. Having said that, the tokens are divided into subwords, following the rule that in order to capture the meaning of infrequent words, the tokens should be split into smaller subwords, while common words don’t require any particular segmentation.

The optimization algorithm used by the model is AdaFactor with a dropout rate of 0.1. AdaFactor is a variant of the Adam optimizer, which is an adaptive learning rate optimizer that can be used in order to replace the common stochastic gradient descent algorithm. AdaFactor is suggested due to its sublinear memory cost. Also, we use the package of ROUGE metrics in order to evaluate the predicted summaries.

All the experiments were conducted with Google Colab, using the available free GPU resources. Due to the limited resources during training, we use a considerably small batch size of 6, with some modifications to the learning rate ranging from $1 \times 10^{-5}$ to $1 \times 10^{-4}$. The code of PEGASUS has been implemented with Tensorflow 1, however we used the PyTorch implementation of Hugging Face, for our experiments.

4.1.1 Downstream Datasets

We use two different downstream datasets for fine-tuning: XSum and CNN/Daily Mail. The target summaries of XSum are much closer to the Bloomberg dataset that we have collected, due to their abstractive character. On the other hand, the CNN/Daily Mail dataset contains
4.1. Experimental Setup

<table>
<thead>
<tr>
<th></th>
<th>CNN/Daily Mail</th>
<th>XSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Size</td>
<td>R-1</td>
<td>R-2</td>
</tr>
<tr>
<td>400</td>
<td>24.90</td>
<td>7.00</td>
</tr>
<tr>
<td>512</td>
<td>23.29</td>
<td>6.78</td>
</tr>
</tbody>
</table>

more extractive summaries, which are based to a large extent, on existing words, phrases or whole sentences from the original document. These datasets are already fine-tuned by PEGASUS with 210K and 30K training steps for the CNN/Daily Mail and XSum, respectively.

We fine-tune the two different models in our dataset for a maximum of 10K training steps, in order to avoid the model from overfitting. The best checkpoints of our models were observed for 3K and 1K steps for XSum and CNN/Daily Mail, respectively.

4.1.2 Input Source Size

We experiment with different input size tokens ranging from 256 to 512 in order to examine the effect of truncated text to the final summary in both XSum and CNN/Daily Mail setups. We expect that the longer size of the initial document will have a positive effect on the quality of the generated summaries.

Comparing the results of the ROUGE metrics in Table 4.1 for 256, 400 and 512 input size, we can observe that actually the larger text does not help significantly the model for providing better quality summaries. This can be explained, taking into consideration that most of the time the most important sentences are in the beginning of an article. This observation allows us to decrease the input token size to 256 and continue our experiments with an increased batch size due to the GPU limits of Colab.
4.2 Human-centric Approach

In this section, we describe the directions we follow to create a human-centric summarization system. We first attempt to make the model’s decisions more understandable to the human providing insights of the complex mechanisms that are utilized by the model by visualizing the self-attention layers of the encoder. Furthermore, we provide various suggestions of diverse hypotheses between possible summaries in order to meet special preferences of different users. We can further train our model with these summaries to achieve better performance and improve score metrics.

4.2.1 Interpretation of Attention Weights

Despite the high rise of DL methods, the limited transparency of the final model’s decisions as well as the difficulty to easily comprehend the model’s behavior, still remains a major issue. DL approaches have a complex inner structure consisting of multiple layers, which are often combined with other complex mechanisms, such as attention mechanisms. The high complexity of these neural approaches makes them operate as black boxes, limiting their explainability and restricting their transparency in contrast to other traditional Machine Learning techniques, such as logistic regression and decision trees, which produce explainable predictions through highly transparent processes during the training phase. As a result, the predictions of such neural approaches are not always comprehensible to humans.

In this thesis, we use a Transformers-based model. For this reason, it is not easy to understand the factors that led the model to give the final predictions. In order to analyze the behavior of the model we examine the self-attention mechanism, which is used for capturing the most important tokens for the final generation. For this purpose, we examine the activation weights of the encoder layers in order to investigate if they can provide us useful hints to explain the behavior of the model.

Even though the use of the attention mechanism as an interpretability metric is a controversial issue (Serrano and Smith 2019; Vaswani
et al. 2017), we attempt to visualize the weights of the encoder self-attention layers in order to give possible explanations and reflect the factors that possibly drive the behavior of the model. In particular, the encoder self-attention activation weights can possibly provide an overview of the most important words of the original document. For example, we expect that a high-value weight will be associated with a significantly important word, while a low-value weight will indicate a not so powerful - for the final generation - token.

In order to visualize these attention weights, we follow a straightforward procedure. First, we extract the tensor of the activation encoder weights of the model, which is a tensor with 4 dimensions of size $L \times H \times I \times I$, where $L$ is the number of the self-attention layers of the encoder, $H$ is the number of the attention heads of each layer (Note that each one of them may pay attention to different parts of the original text) and $I$ represents the input size of the document. For example, for an input size of 512 tokens of the large model of PEGASUS, where $L = 16$, $H = 16$ and $I = 512$, the tensor size is 16x16x512x512.

Towards an overview of all the attention weights of the model, we average the weights of each layer and attention head and we conclude with a vector of size $I \times I$. Each row represents the weight of the token, that is the attention of the word to every other word of the original document. As a result every row of this vector sums to 1. We then add the values of each column to extract the final weights for each token, assuming that if a word is attended by many other words, this will lead to a higher final weight and as a result to a more important token for the generation of the summary. Finally, we highlight the words, according to their weights.

An example of such visualization of the input text is shown in Figure 4.1. We can observe that even if some highlighted tokens are indeed important and also appear in the generated summary, it is not very clear where the model focuses and what are actually the factors that drive its decisions.

However, some useful hints about the behavior of the model are observed. Visualizing each layer of the encoder, instead of just averaging them, leads to the finding that the first layers of the model tend to focus more on whole sentences and phrases, while the last layers tend to learn the natural language syntactical and lexical rules by focusing
Figure 4.1: Example of encoder attention weights of the Cnn/DailyMail model with generated summary:

Government considers again delaying increase in pension contributions. Unions say pay rises since the last delay have been poor.

more on prepositions and articles. This is a significant observation regarding the way a model perceives the language and how it finally learns to generate it. An example of this finding is shown in Figure 4.2

4.2.2 Diverse Hypotheses During Inference

A major issue in existing summarization language models is that they attempt to learn from training examples with low quality summaries. (Bommasani and Cardie 2020). As a result the final generated model’s summary does not always focus on the right parts of the original document and sometimes does not meet the requirements of humans. Furthermore, the most widely used inference algorithm i.e. beam search, despite the good quality between the top probable beams, still suffers from a lack of diversity between the possible summary suggestions, failing to produce diverse targets to satisfy different user’s preferences.
4.3 Results

In order to overcome these limitations, in this thesis, we use diverse beam search to provide different hypotheses for the final summary to the user so he can interact with the summarization system by approving or disapproving the generated summaries or even submitting his own suggestion. We can further use the user-approved generated summaries to customize and fine-tune the model again. In this way, we can ensure the involvement of the human factor in the whole process, achieving improvement in the quality of the model by increasing our dataset for further fine-tuning and also personalizing the model to human suggestions.

An example of the differences between the beam search algorithm and the diverse beam search in our model is shown in Table 4.2. For a beam width=3, the generated summaries of beam search are almost identical differing only to some tokens, while the diverse beam search achieves high diversity among the beams, replacing whole sentences with other possible suggestions.

4.3 Results

We compare the results of the model with and without fine-tuning to our dataset for both XSum and CNN/Daily Mail models. For all the experiments we use the same test and validation dataset which
Chapter 4. Human-Centered Financial Summarization

Table 4.2: Example of the different summaries generated from beam search and diverse beam search algorithm.

**Reference summary:** Central bank leaves asset purchases, interest rates unchanged. Currency surged as president spoke and is near a two-year high.

**Beam search summaries:**
- Currency jumps to highest in more than a week, close to two-year high. ECB keeps bond-buying and deposit rates unchanged.
- Currency jumps to highest in more than a week, close to two-year high. ECB keeps bond-buying program, deposit rate unchanged.
- Currency jumps to highest in more than a week, close to two-year high. ECB keeps bond-buying, deposit rate at 0.5%.

**Diverse beam search summaries:**
- Currency jumps to highest in more than a week after ECB meeting. Lagarde says officials will ‘carefully assess’ exchange rate.
- U.S. central bank keeps bond-buying, deposit rate unchanged. Lagarde says officials will assess exchange rate impact.
- Common currency jumps to the highest in more than a week. ECB keeps bond-buying and deposit rates unchanged.

consists of 99 and 98 examples, respectively. The derived summaries between original models (zero-shot summarization) and further fine-tuning to our dataset are significantly improved both in quality, structure and terminology, as shown in Table 4.3. In particular, we achieve an improvement of 9.75, 4.59, 7.51 and 9.33 for ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-S, respectively for the XSum model.

We can notice that both models adapt very fast to the new dataset and are capable of providing salient summaries specialized in financial terminology. In particular, the XSum model requires 3,000 training
### 4.4 Interface

In order to demonstrate the value of human-centered summarization, we combine all the aforementioned aspects into a simple interface using interactive Jupyter widgets\(^1\) in Colab\(^2\). We use a MySQL\(^3\) database running in a Docker\(^4\) container which is deployed to Heroku\(^5\) in order to store the summaries which are approved by the user. This section presents the interface functionality. A screenshot of our proposed interactive system is shown in Figure 4.3.

---

\(^1\)https://ipywidgets.readthedocs.io
\(^2\)https://colab.research.google.com
\(^3\)https://www.mysql.com/
\(^4\)https://www.docker.com/
\(^5\)https://www.heroku.com/
**Table 4.4:** Example of generated summaries before and after fine-tuning for CNN/Daily Mail and XSum

<table>
<thead>
<tr>
<th>Reference summary: Survey shows traders and analysts expect futures to climb. Shrinking gas supply seen as continuing to be supportive.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CNN/Daily Mail generated summary without fine-tuning:</strong> The most-active futures contract on the New York Mercantile Exchange slumped below $2 per million British thermal units in recent days. But futures for delivery in December through March continue to trade above $3. Almost half of respondents in a Bloomberg News survey of traders and analysts say winter prices could rise even higher.</td>
</tr>
<tr>
<td><strong>CNN/Daily Mail generated summary after fine-tuning:</strong> Almost half of respondents in survey say winter prices could rise. Futures for delivery in December through March continue to trade above $3.</td>
</tr>
<tr>
<td><strong>XSum generated summary without fine-tuning:</strong> It’s been a rough week for energy markets.</td>
</tr>
<tr>
<td><strong>XSum generated summary after fine-tuning:</strong> Almost half in survey see gas trading above $3 this winter. Abundant supply, milder weather seen eroding glut.</td>
</tr>
</tbody>
</table>

We explain here the steps that the user can follow in order to use our proposed system:

1. First, the user enters the text that wants to summarize into the text box. We provide a demo text in order that the user can experiment with the different provided features.

2. The user can visualize the weights of the input text choosing the desired color for the visualization by pressing the button *Visualize Weights*.

3. The user can generate the summaries from the input text using different decoding strategies by selecting the desired decoding strategy from the dropdown menu and pressing the button *Generate Summaries*. The available decoding strategies are greedy search, diverse beam search, random sampling, top-k sampling and top-p or “nucleus” sampling. For all the decoding strategies except from greedy search we provide hyperparameters which can be tuned according to the user’s preferences.
4. Finally, the user can also mix and match from the generated summaries, combining sentences of them in order to synthesize the final summary. After the composition of the summary, the user can also submit it to the database by pressing “Save Sample”. These samples, which are approved by the users, can be used further by our system in order to improve the quality of the underlying summarization model.

The code and the usage instructions of our system is publicly available⁶.

Chapter 5

Conclusions and Future Work

Automatic summarization has definitely attracted the attention of many researchers during the last decades. The explosive growth of the available online information sources, such as newswire articles, blog posts and forums, has provoked the need of appropriate summarization tools. The proper handling of all these documents today is essential more than ever, allowing for identifying and retrieving the huge volume of online information, accurately and promptly.

However, despite the development of various high-performance summarization methodologies which are specialized in newswires articles for a range of topics, such as lifestyle, finance, health and environment, the development of an explicit financially-focused model has not been studied enough yet.

A high-quality summarization tool is significantly important for financial analysis. For example, an investor typically will have to search and process a high number of market analyses and financial reports, before an investment. A summarization system could allow for retrieving only the highlights of each report, providing an overview of the market conditions at the moment, avoiding time-consuming reading and search procedures.

Furthermore, despite the increasing success of the neural summarization methods, the low transparency of the model’s behavior as well as the limited human involvement in the summarization process, restrict the trust and the engagement of the user in the whole process. A human-centered summarization system should be able to generate predictions that are both transparent and understandable for the human and provide diverse and customized summary suggestions according to the user’s needs.
Chapter 5. Conclusions and Future Work

5.1 Conclusions

In this thesis, we proposed a human-centered summarization system allowing for the active involvement of a human in the summarization processes, providing at the same time insights about the model’s behavior.

Moreover, we collected a new dataset which consists exclusively of 2096 financial articles. The pre-trained PEGASUS model that we build upon, adapts very fast to our new dataset and only after a few training steps achieves an increase of 9.75, 4.59 and 7.51 in terms of ROUGE-1, ROUGE-2 and ROUGE-L, respectively in our dataset, after the fine-tuning.

We further investigated the behavior of the model by providing insights about the self-attention mechanism of the encoder through examining the process of learning (first learning the phrases and terms and later syntactic rules).

Finally, we proposed alternative suggestions of the summaries to the user in order to meet different users preferences. This approach allows for increasing our data collection so we can retrain and fine-tune our model after a considerable amount of collected user-approved summaries. All the above aspects are combined into a simple human-centered summarization interface using Jupyter interactive widgets.

5.2 Future Work

One critical issue which has been left for future research is the development of appropriate evaluation metrics for abstractive summarization based on semantics, for example using BERT embeddings.

We also plan to explore further the human-centered aspect of summarization systems by improving and customizing the model according to user’s preferences, utilizing different techniques such as one-shot learning or personalizing only some of the layers of the encoder-decoder model for each user.

Finally, we consider collecting financial company filings which requires different handling from the newswire articles. In order to analyze and handle the complex structure of such reports, we consider
applying extractive techniques, such as clustering or similarity threshold methods, to reduce the size of the original text.
Bibliography


Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio


