EXTRACTIVE SUMMARISATION OF UK ANNUAL REPORTS

A REPORT SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF BACHELOR OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2023

Vladislav Yotkov

Department of Computer Science

Contents

Word Count: 12345

List of Tables

List of Figures

Abstract

Although there has been considerable progress in Natural Language Processing (NLP) over the years, it has not fully reached the Accounting and Finance (AF) industry. In the meantime, companies worldwide produce vast amounts of textual data as part of their reporting packages to comply with regulations and inform shareholders of their financial performance. The glossy annual report is such an example, widely read by investors but it also tends to be quite long. Inspired by the Financial Narrative Summarisation (FNS) 2021 Task, we will design an Automatic Text Summarisation (ATS) system for the narrative parts of UK financial annual reports. With this goal in mind, we will implement and explore the following models for Extractive Text Summarisation (ETS): 1. custom Recurrent Neural Network (RNN), 2. fine-tuned FinBERT. In terms of evaluation, we will use the ROUGE metric to compare the performance of these models against standard ATS baselines: TextRank, and LexRank.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

1 Introduction

1.1 Financial Reports

Due to international regulations, companies are obliged to report their periodic performance (annual, bi-annual, quarterly) to various regulatory authorities[1] and other users (e.g., corporate stakeholders, investors, customers, suppliers, etc.). These reports contain essential information about the operations and finances of a business and are crucial for making informed decisions (from a user perspective), but are different in regulatory forms. For example,

1. 10-K reports filed to the SEC[2] and accessible through their Electronic Data Gathering, Analysis, and Retrieval[3] (EDGAR) system are only for US registered businesses. They follow a standardised template and are plain text, which makes them particularly easy for automated large-scale research ([EHAR+19]). Also, the contents of these reports are strict, requiring solely five information sections[4].

2. UK annual reports, regulated by the UK's Financial Reporting Council (FRC), are typically the primary annual reporting method (also provided as PDF files). Unlike the 10-K, they are glossy and more stakeholder-oriented and enjoy unlimited discretion over non-mandated content ([EHAR+19]) (e.g., photography and company brand material, non-mandatory narrative sections, etc.). However, these are more challenging for automated processing due to their variable section structure, formatting, and rich visual representations.

[1] Regulation authorities worldwide:

- Securities and Exchange Commission (SEC) in the USA
- European Securities and Markets Authority (ESMA) in Europe
- Financial Reporting Council (FRC) in the UK
- International Financial Reporting Standards (IFRS) in 167 jurisdictions worldwide

[2] https://www.sec.gov

[3] https://www.sec.gov/edgar

[4] (a) Business Overview (b) Risk Factors (c) Management's Discussion and Analysis of Financial Condition and Results of Operations (MD&A) (d) Financial Statements (e) Supplementary Disclosures

1.2 UK annual reports

The annual report is the primary corporate disclosure legally required for public companies by regulatory authorities. While it *does not have a rigid document structure* like the 10-K, it typically has a *narrative component*[5] and the financial statements (at the rear).

As we outlined in Section 1.1, UK annual reports have the following inconvenient properties with regard to large-scale text understanding.

- They are very long documents. Throughout the years, their average length has been increasing significantly with the number of pages rising 57% for the median report from 2003 to 2016 (47 to 74 pages, respectively) ([LY19]), due to additional regulations between 2006 and 2008 ([EHAR+19].
- They have variable nomenclature. From firm to firm, naming conventions vary "dramatically", with more than 20 unique titles for various sections (e.g., Chair's letter to shareholders, Management Commentary) ([LY19]).
- They incorporate embedded info-graphics. While domain experts hail the integration of highly interactive elements into corporate reporting ([KB16]), the compilation to PDF makes the task of analysing such unstructured documents automatically even harder ([LY19]).

These challenges motivate the work of [EHRY+19] who (a) established a set of 8 generic section headers[6] and (b) built the CFIE-FRSE[7] extraction tool that converts a text-based PDF annual report to simple text.

1.3 NLP in Accounting and Finance

The relevance of this project should also be understood from the perspective of the development of Natural Language Processing (NLP) in the Accounting and Finance (AF) domain. As outlined

[5] The narrative component of a UK annual report typically consists of 1. Management's Commentary 2. Letter to Shareholders 3. Corporate Governance Statement 4. Auditor's Report 5. Remuneration Report 6. Business Review 7. Environmental, Social, and Governance (ESG) Report 8. Risk Management Report

[6] (a) Chairman Statement (b) CEO Review (c) Corporate Governance Report (d) Directors Remuneration Report (e) Business Review (f) Financial Review (g) Operating Review (h) Highlights

[7] The CFIE-FRSE stands for Corporate Financial Information Environment - Final Report Structure Extractor. It is publicly available at `https://github.com/drelhaj/CFIE-FRSE` and it can be used to convert English, Spanish and Portuguese annual reports.

in [Ell98], investors' trust in the accountability of businesses would be based no longer as much on just the financial statements, but also on more descriptive narratives that define strategy and planning of resource use. While some recognise the importance of understanding in-domain textual information ([L+10]), others like [EHRW+19] report that the industry is still doubtful and cynical about the NLP applications in the analysis of financial market disclosures. Furthermore, the latter also observe that AF researchers rely extensively on bag-of-words models, which are *not sufficient to encode complex contextual and semantic meaning* (especially in a domain with such *specialized language*). As for ATS [CHW19] is said to be the single AF study into disclosure summarisation. It demonstrates that machine-generated summaries are less likely to bias positively investor decisions compared to managerial ones. Therefore, this only confirms the existence of a wide gap in NLP applications in Accounting research, which further motivates our work.

1.4 Financial Narrative Summarisation 2021 (FNS21) Task

The FNS Task is part of the annual Financial Narrative Processing (FNP) Workshop [8] organised by Lancaster University since 2018, which aims to:

- encourage the advancement of financial text mining & narrative processing
- examine methods of structured content retrieval from financial reports
- explore causes and consequences of corporate disclosure

as stated in their inaugural proceedings [9].

For that purpose, they produce datasets of extracted narratives (with the help of the CFIE-FRSE tool) from annual reports of UK companies listed on the London Stock Exchange (LSE).

In their FNS21 Task, there were $3,863$ such reports (Table 1), while the average length was reported at 80 pages, and the maximum of more than 250 pages ([LV21]).

Additionally, for every report, there were at least two gold summaries situated in the annual report itself [10] The workshop's goal was to build ATS systems that generate a single summary

[8] https://wp.lancs.ac.uk/cfie/

[9] https://wp.lancs.ac.uk/cfie/fnp2018/

[10] The gold summaries being already in the annual report is not problematic because these reports are already written by domain experts who know how to summarise the financial state of a company. Hence, multiple sections/paragraphs could achieve this thoroughly, and the organisers have identified & extracted them manually with the help of the

for an annual report, no longer than $1,000$ words (almost just as long as the gold summaries on average).

Data Type	Training	Validation	Testing	Total
Report full text	3,000	363	500	3,863
Gold summaries	9,873	1,250	1,673	12,796

Table 1: FNS21 Data Split

We acknowledge that due to the scarcity of publicly available financial data this third-year project could not have been possible without the kind permission of the FNP organisers to use the training and validation datasets from their FNS21 Task ([EHRZ21]).

professional writers of the individual reports. At this moment, one can begin to doubt the point of applying ATS techniques, but due to the *lack of rigid document structure*, *it is not trivial to automatically find these text excerpts with heuristic methods*. Furthermore, we can formulate this challenge as finding the latent features of a summarising (i.e., "to-be-in-the-summary") sentence, highlighted as one of the fundamental advantages of NLP in AF research ([LY19], [EHRW+19]).

2 Background

2.1 Tf-Idf

Tf-Idf (Term Frequency - Inverse Document Frequency) is a statistical technique intended to reflect the importance of a word to a document in a corpus. The term frequency is the number of occurrences of term t in document d ([Luh57]), but it is not enough to capture the importance of a term due to *all terms being considered equally important* ([MRS08]). Meanwhile, the *document frequency* (df_t) is the number of documents in the corpus containing the term t, which evaluates how *common* and *unimportant* a term is ([LRU20]). Therefore, *idf* becomes $\frac{N}{\mathrm{df}_t}$, where N is the corpus size, and then the tf-idf term weight $w_{t,d}$ is calculated as the product of the two - $\mathrm{tf}_{t,d} \cdot \mathrm{idf}_{t,d}$ ([JM00]). The latter represents now the importance of a term t but normalised by its *commonness*. Tf-Idf is often used as a weighting factor in information retrieval and text mining. It has also been successfully integrated in summarisation methods like LexRank ([ER04]) (Section 2.10) and in FNS competitive systems ([KVL21, EHO22]).

2.2 Word Embeddings

Historically, to represent a token (i.e., word) w_i in a vocabulary V numerically, we define a one-hot-encoding vector of all zeroes except of a one at the index of the word w_i in V (i.e., i).

The results are sparse individual word vectors being orthogonal to each other which 1. waste memory (each word is a $|V|$-sized vector, hence a total of $|V|^2$ for all tokens) and more importantly 2. fail to encode semantic similarity due to their cosine similarity being always zero.

Traditionally, AF research has represented an input text with the help of bag-of-words (BOW) models which can be viewed from the 1. the binary perspective - represent a whole document d as a binary vector containing ones for all words w_i occurring in d from V, 2. the term frequency perspective - encode number of word occurrences in documents instead of binary representation ([XCWS13]), and 3. the tf-idf perspective - extend the latter to penalise ubiquitous terms ([Jon72]). Nevertheless, these vectors are very sparse and unable to encode more complex contextual and semantic meaning.

To address these shortcomings *short*[11] and *dense*[12] word embeddings like Word2Vec ([MCCD13])

[11] i.e., with a small number of dimensions

[12] i.e., continuous real-numbered values instead of 0/1s

and FastText ([BGJM17]) have been developed. In [MCCD13] the authors manage to condense the vector space and ensure that word representations have *multiple degrees of similarity* ([MYZ13]) (e.g., semantic - the meaning of words, morphological - structure of sub-words, etc).

Furthermore, the proposed models - CBOW (Continuous Bag of Words[13]) and Skip-gram evidently capture subtle semantic relationships and allow intuitive arithmetic operations as shown in the popular analogy: $\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \approx \overrightarrow{\text{queen}}$[14].

The intuition for the two Word2Vec models is that in CBOW, the context (i.e., the surrounding tokens) is used to predict the middle token, while in skip-gram, the input token is used to predict the context (i.e., the surrounding tokens) (Figure 1).

Meanwhile, internally, the context prediction is cast as a binary classification task with positive examples being the target word and its surroundings, whereas the negatives ones are generated through random sampling from the dictionary. Then, the CBOW embeddings are the learned weights of a logistic regression classifier with future and history words (i.e., the context window) as the input and the goal of correctly classifying the word in-between. In contrast, the Skip-gram uses the middle word as an input to the classifier and predicts the individual context words around it.

A downside to the Word2Vec models is that they cannot handle out-of-vocabulary (OOV) word tokens, i.e., they cannot generate an embedding vector for words missing from the training data, which is crucial in real-life problems with *noisy* input or morphologically rich languages. For this reason [BGJM17], propose FastText as an extension to the Skip-gram model that makes use of character-level information to deal with unknown tokens. Here, each word is itself a bag of character n-grams which captures meaning of prefixes, suffixes, and morphemes. Additionally, two symbols are further introduced to mark the beginning and the end of a token, and help differentiate between sub-words and short words. For example, the character trigrams of the word *believe* are *<be, bel, eli, lie, iev, eve, ve>* where the sub-word *lie* is different from the word token *<lie>*.

Therefore, the final target word embedding is the sum of its constituent character n-grams which are learned via the Skip-gram model. This makes FastText very convenient for representing unknown words as the sum of *static* constituent n-grams ([JM00]).

[13]CBOW naming is derived from 1. the continuous distributed representation of the context and 2. the projection layer being shared across context words, i.e., the order of words does not affect the projection (similar to how bag-of–words model fails to encode word order).

[14]For a formal explanation on how analogies are realised in word embeddings we direct the readers to [EDH19]

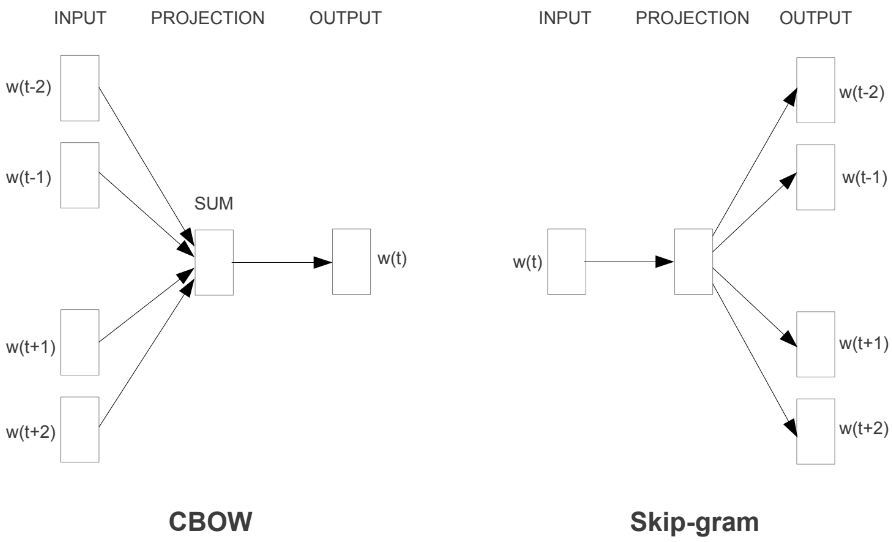


Figure 1: Word-from-context and context-from-word prediction in CBOW and Skip-gram, respectively ([MCCD13])

.

Nevertheless, when it comes to domain-specific problems, general pre-trained word embeddings do not perform very well [RZP21]. demonstrate that even state-of-the-art embedding models like Google's Word2Vec(skip-gram)[15] and Facebook's FastText(skip-gram)[16]trained on 100 billion and 16 billion words, respectively, struggle to understand financial language like 1. *apple* standing for the company *Apple*, 2. ticker analogies, e.g., *amazon* is to *X* as *microsoft* is to *msft*, 3. grouping company name to ticker, exchange and country. In the same paper, the authors propose using the same algorithms (i.e., CBOW, Skip-gram, and FastText) but training solely on financial data instead - 15 years of financial news from the Dow Jones Newswires Text News Feed database, to produce the FinText models. They report a substantial increase in performance in and sensitivity to detecting financial jargon and relationships. In our project we acknowledge that a purpose-built financial word embedding (trained on proprietary data) will be more beneficial and more suited for the task of text summarisation of annual financial reports, which is why we select FinText as our preferred model.

[15] https://code.google.com/archive/p/word2vec/
[16] https://fasttext.cc/

2.3 Recurrent Neural Networks (RNNs)

The vanilla RNN is a basic type of RNN architecture designed for processing sequential data. It learns temporal patterns from the initial data by looping over the hidden layers which allow information to persist (i.e., they serve as a network memory) ([Ola15]). The key component is the recurrent hidden state h_i (Figure 2a) updated at each time step using input data and the previous hidden state (Eq.1). This allows the RNN to capture contextual information and temporal dependencies in the sequence. However, due to the inherent vanishing and exploding gradient problems with the vanilla RNNs, they have limited ability to learn long-term dependencies ([BSF94]). To resolve these issues more advanced RNN architectures like LSTMs and GRUs have been developed.

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h)^{17} \tag{1}$$

$$y_t = g(W_{hy}h_t + b_y) \tag{2}$$

The Long Short-Term Memory (LSTM) recurrent neural network has become a ubiquitous method in sequential problems (e.g., language modelling, time series forecasting). This is so because it allows long-term dependencies to propagate through the network with the help of control gates *input* and *forget*, which reduce the effect of the vanishing gradient issue in the vanilla RNN[18].

A more simple variant of the LSTM is the Gated Recurrent Unit (GRU) which combines the *input* and *forget* gates into an *update* gate for a model with fewer parameters and faster training ([CCG21]). Nevertheless, due to the sequential nature of the LSTM, the training process cannot be parallelised across GPUs, i.e., the learning cannot be made quicker by more computational resources.

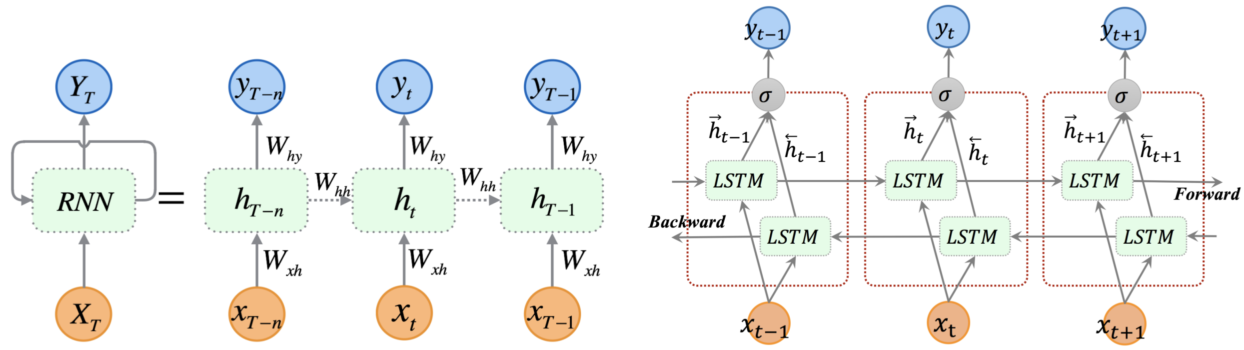
As noted by [GMH13], a limitation of the unidirectional RNNs is that they only make use of previous context in the sequence. To alleviate this and establish more complex relationships between words [SP97], propose a bi-directional architecture consisting of a forward and a backward RNN.

[17] The algebraic formulation of the vanilla RNN has the following variables: 1. x_t is the input at time step t. 2. h_{t-1} is the hidden state at time step $t-1$. 3. h_t is the hidden state at time step t. 4. y_t is the output at time step t. 5. f and g are activation functions for the hidden and output layers, respectively. 6. W_{hh}, W_{xh}, W_{hy} are weight matrices for the hidden-to-hidden, input-to-hidden, and hidden-to-output connections, respectively. 7. b_h and b_y are bias terms for the hidden and output layers, respectively.

[18] We direct readers to [Bay15] where the authors demonstrate that the LSTM's "temporal" gradient is unaffected by the fixed weight factor W of the vanilla RNN that is driving the derivative to zero. This is ensured by the additional architecture unit the *forget* gate, which learns to control the gradient flow in the network.

We can therefore represent the hidden layers (h_i) per time-step (where T is the sequence size) with the following notation: 1. $(\overrightarrow{h_1},\ldots,\overrightarrow{h_T})$ are the forward hidden states from left to right (i.e., $x_1,\ldots,x_T$) and 2. $(\overleftarrow{h_1},\ldots,\overleftarrow{h_T})$ are the backward hidden states from right to left (i.e., $x_T,\ldots,x_1$) (Figure 2b).

Then, for a single word x_i, its respective *annotation* (i.e., condensed representation) is constructed by the concatenation of the forward and backward hidden states - $h_i = [\overrightarrow{h_i^T};\overleftarrow{h_i^T}]^T$ as specified in [BCB16].



(a) Unfolded RNN (b) Unfolded bi-directional LSTM

Figure 2: Unfolded recurrent architectures ([CKW18])

2.4 Encoder-Decoder and Attention

However, to deal with many-to-many sequence-to-sequence problems (e.g., machine translation, speech recognition, abstractive text summarisation) a new type of neural architecture is necessary [SVL14]. and [CvMG+14] introduced the Encoder-Decoder network (Figure 3) with 1. the encoder being an RNN that maps an input sequence $(x_1,x_2,\ldots,x_n)$ to a continuous fixed-length context vector c and 2. the decoder, also an RNN, taking this vector and producing an output sequence $(y_1,y_2,\ldots,y_m)$. They train the two RNNs jointly, maximising the conditional probability of the target sequence given a source sequence, i.e., $p(y_1,\ldots,y_{T'}|x_1,\ldots,x_T)$. For the selection of the neural model the authors had naturally chosen LSTM and GRU, respectively, due to the resolution of the vanishing gradient problem (as discussed in Section 2.3) and for the time being [SVL14] managed to achieve state-of-the-art results in machine translation.

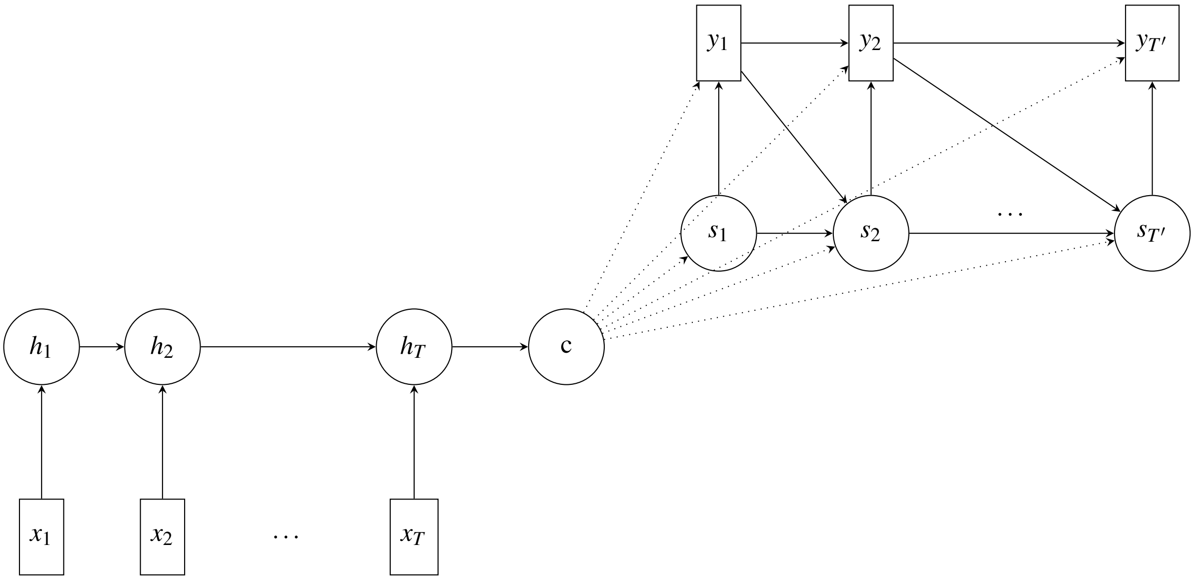


Figure 3: Encoder-Decoder Schema

It is in [BCB16] where the authors suppose that the fixed-length vector results in a bottleneck such that the longer the sequences, the worse the compression performance is for neural networks. Meanwhile, they propose to replace the fixed-length vector with a variable-length one which searches for the most relevant information from the source sequence. This is achieved through an alignment model which accepts as input the produced annotations from the encoder $h_1, \dots, h_T$ (see Section 2.3 for details on how these are generated). Then the context vector c_i which is now distinct for each input word (unlike in [CvMG+14, SVL14]) becomes a weighted sum[19] of the annotations h_j (Eq.3) and each weight α_{ij} is calculated as the normalized attention score (Eq.4). The alignment model is itself a neural network trained jointly with the encoder-decoder system and it evaluates the importance e_{ij} of an annotation h_j in generating a new state s_i and output token y_i (Eq.5). Intuitively, as noted in [GLT20], this attention mechanism computes a weight distribution on the input sequence and *attends* (i.e., assigns a larger weight) to the most relevant parts.

$$c_i = \sum_{j}^{T} \alpha_{ij} h_j \tag{3}$$

[19] And hence it is known as the *additive attention*.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})} \tag{4}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{5}$$

2.5 Transformers

The Transformer ([VSP+17]) is another sequence-to-sequence architecture which follows the overall encoder-decoder architecture ([SVL14]), but differs in the following aspects:

1. *Internal architecture* - instead of RNNs ([CvMG+14]), the encoder and the decoder have multiple identical transformer blocks based on fully-connected feed-forward neural networks with a newly introduced concept of self-attention (Figure 6). Although, prior to that, recurrent hidden layers have been used for compressing sequences into a context vector, the authors of the transformer extend the attention from [BCB16] to generate salient context-aware representations of the input sequence.

2. *Self-Attention and Global context* - while RNN architectures struggle to deal with long-range dependencies (due to the vanishing gradient problem and the last-layer bottleneck [BCB16]), the authors propose *causal*[20] self-attention to allow capturing of global context. This mechanism has two key components (Fig.5):

    - *The scaled dot-product* (Eq.4) computes the attention weights (see Section 2.4) to be used for generating a weighted representation of the input sequence. Each token from the input sequence is represented as a q-query, k-key, and v-value vector (packed together into linearly-projected[21] Q, K, V matrices, respectively). Meanwhile, the dot product computes the similarity score between Q - the current token's focus and K - the context of the other tokens, whereas the scaling factor $\sqrt{d_k}$ prevents against extreme differences in softmax calculation - leading to slow convergence. Once the attention weights have been calculated, they are multiplied with V for a new context-aware representation combining information from other tokens in the sequence. This mechanism

[20] Causal attention is a special case of self-attention where the attention weights are computed only from the past tokens. This is achieved by masking the future tokens in the attention score computation (Figure 7a). We direct readers to [JM00] for a detailed discussion on Transformer language modelling.

[21] The query vectors, key vectors, and value vectors are linearly projected from the input token embeddings using separate learnable weight matrices.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

Figure 4: Attention calculation (Query, Key, Value)

allows the Transformer to learn relationships between any two tokens and hence the significant computational load $O(n^2)$. Nevertheless, since these computations can be performed independently for each token, the entire input sequence can be processed simultaneously unlike in RNN architectures.

- *The multi-head attention* ([VSP+17]) is a mechanism that allows the Transformer to learn multiple representations of the input sequence. This is achieved by stacking multiple attention heads (each with its own Q, K, V matrices) and concatenating their outputs (Figure 5). For each head h_i, the Q, K, and V vectors will be linearly projected with different weight matrices W_i^Q, W_i^K, and W_i^V, respectively. The intuition is that each attention head learns different aspects of the relationships that exist among inputs (e.g., syntactic, semantic, and discourse relationships [JM00]) and the concatenation of the different representations allows the Transformer to learn a richer overall representation of the input sequence.

3. *Positional Encoding* - the Transformer does not have a recurrent hidden layer to capture the sequential nature of the input sequence. Instead, the authors ([VSP+17]) propose adding a positional encoding to the input embeddings as a way to capture positional information.

4. *Training Details* - Because of the non-sequential nature of the Transformer, the heavy self-attention computations can be parallelised on modern hardware - GPUs and TPUs which makes this model easy during training and inference time. Additionally, the proposed dot-product attention is practically much faster and more space-efficient in comparison to the additive one ([BCB16]), due to highly optimized matrix multiplication libraries. Nevertheless, at least two computational drawbacks are that Transformers require an additional positional embedding and also have a quadratic complexity when calculating the self-attention which is extremely expensive for long sequences ([JM00]).

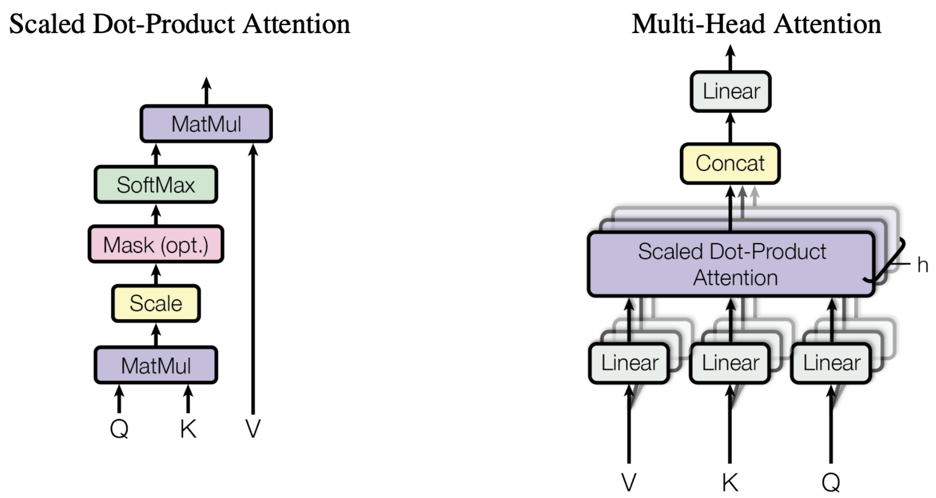


Figure 5: Scaled dot-product and multi-head attention ([VSP+17])

2.6 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a Transformer-based language model that was proposed by Google AI in 2018 ([DCLT19]). It revolutionised the field of NLP by introducing a new pre-training paradigm that outperformed previous state-of-the-art models on a wide range of NLP tasks while being easily applicable to autoregressive generation problems (e.g., abstractive summarization and machine translation). We will discuss the main components of BERT and how it differs from the original Transformer model.

1. *Bidirectional Nature* - [JM00] argue that the uni-directional nature of the Transformer (i.e., causal self-attention) is a drawback that prevents it from capturing the full context of a sentence, especially when applied to sequence classification and labelling tasks. BERT addresses this issue by using a bidirectional Transformer architecture that allows the model to capture the full context of a sentence. This contextualisation is achieved by allowing the self-attention mechanism to range over the entire input as visible from the query-key comparisons in Figure 7b.

2. *Pre-training* - BERT is a model pre-trained on two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

   (a) *Masked Language Model* - BERT uses a pre-training task called Masked Language

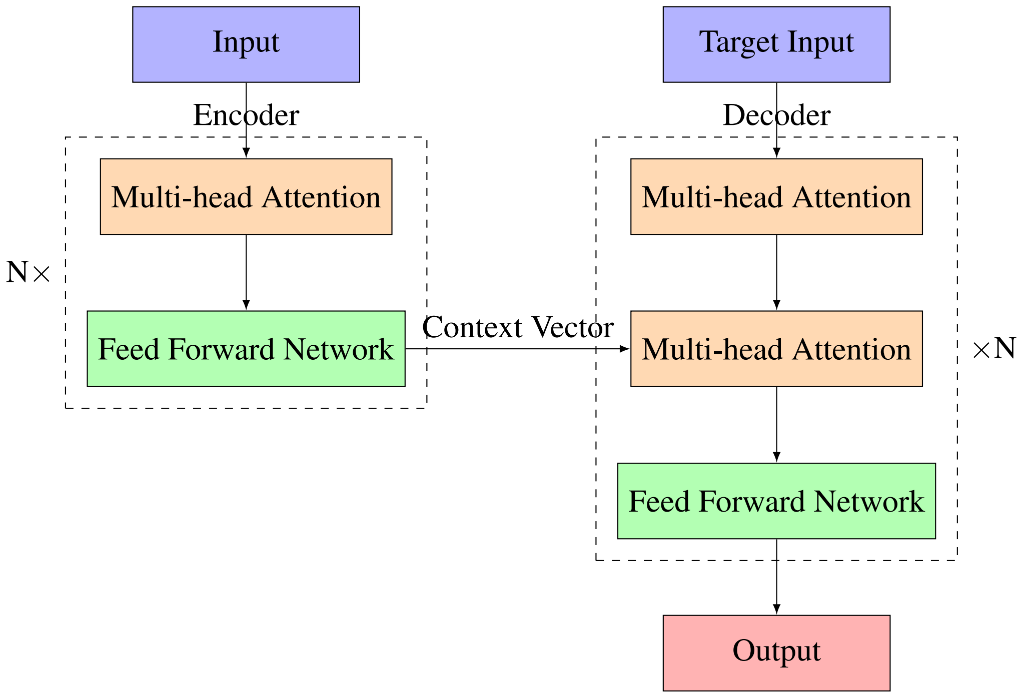


Figure 6: Simplified Transformer encoder-decoder architecture

Model (MLM) to learn the representations of words in a sentence. The authors ([DCLT19]) propose masking 15% of the input tokens at random and then training the model to predict the masked tokens. To make it more accessible for fine-tuning where the [MASK] token is not available, they replace it with a random or with the original token with probabilities of 10% each.

(b) *Next Sentence Prediction* - BERT is also pre-trained on Next Sentence Prediction (NSP) to learn the representations of sentences in a document. To understand the relationship between two sentences, BERT uses a binary classification task where the model is trained to predict whether a sentence B is following another sentence A. The authors ([DCLT19]) capture the sentence structure with the help of two new tokens - [CLS] (added before the sentence pair) and [SEP] (inserted in-between sentences), which are essential for the fine-tuning process (Figure 8). They further propose using 50% of the training data as positive examples (i.e., A and B are consecutive sentences) and 50% as negative examples (i.e., B is a random sentence from the corpus).

3. *Fine-tuning for Sequence Classification* - Unlike in Word2Vec ([MCCD13]) where the word

q_1k_1	$-\infty$	$-\infty$	$-\infty$
q_2k_1	q_2k_2	$-\infty$	$-\infty$
q_3k_1	q_3k_2	q_3k_3	$-\infty$
q_4k_1	q_4k_2	q_4k_3	q_4k_4

(a) Transformer

q_1k_1	q_1k_2	q_1k_3	q_1k_4
q_2k_1	q_2k_2	q_2k_3	q_2k_4
q_3k_1	q_3k_2	q_3k_3	q_3k_4
q_4k_1	q_4k_2	q_4k_3	q_4k_4

(b) BERT

Figure 7: Comparison of query-key dot product representations for Transformer and BERT models.

embeddings are *static* (i.e., related to the single word token only), BERT learns *contextualised word embeddings* which can produce different representations for the same word depending on the context around it. Here, instead of using the output of the last hidden layer (as we do with RNNs and the Transformer), the authors ([DCLT19]) propose a *sentence embedding* y_{CLS} that summarizes the entire sequence of hidden states - the output vector of the model for the [CLS] token. The reasoning is that BERT is pre-trained with the [CLS] token being prepended to the input sequence (during the NSP task) and it can be used as an aggregate representation of the entire sequence. Therefore, fine-tuning for text classification amounts to learning the probability distribution over the possible labels (e.g., 0/1 for a binary task) from the linearly-projected y_{CLS}, i.e. $\text{softmax}(y_{CLS}W_C + b)$, complying with the typical supervised learning paradigm. A slight difference, as [JM00] suggest, is that the backpropagation can affect not only the classifier but also the pretrained language model (resulting in minimal changes in practice).

2.7 FinBERT

FinBERT[22] ([YUH20]) is a pre-trained domain-specific language model based on BERT ([DCLT19]) (Section 2.6) that is trained on a total of 4.9*B* tokens from financial corpora: 1. Corporate Reports 10-K & 10-Q (introduced in Section 1.1), 2. Earnings Call Transcripts (discussing financial performance, business updates, and future expectations), and 3. Analyst Reports (providing an in-depth textual analysis of the company and an earnings forecast). This makes FinBERT a natural choice for our experiments, as the language of the training data is very similar to the one of the UK annual reports (Section 1.2). Furthermore, the authors demonstrate that FinBERT outperforms BERT on

[22] https://github.com/yya518/FinBERT

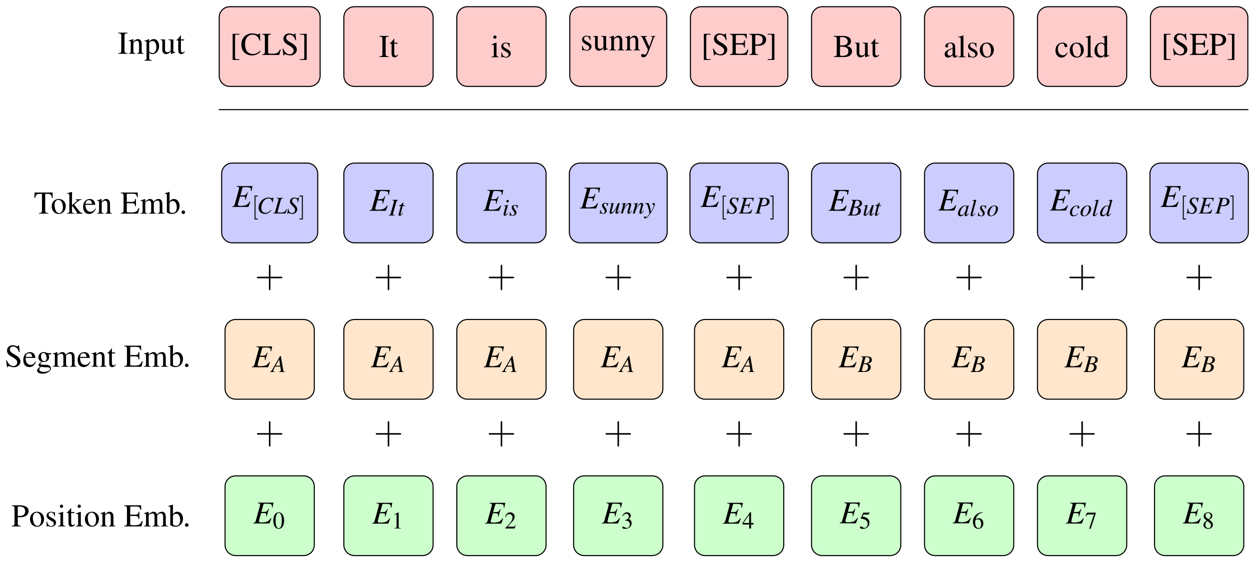


Figure 8: BERT: Input Embeddings

three financial sentiment classification tasks, which is why we select to fine-tune it on our extractive summarisation task.

2.8 Text Summarisation

Text summarisation is the task of transforming a piece of text into a shorter version that retains the most important information. There are two overarching categories: extractive and abstractive text summarisation. The former formulates the problem as a subset selection problem by returning only the most salient text excerpts from the original document ([ZLC+20]), while the latter aims to generate content anew, similar to how humans would do.

We will outline some key models that inspired our work below:

- **Gokhan**: The authors employ an unsupervised summariser based on K-Means clustering of sentences encoded with SentenceBERT ([RG19]). However, their embeddings are pre-trained on general text, and they suggest that employing in-domain language models would result in a better performance.
- **AMUSE** ([LV21]): The authors design an ETS system comprised of the following steps 1. shortening of report with an existing Genetic Algorithm [LL13], 2. encoding sentences with BERT vectors, and 3. performing binary classification with LSTMs for salient sentence

extraction. They suggest that further work should incorporate 1. efficient preliminary sentence removal, and 2. additional neural modelling stages for the representation and detection of relevant input text parts.

- **Hybrid model with RL** ([ZSEHR21]): The authors train a joint extractive-abstractive summarisation model with reinforcement learning optimised for the ROUGE-2 F1 metric. Their networks are based on attentive LSTMs augmented with an additional copy mechanism ([VFJ15]) achieving the second highest F1 score in the FNS21 competition.

- **T5 Hybrid** ([Orz21]): The author used T5 ([RSR+20]) for a hybrid model fine-tuned to generate the beginning of an abstractive summary and find the closest match of the output in the report's full text. This is the best performing algorithm in the FNS21 competition but also the first to consider transformer models from an abstractive summarisation perspective in the FNP workshops so far.

In this work we will be solely exploring the extractive method, and more specifically - the *supervised neural-based* (i.e., RNN, Transformer) type and the *unsupervised graph-based* (i.e., TextRank, LexRank) type.

2.9 TextRank

TextRank ([MT04]) is a graph-based unsupervised algorithm for extractive summarisation with the following key components:

1. **Sentence similarity**: TextRank computes the similarity between two sentences based on either the *cosine similarity* [23] of their vectors (e.g., bag-of-words, Word2Vec, FastText, etc.) or the *Jaccard similarity* [24] of their sets of words,

2. **Graph Construction**: TextRank represents sentences as nodes in a graph, and the similarity between each two sentences is represented as an edge between them.

[23] Cosine similarity is defined as the dot product of two vectors divided by the product of their norms, i.e., $\cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$.

[24] Jaccard similarity is defined as the size of the intersection divided by the size of the union of two sets, i.e., $J(A,B) = \frac{|A\cup B|}{|A\cap B|}$. For example, if sentence A is "I love apples." and sentence B is "I love oranges.", then $J(A,B) = \frac{2}{4} = \frac{1}{2}$, indicating that A and B share 50% of their unique words.

3. **Sentence ranking**: The constructed graph is passed into the PageRank algorithm ([BP98]) that assigns each sentence a score based on the importance of its neighbours. The final summary is then assembled from the selection of the top k sentences with the highest scores.

Nevertheless, [SGWM20] report two considerable weaknesses of TextRank (hence the need for a better baseline - LexRank):

1. **Extraneous words** - TextRank does not penalise the extraneous words (i.e., words that do not add any essential information to the sentence) which can artificially increase the PageRank score (i.e., the importance) of a sentence.

2. **Frequent words** - There is no weighting applied regarding the frequency (or rarity) of words in the sentence, which can lead to a bias towards sentences with more frequent words.

As a countermeasure, *stop words* (e.g, "a", "the", "and", etc) can be removed and *tf-idf* can be integrated into the similarity metric for a more balanced scoring mechanism. LexRank ([ER04]) builds up and resolves the issues of this algorithm, and we will discuss it in the next section.

2.10 LexRank

LexRank ([ER04]) is another unsupervised extractive summarisation method consistently used as a baseline in the FNS21 and previous challenges. It retrieves the most salient document sentences by computing their importance based on *eigenvector centrality*. To do that the algorithm creates a graph where each sentence represents a node and each edge is a weight between two nodes ([SGWM20]). The sentences are encoded as bag-of-words vectors of size N - the vocabulary size, and the weight metric is a combination of tf-idf (Eq.7,8) and cosine similarity - Eq.9.

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{7}$$

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{8}$$

$$\text{tf\_idf\_cosine\_similarity}(s_1, s_2) = \frac{\sum_{t \in T} \text{tf-idf}(t, s_1, D) \cdot \text{tf-idf}(t, s_2, D)}{\sqrt{\sum_{t \in T} \text{tf-idf}(t, s_1, D)^2} \cdot \sqrt{\sum_{t \in T} \text{tf-idf}(t, s_2, D)^2}} \tag{9}$$

where t is a term, d is a document within a collection of documents/sentences D.

Also, s_1 and s_2 are two sentences and T represents the set of all terms in both of them while $tf(t,d)$ denotes the term frequency of t in d, and $idf(t,D)$ is the inverse document frequency of t in the collection D.

The authors further propose finding the most important sentences by 1. applying a threshold for the creation of edges with Eq.9, 2. building an adjacency matrix and normalizing it to produce *transition probabilities*, 3. computing in an iterative fashion the *eigenvector centrality* until convergence, and finally 4. ranking sentences based on their *lexical* PageRank ([BP98]) score similarly to TextRank ([MT04]).

2.11 Evaluation Metrics

Confusion Matrix

The confusion matrix is an essential tool to visualise and help assessing the performance of trained classifiers against the true labels y_i.

For the problem of binary classification, it is a square matrix (Table 2) that displays the following key elements:

- True Positives (TP): Correct predictions of the positive class.
- True Negatives (TN): Correct predictions of the negative class.
- False Positives (FP): Incorrect predictions of the positive class (Type I error).
- False Negatives (FN): Incorrect predictions of the negative class (Type II error).

where for the problem of extractive text summarisation, the positive and negative classes correspond to *summary* and *non-summary* sentences, respectively. These matrix elements can be further combined into informative classification metrics:

- Accuracy: Proportion of correctly classified instances out of the total instances. Formulated as $\frac{TP+TN}{TP+TN+FP+FN}$.
- Precision (or Positive Predictive Value): Proportion of true positive instances out of all instances predicted as positive. Formulated as $\frac{TP}{TP+FP}$.

- Recall (or Sensitivity): Proportion of true positive instances out of all actual positive instances. Formulated as $\frac{TP}{TP+FN}$.

- F1-score: Harmonic mean of precision and recall (i.e., the trade-off between the two). Formulated as $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$.

	Actual	
	Positive	**Negative**
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 2: Confusion Matrix

ROUGE

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) ([Lin04]) is a popular recall-based evaluation metric for summarisation tasks which measures the overlapping units (i.e., n-grams) of a predicted summary c and a reference summary c^*. Here we briefly describe some of the most common variants of ROUGE (especially in the context of the FNS21 Task):

1. **ROUGE-N** measures the overlap of n-grams (sequences of n words) between c and c^*. For example, ROUGE-1 and ROUGE-2[25] (the official FNS21 metric) correspond to the overlap measurement of unigrams and bigrams, respectively.

2. **ROUGE-L** uses the longest common subsequence (LCS) to measure the similarity between c and c^*. The LCS is the longest sequence of words that appears in both summaries *in the same order*, but *not necessarily consecutively*. Thus, ROUGE-L is less sensitive to word order compared to ROUGE-N and can capture longer-range dependencies.

While ROUGE scores are explainable and easily implementable, they have certain limitations like 1. inability to capture semantic similarity ([ABK22]), 2. insensitivity to summary coherence

[25] We briefly demonstrate how to compute ROUGE-2 for a sentence pair c=”Fox can run” and c^*=”Fox can walk”. Here, the bigrams of c are {(”Fox”, ”can”), (”can”, ”run”)} and the bigrams of c^* are {(”Fox”, ”can”), (”can”, ”walk”)}. The intersection of the two sets is {(”Fox”, ”can”)}, which is the only bigram that appears in both c and c^*. Therefore, ROUGE-2 recall = Overlapping bigrams / Total bigrams in reference summary = $\frac{1}{2} = 0.5$.

([CMSE13]), and 3. lack of correlation with human evaluation ([LL10]). Despite these limitations, ROUGE remains a widely-used evaluation measure, particularly in extractive summarization and we use it in this work as well.

$$ROUGE - N = \frac{\sum_{S \in R} \sum_{n-gram \in S} count_{match}(n-gram)}{\sum_{S \in R} \sum_{n-gram \in S} count(n-gram)} \quad (10)$$

Figure 9: ROUGE-N: N-gram Co-Occurrence Statistics

3 Methodology

3.1 Data

The data for the FNS21 task is a collection of narrative parts of annual reports, converted from PDF to plain text. As discussed in Section 1.4 due to the rich visual representations in the PDFs, the resulting text suffers from various problems like 1. *spacing inconsistencies* - mixing of tab-space word delimiters, over-segmentation (i.e., a split into incoherent chunks) and under-segmentation (i.e., merging of unrelated words), 2. *symbol encoding issues* - introduction of unreadable non-alphanumeric characters, and 3. *formatting issues* - words having different casing, hyphenation at the end of a line, etc 4. *conversion of tables to text* - financial figures spanning over multiple lines and being mixed with the text. (Figure 10).

1.
```
 Following my appointment as Chief
E x e c u t i v e in J u l y 2 0 1 0 ,g r e ate r
e m p h a s i s
h a s b e e n p l a c e d o n f u l fi l l in g t h e
s u p p l y o f
tonnage due under legacy contracts and
```

2.
```
However,   the   Directors   further   believe   that   additional
capital    could    be   deployed   to  beneficial    effect.
```

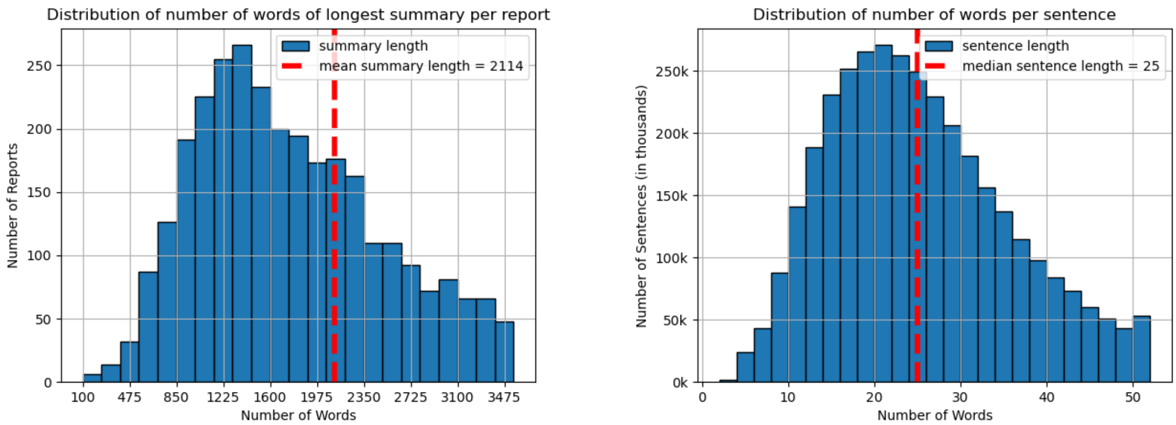
3.
```
Opening net book amount 116,635 35,624 166,754 319,013
Additions 51,380 7,647 307,546 366,573
```

4.
```
  This means that buyers can
_0_@uk_ar06_front.indd   5 20/04/2007   09:13:30 05
@UK PLC
Annual Report and Accounts 2006
use our network to purchase from their suppliers.
```

Figure 10: PDF-to-text conversion issues.

To address these issues, we have developed a rigorous data cleaning pipeline that achieves the following key objectives: 1. handles space-tab mixing via hand-crafted rules (derived from observation[26]), 2. retains alphanumeric characters, punctuation, spaces, financial symbols, and 3. removes sentences shorter than 3 words.

As discussed in Section 1.4, the annual reports are extremely long documents with an average length reported at 80 pages ([LV21]). Each one has at least two–three gold summaries provided by the FNS21 organisers, and we compute some statistics 1. helpful for grasping the nature of the output text, but also 2. useful for the evaluation of the summarisation models. On one hand, we can see that the average number of words in the longest summary is over $2,000$ (Figure 11a), while the FNS21 regulations specify an expected output of at most $1,000$ words. Furthermore, as we are not competing in the FNS21 task, for simplicity, during evaluation we will generate only summaries with at most 40 sentences. We arrive at this number by observing that the median number of words in the longest summaries is 25 (Figure 11b), and calculating that $\frac{1{,}000\text{words}}{25\text{words}} = 40$ sentences can suffice.



(a) Number of words in longest report summary

(b) Number of words in training sentences

Figure 11: Distribution of number of words in training sentences and report summaries

As we were only provided with the training and the validation FNS21 datasets (Table 1), we decided to treat the validation set as a testing set (Table 3)[27] and perform our own training-validation

[26]E.g., for some of the lines, characters were separated by spaces and words with tabs, hence the need for a custom rule.

[27]We observed that two of the annual report files were empty, hence the difference of 3,361 and 3,363 (Table 1 without testing set).

data split on a sentence level instead due to the significant variation in report lengths ([LV21]). Specifically, we use a 90–10 *stratified split* (i.e., the label distribution is retained in both sets) for

Data Type	Training + Validation	Testing	Total
Report full text	2,998	363	3,361

Table 3: Training-Validation-Testing Data Split

training and validation, respectively. We are aware that a validation and a training sentence can come from the same report, but we claim that this is not problematic for the following reasons:

1. Sentences are *de-contextualised* (i.e., without references or dependencies to others, taken out of context) and *shuffled*.

2. Sentences contain a *great deal of textual noise* due to the PDF-to-text conversion.

3. Annual reports are *numerous* but also *extremely long* (i.e., containing a lot of sentences).

Therefore, we believe that the training and validation sentences are to a large extent independent, and as for the process of sentence extraction, we refer you to Section 3.2 for an in-depth discussion.

3.2 Sentence Extraction

We approach the annual report summarisation problem from a supervised perspective - we cast the task of Extractive Text Summarisation (ETS) as a binary classification problem defined on the sentence level. More formally, we can describe the annual report as $d = \{s_1, s_2, \ldots, s_n\}$, where d is a document, represented in terms of sentences $s_i,\ 1 \leq i \leq n$ ([Liu19]).

Then, a candidate summary[28] can be $c = \{s_1, s_2, \ldots, s_k | s_i \in d\},\ 0 \leq k \leq n$. We further need to define the *gold summary*, c^* for a document d. In the case of the FNS21 task, there are at least two summaries per report, hence we will use the following notation for the set of all gold summaries for each document $C^* = \{c_1^*, c_2^*, \ldots, c_p^*\}$. Furthermore, the supervised learning labels are $y_i \in \{1, 0\}$ for each sentence s_i in d if the sentence is or is not in ***any*** of the gold summaries c_j^* for that document. We argue that in order to increase the positive samples (i.e., the summarizing sentences) we should not restrict ourselves to just one gold summary in the training process unlike [Orz21]. Our goal

[28] A candidate summary is generated from a model m_i but it is not yet a *best summary*.

is to achieve better latent feature extraction of summaries through the employment of all existing data, hence using ***any*** gold summaries. However, we are aware that this approach is more likely to encounter standard ETS issues, specifically - extracted summary sentences could be retrieved from unrelated paragraphs in the report. This can cause the "dangling anaphora" phenomenon, i.e. de-contextualised extracts are stitched together and can mislead the reader due to out-of-context references ([Lin09]).

While some authors ([ZSEHR21]) follow the greedy ROUGE-maximisation method of matching summary sentences to document sentences (established in [NZZ17]), we approach the problem in a more practical and faster fashion. After manual observation of the reports against their gold summaries, it became clear that almost for all sentences belonging to c_i^*, there was an exact match with a sentence in the whole annual report d.

This hypothesis was proven correct by one of the FNS21 contestants ([Orz21]) who reported that 99.4% of the summaries were included in the report as whole subsequences. Hence, after having pre-processed the text documents we iteratively match the sentences and generate the binary classification labels ($\{1,0\}$ representing *summary* and *non-summary*, respectively) for both the training and testing datasets.

We perform the sentence extraction as discussed above to produce a dataset (training and validation combined) of $3,554,800$ and $361,703$ sentences for classes 0 and 1, respectively. Additionally, the training and validation sets are split in a 90–10 stratified fashion.

3.3 Under-sampling and Data Augmentation

Due to the fact that the annual reports are extremely long while the summaries are very short, the sentence dataset is highly imbalanced with a ratio of around 1:10. Therefore, we took two[29] different approaches to balance the classes:

1. **Random under-sampling** – As described in [Wei13, WHB23], we *randomly remove 90% of the sentences* from the majority class (i.e., non-summary sentences) to produce a ratio of 1:1 summary to non-summary sentences (Figure 4).

2. **Data Augmentation** – We use the *back-translation* technique ([HKHC18]) to generate new sentences from the minority class (i.e., summary sentences), followed by a 80% random

[29] We also tried a third approach that is to augment sentences with the help of DINO ([SS21]) used for high-quality semantic augmentation, but we did not manage to recreate the desired output.

	initial data	**90% under-sampled training**	**validation**
label 0	3,199,319	319,931	355,481
label 1	325,533	325,533	36,170

Table 4: 90% Random Under-sampling

under-sampling of the majority class (Figure 5). For that purpose, we translate all training summary sentences from English to French and back to English with the help of the MarianMT model ([JDGD+18])[30]. The resulting dataset is then directly injected during the training process.

	initial data	**80% under-sampled training**	**augmented training**	**validation**
label 0	3,199,319	639,863	639,863	355,481
label 1	325,533	325,533	651,066	36,170

Table 5: Data Augmentation + 80% Random Under-sampling

.

Refer to Section 4.2 for the experiments with the different data balancing techniques.

4 Recurrent Extractor Model

As our main recurrent model we propose a GRU-based architecture ([CvMG+14]), inspired by [ZSEHR21] and depicted in Figure 12.

[30] https://huggingface.co/Helsinki-NLP/opus-mt-fr-en

The model consists of a word embedding layer, a fully-connected feed-forward neural network (FCFFNN), two bidirectional gated recurrent units (GRU) layers, a dot-product attention layer, and a linear projection layer with softmax activation.

The word embedding layer is used to convert the pre-processed input sentence into a vector representation. One of our implementation innovations is that we use FinText's FastText word embeddings [RZP21] because they 1. are character-based and thus can handle noisy or out-of-vocabulary words, and 2. are pre-trained on large corpora of financial news, achieving considerable in-domain performance improvements over general-purpose embeddings.

We use the FCFFNN layer to *map the vectorized sentences to a higher-level representation* (similar to [SDEB20]) capturing more complex features or patterns from the input text, but also to *reduce the dimensionality* of the input.

Two stacked GRU layers are used to *extract the latent recurrent features* from the compressed vector representation in both directions - forward and backward (refer to Section 2.3 for details).

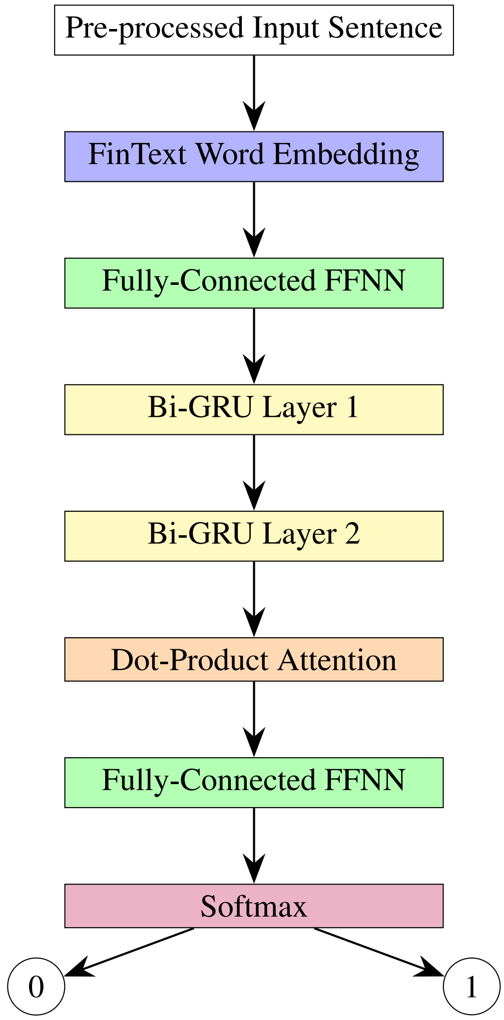


Figure 12: GRU-based extractive model

We further implement the scaled dot-product attention (Eq.4 from [VSP+17]) to compute a new weighted context-aware representation from the extracted features by the GRU layers.

The final layer is a fully-connected feed-forward neural network (FCFFNN) with a softmax activation function, which is used to *map the latent features to a binary classification* of the input sentence.

4.1 Training

We trained both models on Tesla V100-SXM2-16GB[31] and provide the following specifications:

1. **RNN architectures** - below we provide the common general details, however for an in-depth discussion on the hyperparameter tuning, see Section 4.2.

    - **Loss function:** Binary cross-entropy loss
    - **Optimizer:** Adam [KB17]
    - **Batch size:** 32
    - **Epochs:** set to 60, but due to early stopping, the model practically trains for less than 10 epochs
    - **Early stopping:** patience (i.e., the number of epochs to wait for improvement based on validation loss) set to 1

2. **FinBERT** - We followed the prescribed fine-tuning specifications provided in the original BERT paper [DCLT19]:

    - **Loss function:** Binary cross-entropy loss
    - **Optimizer:** Adam [KB17]
    - **Batch size:** 32
    - **Epochs:** 3
    - **Learning rate:** 2e-5
    - **Weight decay:** 0.01

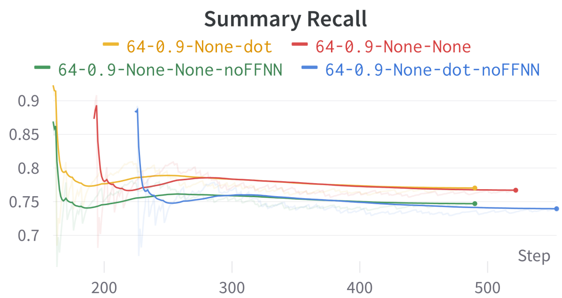
4.2 Hyperparameter Tuning

We have experimented with a number of hyperparameters for our recurrent model, including the use of a FCFFNN, the recurrence type, the hidden units, the effect of applying attention, the dropout rate, the learning rate, and the effect of data augmentation.

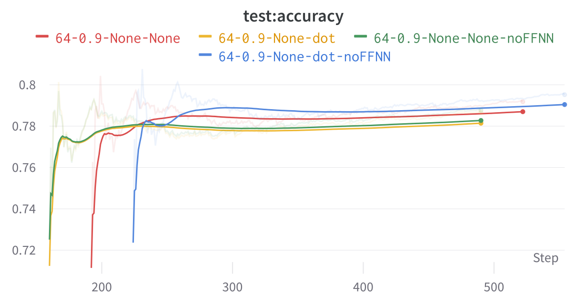
[31] We extend our gratitude to the University of Manchester's Computational Shared Facility (CSF) for kindly agreeing to provide us with the computational resources for this research.

For the analysis we will be extensively using the test accuracy, $F1$-score, and the *summary recall* metric. The latter is defined as the ratio of the number of correctly predicted summary sentences to the total number of summary sentences in the test dataset. We consider this metric to be extremely relevant because in the context of extractive summarisation, our goal is to minimise the Type II error (i.e., the number of sentences that should be in the summary but are not). Our reasoning is that our classifier must be as good as possible in recognising salient sentences (i.e., summarising sentences) even if it introduces some false positives (i.e., non-summarising sentences). In practice, the user can always remove irrelevant sentences, but it is much harder to add sentences that should have already been in the summary.

Each sentence in the report is represented as a (100, 300)-sized word embeddings vector, where 100 is the longest possible sentence length (i.e., implying long sentences are trimmed) and 300 is the dimensionality of the word embeddings. We test the effect of inserting an FCFFNN layer between the word embeddings and the GRU layers (each with 64 hidden units) and arrive at the following results: adding an FCFFNN layer increases Summary Recall by 2.5% (Fig. 13a), but marginally reduces Test Accuracy by less than 1% (Fig. 13b). We attribute the increase in Summary Recall to the fact that the FCFFNN layer is able to extract an additional mix of features from the word embeddings, which are then used by the GRU layers to make better predictions. As for the Test Accuracy, we believe that the small decrease is insignificant and we therefore choose to use the FCFFNN layer in our final model.



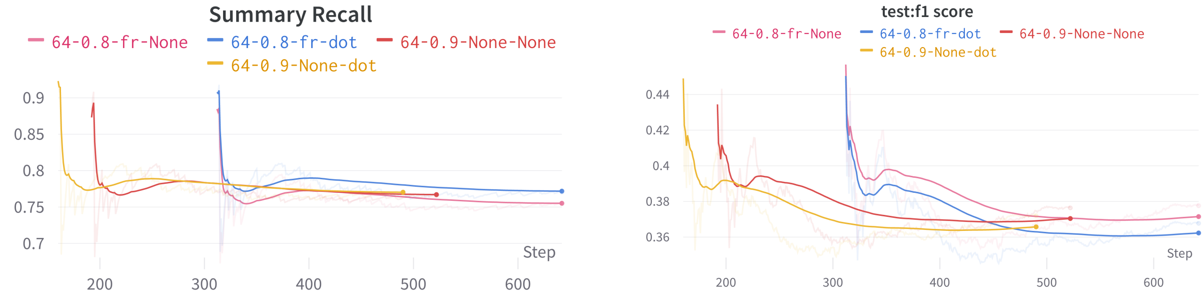
(a) Effect of FCFFNN layer on summary recall



(b) Effect of FCFFNN layer on test accuracy

Figure 13: Effect of FCFFNN layer on summary recall and test accuracy

We also explore the effect of using back-translated data (Section 3.3) on the model performance. Results shown in Fig. 14 suggest that the data augmentation does not improve the $F1$-score or the summary recall with a statistically significant margin. At the same time, it seems to be amplifying the effect of the scaled dot-product attention (Section 2.4, Figure 14a). While we



(a) Effect of data augmentation on summary recall

(b) Effect of data augmentation on $F1$-score

Figure 14: Effect of back-translation data augmentation on summary recall and $F1$-score

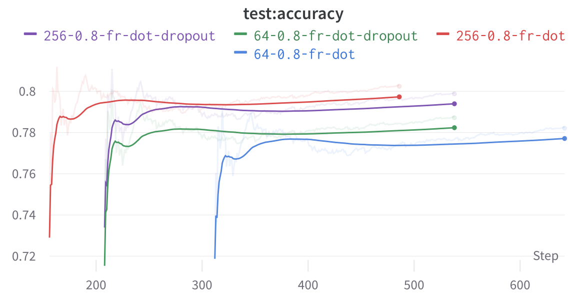
are disappointed by these results, we believe that there can be a number of reasons for this. For a machine translated sentence s_i^m and its original sentence s_i, we *hypothesise* that: 1. s_i^m contain a similar amount of noise as s_i (due to the pdf-to-text conversion process), 2. s_i^m does not introduce enough variation to s_i (i.e., s_i^m and s_i are too similar), and 3. while financial language itself is very domain-specific, it is not very semantically diverse (i.e., metaphors, idioms, etc. are limited in use).

In our proposed architecture we choose a bidirectional GRU (Bi-GRU) instead of a Bi-LSTM because 1. GRUs have a simpler structure than LSTMs and are easier to train (Section 2.3), and 2. our experiments show that GRUs outperform LSTMs with 1% in terms of Summary Recall. Furthermore, in terms of the hidden units, we select 64 over 256 because: 1. the architecture has $143,938$ and $2,050,306$ parameters, respectively (i.e., 256 would result in an over-parameterised model), 2. almost 4% increase in Summary Recall (Fig. 15a), although a 2% decrease in Test Accuracy (Fig. 15b), if we use 64 hidden units.

From the above experiments it is hard to make any certain conclusions on the effect of dropout[32] or the use of an attention mechanism (Section 2.4). We believe that only one *single-head attention* is not sufficient to learn all the complex relationships between words in the sentences. Our reasoning is that because a particular head specializes to only specific language aspects (i.e., syntactic, semantic, etc) ([CKLM19]), for future experiments it would be much more reasonable to

[32] It would make more sense for its application in the over-parameterised model, though this does not seem to be the case (Fig. 15).

(a) Summary Recall: 256 vs 64 hidden units + dropout



(b) Test Accuracy: 256 vs 64 hidden units + dropout

Figure 15: Effect of hidden units and dropout on summary recall and test accuracy

use multiple heads instead. Nevertheless, there still are some practical benefits of attention to our extractive summarisation system which we will discuss in Section 5.

5 Evaluation

In general, to assess the quality of a candidate summary c, we measure its similarity with the gold summary c^* based on their n-gram overlap $R = (c, c^*)$, where R is the ROUGE-F_1[33] metric([Lin04]). For the FNS21 task due to the extractive nature of our approach we will evaluate our models based on the ROUGE–maximising c_i^* gold summary, i.e.,

$$r = \underset{c^* \in C^*}{\text{argmax}} R(c, c_i^*) \tag{11}$$

Figure 16: Candidate summary evaluation as a gold summary ROUGE-maximisation

The intuition is that by extracting multiple sentences from the report, our generated candidate summary can retain sentences from *any* of the gold summaries. Hence, there must be at least one such gold summary where the overlap is maximal. The practical implications are that two models, m_1 and m_2 can produce two different candidate summaries c_1 and c_2, respectively. Their individual evaluation is based on gold summaries c_1^* and c_2^* (which can be the same when the candidates c_1 and c_2 are identical).

Following this evaluation mechanism, we compare our models in terms of their ROUGE metrics in Table 6. Please keep in mind that we include official results from the FNS21 competition for comparison, namely the T5-LONG-EXTRACT model ([Orz21]) and the MUSE model ([LL13]), which have been 1. trained on more data than our models due to us using the official validation as a test set (see Section 3.1), 2. evaluated on the official test set which we did not have access to.

We can provide the following commentary on the results:

- The best performing model is FinBERT-base ([YUH20]), which is a pre-trained on financial communication documents, achieving a ROUGE-2 score of 0.382, which is *at least as good as the official best-performing model* in the FNS21 competition: the T5-LONG-EXTRACT ([Orz21], Section 2.8). Surprisingly, data augmentation does not improve the performance

33

- We use a slightly different but faster version of ROUGE compared to the official metric [Lin04]. It can be accessed at: `https://github.com/pltrdy/rouge`
- The official FNS21 evaluation metric is the $F1$-score of ROUGE-2 and we will use it for the final evaluation.

Model	ROUGE-1	ROUGE-2	ROUGE-L
⋆ FinBERT-base	0.544	0.382	0.524
T5-LONG-EXTRACT ([Orz21])*	0.54	0.38	0.52
FinBERT-base + back-translation	0.490	0.321	0.468
MUSE ([LL13])*	0.5	0.28	0.45
⋆ GRU-base + attention + back-translation	0.276	0.106	0.249
GRU-base + back-translation	0.266	0.100	0.247
LexRank	0.250	0.086	0.227
TaxtRank	0.220	0.064	0.196
GRU-base	0.220	0.063	0.201
GRU-base + attention	0.221	0.062	0.204

Table 6: ROUGE: Model performance on the FNS21 test dataset.

of FinBERT-base, and we believe this was caused by the back-translation hypothesis we made in Section 4.2. Nevertheless, both models outperform the official top-line ([LL13]) by a significant margin for ROUGE-2.

- The GRU-base + attention + back-translation model is the best performing model out of all recurrent neural architectures.

- The LexRank and TextRank models are the best performing models that do not use a neural architecture. This shows that the neural architecture is beneficial for the task.

Bibliography

[ABK22] Mousumi Akter, Naman Bansal, and Shubhra Kanti Karmaker. Revisiting automatic evaluation of extractive summarization task: Can we do better than ROUGE? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1547–1560, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[Bay15] Justin Simon Bayer. Learning sequence representations, 2015.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117, 1998.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[CCG21] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. A comparison of LSTM and GRU networks for learning symbolic sequences. *CoRR*, abs/2107.02248, 2021.

[CHW19] Eddy Cardinaels, Stephan Hollander, and Brian White. Automatic summarization of earnings releases: attributes and effects on investors' judgments. *Review of Accounting Studies*, 24, 09 2019.

[CKLM19] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics.

[CKW18] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction. *CoRR*, abs/1801.02143, 2018.

[CMSE13] Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. Towards coherent multi-document summarization. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1173, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[CvMG+14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[EDH19] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy, July 2019. Association for Computational Linguistics.

[EHAR+19] Mahmoud El-Haj, Paulo Alves, Paul Rayson, Martin Walker, and Steven Young. Retrieving, classifying and analysing narrative commentary in unstructured (glossy) annual reports published as pdf files. *Accounting and Business Research*, 2019. Forthcoming.

[EHO22] Mahmoud El-Haj and Andrew Ogden. Financial narrative summarisation using a hybrid TF-IDF and clustering summariser: AO-lancs system at FNS 2022. In *Proceedings of the 4th Financial Narrative Processing Workshop @LREC2022*, pages 79–82, Marseille, France, June 2022. European Language Resources Association.

[EHRW+19] Mahmoud El-Haj, Paul Rayson, Martin Walker, Steven Young, and Vasiliki Simaki. In search of meaning: Lessons, resources and next steps for computational analysis of financial discourse. *Journal of Business Finance & Accounting*, 46(3-4):265–306, 2019.

[EHRY+19] Mahmoud El Haj, Paul Edward Rayson, Steven Eric Young, Paulo Alves, and Carlos Herrero Zorita. *Multilingual Financial Narrative Processing: Analysing Annual Reports in English, Spanish and Portuguese*. World Scientific Publishing, 2019.

[EHRZ21] Mahmoud El-Haj, Paul Rayson, and Nadhem Zmandar, editors. *Proceedings of the 3rd Financial Narrative Processing Workshop*, Lancaster, United Kingdom, 15-16 September 2021. Association for Computational Linguistics.

[Ell98] Robert K Elliott. Accounting in the 21st century. 1998.

[ER04] Gunes Erkan and Dragomir R. Radev. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004.

[GLT20] Andrea Galassi, Marco Lippi, and Paolo Torroni. Attention in natural language processing. *IEEE transactions on neural networks and learning systems*, 32(10):4291–4308, 2020.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[HKHC18] Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[JDGD$^+$18] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in c++, 2018.

[JM00] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000.

[Jon72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[KB16] P Kriz and H Blomme. The future of corporate reporting–creating the dynamics for change. *International Federation of Accountants (IFAC), available at: www. ifac. org/global-knowledgegateway/viewpoints/future-corporate-reporting-creating-dynamics-change (accessed 29 May 2016)*, 2016.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KVL21] Sophie Krimberg, Natalia Vanetik, and Marina Litvak. Summarization of financial documents with tf-idf weighting of multi-word terms. In *FNP*, 2021.

[L$^+$10] Feng Li et al. Textual analysis of corporate disclosures: A survey of the literature. *Journal of accounting literature*, 29(1):143–165, 2010.

[Lin04] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[Lin09] Jimmy Lin. Summarization. In *Encyclopedia of Database Systems*, pages 2906–2910. Springer, Heidelberg, Germany, 2009.

[Liu19] Yang Liu. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

[LL10] Feifan Liu and Yang Liu. Exploring correlation between rouge and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(1):187–196, 2010.

[LL13] Marina Litvak and Mark Last. Multilingual single-document summarization with MUSE. In *Proceedings of the MultiLing 2013 Workshop on Multilingual Multi-document Summarization*, pages 77–81, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[LRU20] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 3 edition, 2020.

[Luh57] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1(4):309–317, oct 1957.

[LV21] Marina Litvak and Natalia Vanetik. Summarization of financial reports with AMUSE. In *Proceedings of the 3rd Financial Narrative Processing Workshop*, pages 31–36, Lancaster, United Kingdom, 15-16 September 2021. Association for Computational Linguistics.

[LY19] Craig Lewis and Steven Young. Fad or future? automated analysis of financial text and its implications for corporate reporting. *Accounting and Business Research*, 49(5):587–615, 2019.

[MCCD13] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MT04] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[MYZ13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[NZZ17] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[Ola15] Christopher Olah. Understanding lstm networks, 2015.

[Orz21] Mikhail Orzhenovskii. T5-LONG-EXTRACT at FNS-2021 shared task. In *Proceedings of the 3rd Financial Narrative Processing Workshop*, pages 67–69, Lancaster, United Kingdom, 15-16 September 2021. Association for Computational Linguistics.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. 21(1), 2020.

[RZP21] Eghbal Rahimikia, Stefan Zohren, and Ser-Huang Poon. Realised volatility forecasting: Machine learning via financial word embedding. *SSRN Electronic Journal*, preprint:1–20, July 2021.

[SDEB20] Tanik Saikh, Arkadipta De, Asif Ekbal, and Pushpak Bhattacharyya. A deep learning approach for automatic detection of fake news, 2020.

[SGWM20] Steven Shearing, Abigail S. Gertner, Ben Wellner, and Liz Merkhofer. Automated text summarization: A review and recommendations. 2020.

[SP97] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SS21] Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models, 2021.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[VFJ15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[Wei13] Gary M. Weiss. *Foundations of Imbalanced Learning*, chapter 2, pages 13–41. John Wiley and Sons, Ltd, 2013.

[WHB23] Tarid Wongvorachan, Surina He, and Okan Bulut. A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information*, 14(1), 2023.

[XCWS13] Zhixiang Eddie Xu, Minmin Chen, Kilian Q. Weinberger, and Fei Sha. An alternative text representation to tf-idf and bag-of-words. *ArXiv*, abs/1301.6770, 2013.

[YUH20] Yi Yang, Mark Christopher Siy UY, and Allen Huang. Finbert: A pretrained language model for financial communications, 2020.

[ZLC+20] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online, July 2020. Association for Computational Linguistics.

[ZSEHR21] Nadhem Zmandar, Abhishek Singh, Mahmoud El-Haj, and Paul Rayson. Joint abstractive and extractive method for long financial document summarization. In *Proceedings of the 3rd Financial Narrative Processing Workshop*, pages 99–105, Lancaster, United Kingdom, 15-16 September 2021. Association for Computational Linguistics.