Imperial College London

BEng Individual Project

Imperial College London

Department of Computing

Finding the Needle in a Haystack: Zero-shot Rationale Extraction for Long Text Classifiers

Author:
Kamil Bujel

Supervisor:
Dr Marek Rei

Co-Supervisor:
Dr Helen Yannakoudakis

Second Marker:
Dr Josiah Wang

June 21, 2022

Submitted in partial fulfillment of the requirements
for the BEng in Mathematics and Computer Science of Imperial College London

Abstract

We investigate various soft attention architectures to extract plausible token-level rationale from long document Transformers, such as Longformer. We find that a direct application of Weighted Soft Attention, a method used to extract rationale from sentence classifiers, does not select meaningful rationale from long text classifiers. We suspect it is due to the insufficient token-level supervision signal. We propose Mean Soft Attention and Top-k Rest-0 Soft Attention as modifications to the original system that significantly improve the quality of the extracted rationale.

We report slow runtimes of the soft attention architectures for long documents. We propose a novel Compositional Soft Attention system that uses a soft attention layer to compose contextual token embeddings obtained for individual sentences. When combined with RoBERTa, we find the Compositional system to be $30 - 65\%$ faster than long document Transformers. We learn that the Compositional Soft Attention ranks individual tokens substantially better than other soft attention systems, but note that it underperforms on the task of sequence labelling and document classification.

Acknowledgements

I would like to thank my supervisors, Dr Marek Rei and Dr Helen Yannakoudakis, for their guidance and help throughout the project. They assisted me with the initial research proposal and agreed to supervise despite their numerous commitments. Special thanks go to Marek for also being my personal tutor throughout my degree at Imperial College London.

I would also like to thank Dr Andrew Caines for being part of the project and contributing useful insights to our approaches and suggesting new datasets we could use.

Further, I would like to thank my second marker, Dr Josiah Wang for his feedback on the background section. I would also like to thank Joe Stacey for providing me with exclusive access to the GPU cluster during the final experimentation stage.

Lastly, I would like to thank my partner, Peini Yu, for being supportive and understanding of me working during the sunny weekends of May and June 2022.

Contents

Chapter 1

Introduction

Since the introduction of the Transformer architecture [1], the research in Natural Language Processing (NLP) has focussed on adapting large pretrained language models to perform well on downstream tasks. The variations of the original Transformer model, such as BERT [2] or RoBERTa [3], have been shown to achieve state-of-the-art performance on tasks such as sentence classification [4], question answering [3] or sequence labelling [5].

The multihead self-attention layers present in Transformers have been shown to attend to various linguistic properties, such as syntax and coreference [6], contributing to the Transformers success [7]. However, due to the quadratic growth of the attention with the sequence length, it has also been one of its main limitations. In order to decrease the computational complexity, sequences larger than 512 tokens are usually truncated [2]. This approach works well for a single sentence, or a concatenation of such, but does not represent larger documents well [8, 9]. That is in contrast with Recurrent Neural Networks (RNNs) [10], which could be applied to sequences of arbitrary length [11], although suffering from vanishing gradients.

In order to improve the long document representation, sequential and hierarchical Transformers-based approaches have been proposed [12, 13]. However, those methods have been outperformed by sparse attention approaches, such as Big Bird [8] and Longformer [9]. These architectures attempt to reduce the computation required by implementing sparse attention functions that combine local and global information. However, they are signficantly slower than standard Transformers.

While the explainability of short-text Transformers is a well-explored topic in the literature [14, 15], it is not the case for long document models. Shi et al. [16] and Feucht et al. [17] have both investigated explanations of Longformer, but have only provided a qualitative evaluation. Meister et al. [18] have shown that sparsity of attention does not lead to improved explainability, but have only investigated it in the context of the vanilla BERT model. To the best of our knowledge, there is no work that quantitavely investigates explainability of long text Transformers.

1.1 Objectives

In this project, we investigate the representation and explainability of long document classification. In particular, we focus on designing *rationale extractors* [19] for long text Transformers that can be evaluated against token-level human annotations by measuring plausability (agreeability to human annotators; [20]). The rationales are extracted from token-level scores assigned by the system as part of the zero-shot sequence labelling task.

We first adapt the soft attention architecture proposed by Rei and Søgaard [21] and modified for Transformers by Bujel et al. [15] to work well for long document classification. We show that a direct application of the Weighted Soft Attention architecture does not perform well. We propose non-trivial modifications, namely Mean Soft Attention and Top-k Rest-0 Soft Attention, to allow the supervision signal to reach more tokens. The modifications lead to significant improvement of the token-level classification performance, with $1.99\% - 9.42\%$ absolute F_1 improvement for Mean Soft Attention.

We further propose Compositional Soft Attention, a novel architecture that obtains contextual token embeddings from a standard Transformer applied sentence-wise. These token representations are then composed across sentences to obtain a document-level representation. We show that this approach is $30\% - 65\%$ faster than a standard application of soft attention with long text Transformers. We find the system to determine the correct ranking of tokens, but learn a suboptimal distribution of token scores, as indicated by the substantial improvement in the token-level MAP, but lower F_1 score. We propose modifications to the loss function to alleviate this issue in future work. We also note the significantly lower document-level performance of the model, which we suspect is due to a poorly defined early stopping criterion.

We publicly release our code and all experiments configuration files[1].

1.2 Contributions

We believe the following are the unique contributions of this project:

- We quantitatively evaluate zero-shot rationale extractors for long text Transformer-based classifiers. To the best of our knowledge, this is the first time these systems are evaluated in such a setting.
- We modify the Weighted Soft Attention to work well for long text classification and introduce Mean and Top-k Rest-0 Soft Attention architectures that perform significantly better on the token-level rationale extraction.
- We propose novel Compositional Soft Attention architecture that composes contextual token embeddings obtained sentence-wise to build document representation and provide token-level rationale. We find the system to be $30 - 65\%$ faster than Longformer-based soft attention systems.
- We adapt existing Grammatical Error Detection datasets to binary document classification and token-level rationale plausibility extraction.

[1] https://github.com/bujol12/document-classification-transformers

1.3 Challenges

This project involved diverse experimentation to arrive at the final architectures and results. Additionally, we faced the following challenges with regards to the chosen goals:

- **Limited number of suitable datasets**: Despite a large number of publicly available datasets for document classification, there is a shortage of document classification datasets that contain token-level human annotations. That is due to the prohibitive costs of such labelling process. Therefore, we were forced to work with datasets that did not contain enough samples or not enough long documents. We also had to split some of the datasets into our own binary classes. We designed preprocessing scripts to convert them into a common format, as each dataset followed a different convention for representing the token-level annotations.
- **Limited GPU access to large GPUs**: While we benefited from a largely undisturbed access to GPUs with memory ranging from 10GB to 16GB, we could not gain access to a larger family of GPUs. This had an impact on the training times of the long document models, which had to be be optimised with a batch size of 1. Further, we did not manage to test BigBird as we did not manage to fit it onto any GPU available to us.
- **Batching for Compositional Soft Attention**: In order to improve the computational efficiency of the Compositional Soft Attention model, we had to perform a 2-level batching. This included batching documents together as input for the soft attention component, while also batching individual sentences as input to obtain contextual token embeddings. This approach presented numerous challenges with masking and padding, which we had to resolve.

1.4 Potential Publication

We intend to submit the presented work to the *BlackboxNLP 2022* workshop co-located with The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022), with the focus on Compositional Soft Attention architecture.

This will be done following further research on the proposed architecture. We would like to refine the system to learn better distributions of token scores. We hope this will improve the token-level F_1 performance and allow the model to set new state-of-the-art across all datasets. Currently, the model learns correct rankings of tokens, but fails to optimise some of the token scores to be below the classification threshold, leading to substantial improvement on the token-level MAP scores, but lower token-level F_1 metric.

Chapter 2

Background

In this chapter, we overview the research in Natural Language Processing. We start by introducing Recurrent Neural Networks (RNNs) and the attention architectures that were first proposed for RNNs. In particular, we explore the soft attention architectures proposed by Rei and Søgaard [21] for zero-shot sequence labelling.

We overview Transformers - a family of state-of-the-art models that are widely used in Natural Language Processing and form the backbone of most systems. Specifically, we delve into the efforts in designing efficient long text Transformers, which we use throughout the project. Additionally, we introduce the soft attention architecture that was adapted for Transformers by Bujel et al. [15].

Lastly, we review the recent efforts in explainability of NLP models. We define the key explainability concepts such as *plausibility* and *faithfulness*. We then review recent work in extracting plausible rationale and how zero-shot sequence labellers can be used to extract these rationales.

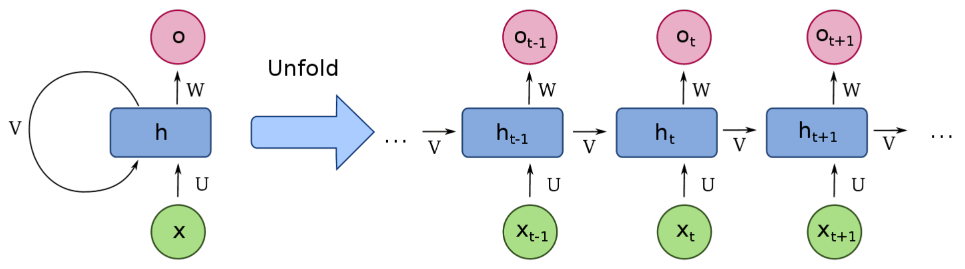


Figure 2.1: The representation of a Recurrent Neural Network cell h in its folded state on the left-hand side and the unfolded state on the right-hand side. U and V on the diagram are equivalent to W_h and U_h in Eq. 2.1 respectively. W and o_t represent an output layer obtained from the hidden output vector h_t. Diagram borrowed from (https://en.wikipedia.org/wiki/Recurrent_neural_network)

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) were first introduced by Rumelhart et al. [10] in the context of learning internal representations and error propagation. Unlike feedforward networks [22], they included recurrent connections between neurons, permitting them to model temporal and sequential features.

Assuming input matrix $X = \begin{pmatrix} x_0 & x_1 & \dots & x_n \end{pmatrix}^T \in \mathbb{R}^{n \times m}$ consists of n input row vectors x_i, a Recurrent Neural Network cell for input row vector $x_t \in \mathbb{R}^m$ was defined as:

$$h_t = \phi_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.1}$$

where ϕ_h is the activation function, $W_h \in \mathbb{R}^{h \times m}$ and $U_h \in \mathbb{R}^{h \times h}$ are weight matrices, $b_h \in \mathbb{R}^h$ is the vector of biases and $h_t \in \mathbb{R}^h$ is the (hidden) output vector, with $h_0 = 0$. h is the size of the recurrent layer. Figure 2.1 presents the folded and unfolded RNN hidden cell h_t.

2.1.1 Long-Short Term Networks

The work on RNNs led to the design of Long-Short Term Networks (LSTMs) [24], a variation of RNNs aiming to solve the issues with vanishing gradients in the previous architectures [25] by introducing Constant Error Carousel units. This was followed by Gers et al. [26] suggestion of an extra component of LSTMs - a forget gate. These modifications have been widely attributed to the LSTMs success [23].

For the previously defined input matrix x, we use the no-peephole LSTM definition by Greff et al. [23]:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.2}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.3}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.4}$$

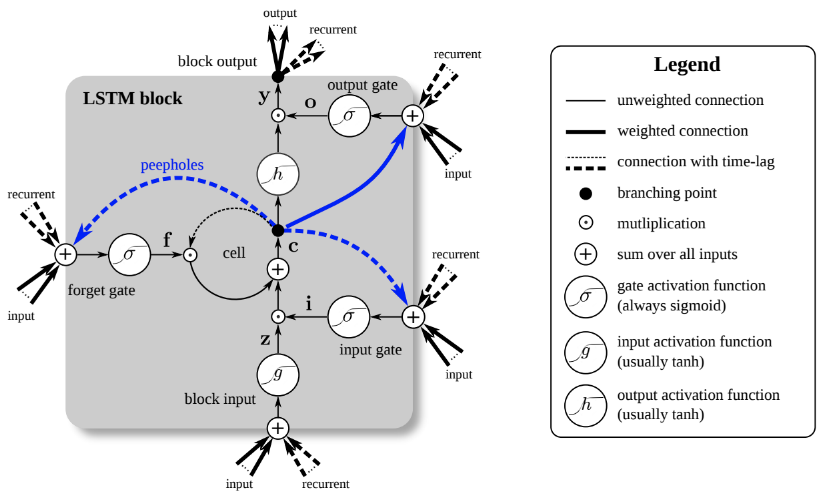


Figure 2.2: A modern day LSTM cell, consisting of the input, output and forget gates, as well as the peephole connections. As presented by Greff et al. [23]. y on the diagram represents h_t in Eq. 2.7.

where $i_t \in \mathbb{R}^h$ is the input gate, $f_t \in \mathbb{R}^h$ is the forget gate and $o_t \in \mathbb{R}^h$ is the output gate. W_i, W_f, $W_o \in \mathbb{R}^{h \times m}$ and W_i, W_f, $W_o \in \mathbb{R}^{h \times h}$ are weight matrices, $b_i, b_f, b_o \in \mathbb{R}^h$ are bias vectors and σ is the sigmoid activation function.

Additionally, we define the cell input block $z_t \in \mathbb{R}^h$:

$$z_t = \tanh(W_z x_t + U_z h_{t-1} + b_z) \tag{2.5}$$

with W_z, U_z, b_z defined similar as above.

These gates and cell input blocks are then combined to obtain the hidden (output) state $h_t \in \mathbb{R}^h$ and the memory cell $c_t \in \mathbb{R}^h$:

$$c_t = f_t \circ c_{t-1} + i_t \circ z_t \tag{2.6}$$
$$h_t = o_t \circ \tanh(c_t) \tag{2.7}$$

The hidden states h_t can then be used for further inference on each of the input vectors. Figure 2.2 represents a modern day LSTM cell, with additional peephole connections, pictured by Greff et al. [23].

In NLP, LSTMs have seen success for various tasks, such as language modelling [27], machine translation [28], sequence tagging [29] or reading comprehension [30]. Usually a bidirectional LSTM (biLSTM) model [31] is used to allow the architecture to represent both forward and backward dependencies. Hermann et al. [30] also showed that LSTMs are effective at modelling long documents, with their dataset including sequences of up to 2000 tokens of news articles.

2.1.2 Attention

Following the adaptation of RNNs for various NLP tasks, Bahdanau et al. [32] proposed the use of dynamic attention over words to construct sentence representations as context vectors $r \in \mathbb{R}^h$:

$$r = \sum_{i=1}^{n} \alpha_i h_i \tag{2.8}$$

where $h_i \in \mathbb{R}^h$ is the hidden state of a biLSTM and $\alpha_i \in \mathbb{R}$ is an attention value:

$$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)} \qquad e_j = \phi(W_e h_j + b_e) \tag{2.9}$$

where $W_e \in \mathbb{R}^{1\times h}$ and $b_e \in \mathbb{R}$ are trainable parameters, ϕ is an activation function and $e_j \in \mathbb{R}$.

The addition of an attention component allowed the network to learn which words to focus on more in order to obtain better sentence representations. While the original application was machine translation, similar architectures have been successful for sentence summarisation [33], error correction [34] and sequence labeling [35].

2.1.3 Zero-Shot Sequence Labelling

Rei and Søgaard [21] proposed a soft attention architecture based on Bahdanau et al. [32] to perform the task of zero-shot sequence labelling. They supervised a system with sentence-level annotations, but measured the performance on the token-level. The system intended the token-level predictions to serve as rationale of the sentence-level predictions.

The architecture consisted of a biLSTM component that operated on word embeddings [36], with bidirectional hidden states concatenated to form $\widetilde{h_i} = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. This was then passed through a feed-forward layer to obtain the representation h_i of a particular word w_i:

$$h_i = \tanh(W_h \widetilde{h_i} + b_h) \tag{2.10}$$

On top of these word representations, an attention layer is fitted in order to obtain attention values e_i:

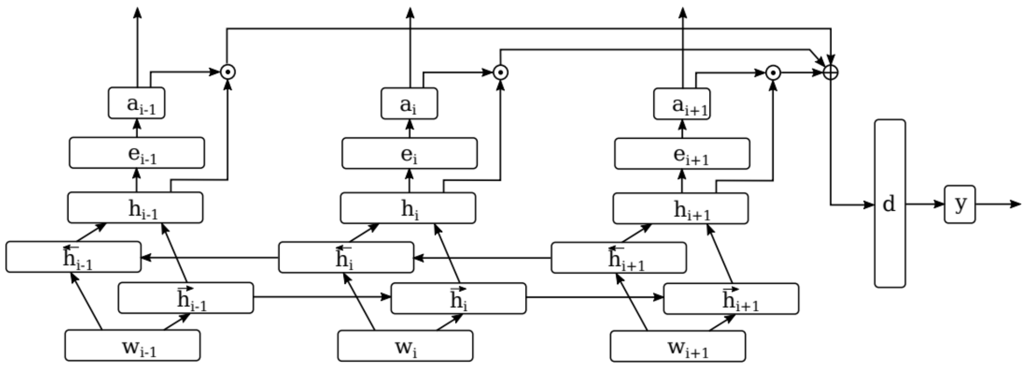


Figure 2.3: Soft Attention biLSTM architecture, as presented by Rei and Søgaard [21]. w_i represent input word embeddings, h_i the word representations, while e_i and a_i show the attention values and weights respectively. d is the overall sentence representation, while y is the predicted sentence label.

$$e_i = \tanh(W_e h_i + b_e) \tag{2.11}$$

In order to enable binary classification of tokens, a soft attention score $\widetilde{a_i}$ was introduced to ensure that more than one token can have attention score $\widetilde{a_i} >= 0.50$ to provide a natural classification threshold. These soft attention scores $\widetilde{a_i}$ are obtained using:

$$\widetilde{a_i} = \sigma(\widetilde{e_i}) \quad \widetilde{e_i} = W_{\tilde{e}} e_i + b_{\tilde{e}} \tag{2.12}$$

where $\widetilde{e_i}$ is a single scalar value.

To build the sentence representation c, the soft attention scores $\widetilde{a_i}$ need to be normalised to obtain soft attention weights a_i. We then combine those weights with word representations h_i:

$$a_i = \frac{\widetilde{a_i}}{\sum_{j=1}^{N} \widetilde{a_j}} \quad c = \sum_{i=1}^{N} a_i h_i \tag{2.13}$$

Finally, the sentence-level classification is performed:

$$d = \tanh(W_d c + b_d) \quad y = \sigma(W_y d + b_y) \tag{2.14}$$

where $0 \leq y \leq 1$ is a single value, with $y > 0.50$ representing positive class. Figure 2.3 represents this soft attention architecture, as presented by Rei and Søgaard [21].

The model was supervised with annotations on the sentence-level, with additional loss functions used to encourage the model to learn attention scores for tokens:

$$L_1 = \sum_j (y^{(j)} - \tilde{y}^{(j)})^2 \tag{2.15}$$

$$L_2 = \sum_j (min(\widetilde{a_i}) - 0)^2 \tag{2.16}$$

$$L_3 = \sum_j (max(\widetilde{a_i}) - \tilde{y}^{(j)})^2 \tag{2.17}$$

$$L = L_1 + \gamma(L_2 + L_3) \tag{2.18}$$

Eq. 2.15 indicates the sentence-level loss. Eq. 2.16 represents the property that only some, but not all tokens can have a positive label, while Eq. 2.17 uses the fact that for a sentence with a positive label, there needs to be at least one token with a positive label. Eq. 2.18 is the overall loss used for backpropagation.

The obtained attention scores can be used to infer token-level labels and be evaluated based on the token-level annotations. By design, the attention scores also provide a mechanism for interpreting the model and quantitatively evaluating the explanations using the token-level annotations.

This sentence-level method forms the basis of our proposed approach for long-document classification. We propose a soft attention architecture that utilises token-level representations to obtain document-level predictions.

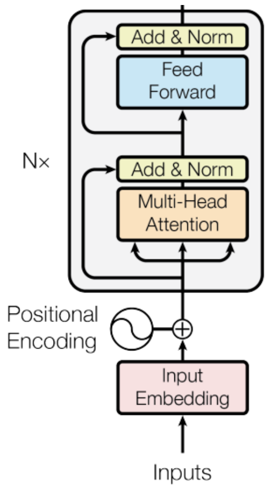


Figure 2.4: The Transformers encoder, as presented by Vaswani et al. [1]. It contains the Input Encoding component (Input Embedding & Positional Encoding) that generates the initial embeddings, followed by N the Transformer layers, each consisting of Multi-head Attention and a Feed-Forward layer, combined with Addition and Normalisation layers.

2.2 Transformers

Vaswani et al. [1] introduced Transformers, an architecture with a novel self-attention mechanism. The system does not include the recurrence relationship inherent to Recurrent Neural Networks. The initial experiments showed this new architecture to significantly outperform previous approaches on tasks of machine translation and constituency parsing. Following this work, Transformer-based models such as BERT [2] and RoBERTa [3] were introduced and shown to generalise well on numerous tasks and outperform previous encoder-decoder approaches in the GLUE behnchmark [37]. This section first describes a general Transformer encoder architecture, as proposed by Vaswani et al. [1], followed by the exploration of vast applications of Transformer-based models in modern NLP.

2.2.1 Architecture

The initial Transformer model consists of 6 identical layers, with each layer consisting of 2 sublayers - a multi-head attention and a fully connected feed-forward networks. Normalised residual connections [38, 39] are also present for each sublayer. The dimensions of each layer are d.

The initial input tokens are embedded into vectors of dimensions d, which are then summed with positional embeddings. The general overview of the architecture and a single layer is presented in Figure 2.4.

2.2.2 Multi-Head Attention

The key feature of the Transformer architecture is the multi-head self-attention mechanism, which permits each layer to learn to represent certain linguistic properties and model long-term forward and backward dependencies [6].

The self-attention layer can be described as acting on 3 different types of inputs: Queries (Q), Keys (K) and Values (V):

$$A(Q,K,V) = \sigma(\frac{QK^T}{\sqrt{d_k}})V \tag{2.19}$$

where d_k is the dimension of the matrix K. In the encoder Transformer architecture, $Q = K = V$, with the values being the output of the previous layer.

These self-attention layers are stacked together and passed through a feed-forward layer to obtain multi-head self-attention:

$$MHA(Q,K,V) = [A(Q_1,K_1,V_1),...,A(Q_h,K_h,V_h)]W^O \tag{2.20}$$

where $Q_i = QW_i^Q$, $K_i = KW_i^K$, $V_i = VW_i^V$, $W^O \in \mathbb{R}^{hd_k \times d}$ and h is the number of heads per layer.

2.2.3 Feed-Forward Network

The feed-forward network consists of 2 layers, with all transformations applied position-wise to input matrix X and a ReLU [40] activation in between:

$$N(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.21}$$

where $W_1 \in \mathbb{R}^{d \times d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff} \times d}$. $d_{ff} = 1024$ in the intial Transformer architecture.

2.2.4 Positional Encodings

In order to facilitate the ordering of the sequence of words without the recurrence relationship, posititional encoding was proposed, which is added to the initial token embeddings. For a token in position pos, the vector $p \in \mathbb{R}^d$ of positional encodings can be expressed as:

$$p_{2i}(pos) = \sin(pos/10000^{2i/d}) \tag{2.22}$$

$$p_{2i+1}(pos) = \cos(pos/10000^{2i/d}) \tag{2.23}$$

The reason for the function having such a format was because for any fixed offset k, $p(pos + k)$ is a linear function of $p(pos)$, therefore making it easier for the self-attention mechanism to attend to relative positions.

We note that modern Transformers such as BERT models use positional encodings p_i that are learnt during the pre-training process.

2.2.5 Pre-trained Language Models

The Transfomer architecture proposed by Vaswani et al. [1] led to a series of systems being released that had substantially improved on the original Transformer and applied it to a wider variety of tasks. Radford et al. [41] demonstrated how unsupervised generative pre-training of language models followed by fine-tuning on downstream tasks can lead to state-of-the-art performance on a 9 out of 12 NLP tasks explored. The proposed model, GPT, was optimised during the pre-training phase on the language modelling likelihood objective, where $u_1, u_2, ...$ is a sequence of tokens:

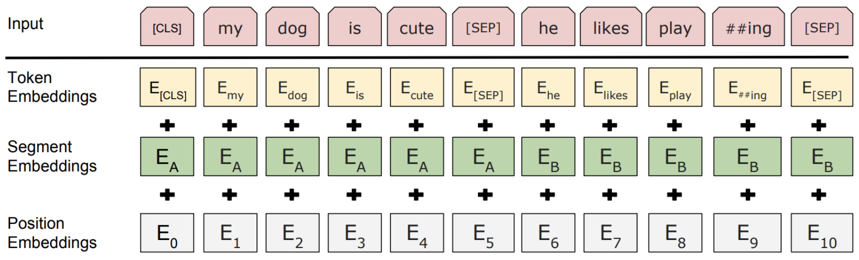


Figure 2.5: Representation of BERT tokenisation process, as presented by Devlin et al. [2]. At the top, the "input" row contains the original tokens, with special classification $[CLS]$ and the separator $[SEP]$ tokens added. These input tokens are converted to token embeddings E_{tok}, which are then summed with segment embeddings E_{segId} representing the segment the token belongs to and the positional embeddings E_i representing the position of the token in the sequence.

$$L_1 = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1} | \Theta) \tag{2.24}$$

where Θ is a Transformer model and k is the context window size.

The GPT model permitted pre-training based on the next token only. In order to exploit both forward and backward relationships between tokens during pre-training, Devlin et al. [2] proposed Bidirectional Encoder Representations from Transformers (BERT). This was achieved by introducing a novel Masked Language Modelling objective (MLM), where some of the input tokens are masked at random, with the model objective to predict those masked tokens. The pre-trained BERT model could then be extended with a single extra layer and subsequently fine-tuned for the downstream task to achieve state-of-the-art performance on 11 different NLP tasks.

Given the computational cost of pre-training BERT, the pre-trained version was released online and led to BERT being widely used to obtain *contextual word embeddings* of tokens for a variety of tasks and architectures [42].

Further improvements to Transfomers have been proposed. Liu et al. [3] found BERT to be severely undertrained and released Robustly Optimised BERT Pre-training Approach (RoBERTa) that achieved state-of-the-results. Yang et al. [43] found that the BERT masking procedure ignores dependencies between masked positions and proposed an autoregressive pre-training procedure (XLNet) to overcome it.

2.2.6 Tokenisation & Input/Output Representation

For BERT, WordPiece embeddings [44] with 30,000 vocabulary are used to tokenise the input words to token embeddings E_{tok}. The tokenisation is prepended with a special classification token $[CLS]$ used for inferring text-level representation. These embeddings are summed with additional segment embeddings E_{segId}, to indicate which sentence the token belongs to, and standard Transformers positional embeddings $E_i = p_i$ (§2.2.4). The full input embeddings representation is presented in Figure 2.5.

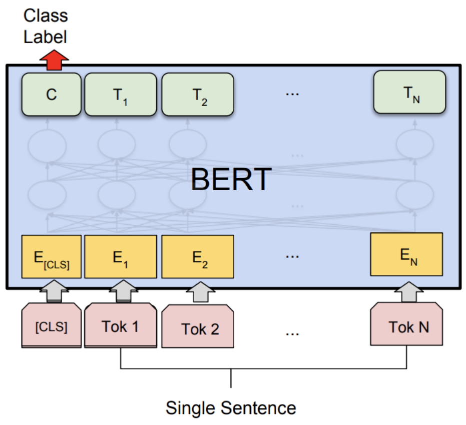


Figure 2.6: The BERT architecture for sentence classification, as presented by Devlin et al. [2]. The bottom input tokens Tok represent the input sentence tokenised by the BERT's tokeniser, as well as the special purpose classification token $[CLS]$. These are then converted to token embeddings $E = [E_{CLS}, E_1, ..., E_N]$ and fed into the Transformers part of BERT. $T = [T_1, ..., T_N]$ represents the contextual embeddings of input tokens, with C representing the contextual embedding of the $[CLS]$ token. C is then used to predict the text class label.

As output, BERT returns the final representation of the $[CLS]$ token, $C \in \mathbb{R}^d$ and token contextual embeddings $T = [T_1, ..., T_N]^T \in \mathbb{R}^{N \times d}$. These outputs can be used for predictions on the text or token level.

2.2.7 Text Classification

In particular, we are interested in fine-tuning the pre-trained language models for the task of text classification. Sentence classification is one of the downstream tasks explored by Devlin et al. [2] when evaluating BERT. To obtain a sentence representation for predictions, they utilise the special $[CLS]$ token and its corresponding output $C \in \mathbb{R}^d$ that is fed into a separate classification layer to obtain the sentence logits:

$$y = \phi(W_c C + b_c) \tag{2.25}$$

where $W_c \in \mathbb{R}^{c \times d}$ and b_c are parameters, ϕ is the activation function, $c \in \mathbb{N}$ is the number of classes and $y \in \mathbb{R}^c$ are the output logits. The architecture is presented in Figure 2.6.

Sun et al. [4] showed that a similar approach based on the $[CLS]$ token prediction works well for texts consisting of multiple sentences.

2.2.8 Zero-Shot Sequence Labelling on Transformers

Bujel et al. [15] explored how Transformers sentence classifiers can be used as zero-shot sequence labellers. They replaced the biLSTM token representations h_i (Eq. 2.11, 2.13) with contextual token embeddings T_i obtained from RoBERTa as input to the soft attention layer:

$$e_i = \tanh(W_e T_i + b_e) \qquad c = \sum_{i=1}^{N} a_i T_i \tag{2.26}$$

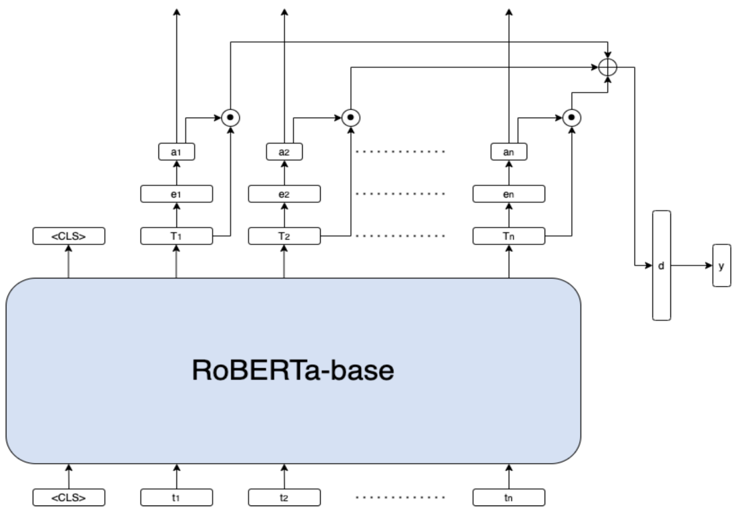


Figure 2.7: Weighted Soft Attention architecture for Transformers, as presented by Bujel et al. [15]. t_i represents the input tokens, T_i the contextual token embeddings, with e_i and a_i being the attention values and attention weights respectively. d is the sentence-level representation, with y being the predicted sentence label.

The full architecture is presented in Figure 2.7.

They found that the standard soft attention approach presented by Rei and Søgaard [21] for LSTMs needed adaptation of the soft attention function to work well for Transformers. The investigation showed that it was likely because the top layers of Transformers already learn well to attend to individual tokens, despite lack of supervision. Additionally, contextual embeddings for neighbouring tokens were found to be of high similarity, which made the original soft attention function unable to differentiate between tokens. Therefore, a Weighted Soft Attention function was proposed:

$$a_i = \frac{\widetilde{a_i}^{\beta}}{\sum_{j=1}^{N} \widetilde{a_j}^{\beta}} \tag{2.27}$$

where β is a hyperparameter, with values $\beta > 1$ allowing for a sharper distribution of attention scores $\widetilde{a_i}$. The authors report $\beta = 2$ to work best.

Following the application of Weighted Soft Attention mechanism on top of Transformer contextual embeddings, the architecture was found to achieve new state-of-the-art performance on zero-shot sequence labelling.

We note that the Weighted Soft Attention architecture has not been evaluated on documents or in combination with long text Transformers (§2.2.9). One of the aims of this work is to measure the document-level performance of Weighted Soft Attention in combination with both standard (e.g. RoBERTa) and long text Transformers.

2.2.9 Long Text Transformers

All of the initial Transformer-based models conventionally supported sequences up to 512 tokens, due to memory limitations on modern GPUs. The limiting fac-

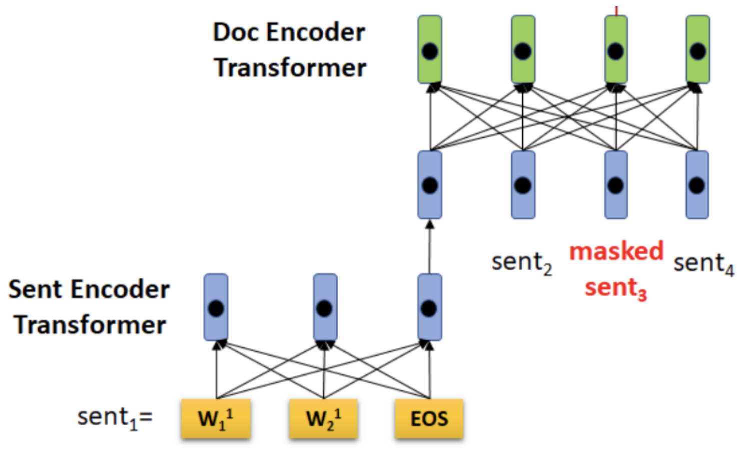


Figure 2.8: HIBERT architecture, as presented by Zhang et al. [13]. *Sent Encoder Transformer* represents a standard token-level Transformer. Its sentence representation from the *EOS* token is then used as input to a sentence-level Transformer that produces document representation (*Doc Encoder Transformer*). *masked sent*$_3$ is an example of a masked sentence in the Doc Encoder Transformer.

tor was the self-attention layer, which had quadratic computational complexity to the sequence length. Sequences longer than the maximum size were simply truncated [2], which led to long documents not being represented well [8, 9]. This problem of computational inefficiency and lack of scalability of standard Transformers was tackled by the long text Transformers, which we overview below.

Hierarchical BERT

Pappagari et al. [12] proposed a hierarchical approach, where BERT is applied to individual sentences, with the sentence-level representation being combined into a document-level representation with an extra 2-layer Transformer that takes sentence representations as input. This was found to outperform the approach where a biLSTM layer was used to combine representations of each sentence.

Zhang et al. [13] extended this approach to propose Hierarchical BERT (HIBERT) that additionally included *Document Masking*, a pre-training method that masks out some input sentences of a sentence-level Transformer, which are then predicted by the sentence-level Transformer. Figure 2.8 pictures the architecture proposed by Zhang et al. [13].

Transformer-XL

Dai et al. [45] introduced Transformer-XL, which added a recurrence relationship into the Transformer and proposed relative positional encodings. The architecture introduced fixed attention context windows was designed to be applied to segments of text and was shown to be able to model the long-term dependencies of up to 900 tokens.

Sparse Transformer

These hierarchical and fixed-context approaches were then outperformed by the attempts to make the self-attention component of Transformers sparse. Child et al. [46] introduced Sparse Transformer, that reduces the attention operation to

$O(n\sqrt{n})$ by introducing *factorised self-attention* that ensures that each input token is connected to every output token with a maximum of $p = 2$ attention steps. In particular, a *fixed factorised self-attention* is found to do well for NLP tasks. It uses a 2-head attention pattern, where the first one ($A^{(1)}$) attends to all tokens within the same stride of size $l = 128$ and the second attention head ($A^{(2)}$) attends to all previous strides. Formally, for a token in position i:

$$A_i^{(1)} = \{j : \lfloor j/l \rfloor = \lfloor i/l \rfloor\} \tag{2.28}$$

$$A_i^{(2)} = \{j : j \mod l \in \{t, t+1, ..., l\}\} \tag{2.29}$$

where $t = l - c$, and $c = 8$. The model was trained with a maximum context length of 12,288 tokens, the largest approach at a time for Transformers. It also outperformed a similarly sized Transformer-XL.

Longformer

Beltagy et al. [9] introduced a Longformer architecture that contains an attention layer that scales linearly with the input size. It consists of a *sliding window* attention mechanism that attends to $\frac{1}{2}w$ tokens either side of the given token, with w being the window size, a hyperparameter. This approach gets extended to the *dilated sliding window*, which includes gaps of size d in between tokens being attended to, in order to increase the scope of attention. For an attention head h and token i, this can be expressed as:

$$A_{h,i} = \{j : |i - j| \mod (d_h + 1) == 0\} \tag{2.30}$$

Each attention head is configured to have a different dilation d_h. The Longformer dilated attention pattern is presented in Figure 2.9. Additionally, all tokens attend to the global BERT tokens, such as `[CLS]` or `[SEP]`. We note that the dilated attention requires a memory and compute efficient custom CUDA kernel and at the time of writing is not available in the HuggingFace library [47].

This combination of dilated sliding window and global attention led to Longformer outperforming previous models, as it was able to extract both short and long distance relationships, as well as build global representations using the global tokens. Maximum sequence length used was 32,256.

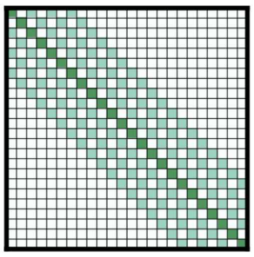


Figure 2.9: Dilated Attention Pattern used by Longformer, as presented by Beltagy et al. [9]. The main diagonal represents each token attending to itself, while the off-diagonal coloured elements represent other non-zero values. In this case, dilution $d_h = 1$.

Big Bird
Zaheer et al. [8] introduced Big Bird, another model that relies on the attention that uses a (non-dilated) sliding window approach and attendance to global tokens. Additionally, Big Bird attention also attends to r random tokens. The attention pattern is presented in Figure 2.10. In the experiments by Zaheer et al. [8], Big Bird outperformed Longformer architecture, however, we note that it has been evaluated on a much lower maximum sequence length of 4096.

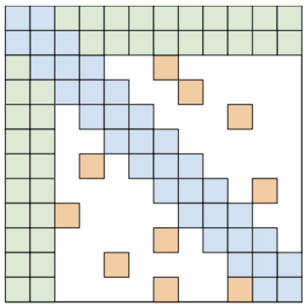


Figure 2.10: Attention patterns of Big Bird. Sliding Window Attention (blue), Global Attention (green) and random attention (yellow). As presented by Zaheer et al. [8].

Other novel architectures
We also note the ongoing effort in developing novel long document Transformers architectures. Reformer [48] and Linformer [49] propose improvements to the self-attention mechanism that decrease its computational complexity from $O(n^2)$ to $O(nlogn)$ and $O(n)$ respectively. Zhu et al. [50] propose Long-Short Transformer that includes short-term attention coupled with dynamic projection to long-range attention that models long-term dependencies. Scaling Transfomers [51] levarage sparsity to improve the speed of long text Transformers, while Hourglass [52] propose a hierarchy of differently-sized Transformer layers to model long documents.

We choose to focus on Big Bird and Longformer due to their widespread use and availability in the HuggingFace library [47].

2.3 Explainability

Explainability techniques can be broadly divided into two groups: white-box and black-box approaches [53]. The former involves modification to the model architecture in order for it to provide explanations of the decisions [21, 54, 55]. The latter operate on already trained models and attempt to learn the decision boundary or explore internals of the model [56, 57, 58].

Since the release of the first attention models [32], attention was explored as a potential method to understand and explain which parts of input the model focuses on [21, 53]. In particular, Thorne et al. [53] found that black-box approaches, such as LIME [56], perform better than white-box approaches, but also bring a 1000× increase in computation for biLSTMs evaluated on the task of Nat-

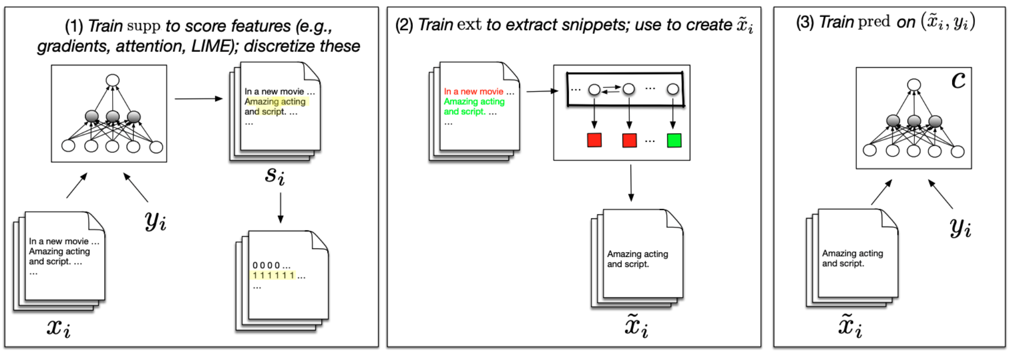

(1) Train supp to score features (e.g., gradients, attention, LIME); discretize these
In a new movie ... Amazing acting and script. ...
y_i
x_i
In a new movie ... Amazing acting and script. ...
s_i
0 0 0 0 ... 1 1 1 1 1 1 ...
(2) Train ext to extract snippets; use to create $\tilde{x}_i$
In a new movie ... Amazing acting and script. ...
Amazing acting and script.
$\tilde{x}_i$
(3) Train pred on $(\tilde{x}_i, y_i)$
c
Amazing acting and script.
y_i
$\tilde{x}_i$

Figure 2.11: The 3 independent stages of the FRESH architecture: (1) the support model (*supp*), (2) the rationale extractor (*ext*), (3) the classifier/predictor (*pred*). We can observe how in (1) a model is used to classify a text, obtain importance scores and discretise them. (2) then selects a snippet containing only a subset of tokens, which become faithful explanation for the prediction in (3). As presented by Jain et al. [19].

ural Language Inference.

2.3.1 Attention (not) as Explanation

With the release of Transformers, there have been efforts to understand if the multi-head self-attention layers already provide satisfactory explanations. Clark et al. [6] qualitatively evaluated the attention heads and found them to learn how to attend to certain linguistic features. Bujel et al. [15] showed that some of the top self-attention layers learn to attend to correct tokens, but also that the adaptation of a soft attention layer proposed by Rei and Søgaard [21] still outperforms the self-attention layers on the task of zero-shot sequence labelling. Atanasova et al. [14] explored various black-box explainability techniques and found that a gradient-based approach `InputXGradient` [59] usually performs best.

Rudin [60] made an argument against black-box explainability approaches in favour of designing *interpretable* models. They introduced the key concept of *faithfulness*, a property of explanations that indicates they are true to the system's computation. DeYoung et al. [20] defined *plausible* explanations as explanations agreeable to human annotators and argued that an explainable system should provide faithful and plausible explanations. Following the work, there have been a number of efforts to obtain both faithful and plausible explanations of Transformers using attention [61, 62].

The notion that attention is indeed an explanation was challenged by Jain and Wallace [63], who showed only weak correlation between attention weights and the gradient-based explanations or model outputs. In response, Wiegreffe and Pinter [64] showed that some models may have non-unique explanations and that attention can serve as explanation as long as it is *plausible*.

2.3.2 Faithful Rationale Extraction & Plausibility (FRESH)

Jain et al. [19] proposed FRESH (Faithful Rationale Extraction from Saliency tHresholding) model that provides faithful rationale 'by construction', while also allowing for the evaluation of plausibility against human annotators.

This is achieved by having 3 independent components: the *support model*, the *rationale extractor* and the *classifier*. The support model (e.g. BERT+LIME) is trained to predict the target classification label from the whole input text and provide token-level importance scores that are then discretised. The rationale extractor is a model trained to extract a subset of tokens as rationale, based on the discrete importance scores. The classifier (e.g. another BERT) is trained only on the tokens provided by the rationale extractor. These 3 stages are illustrated in Figure 2.11. In our work, we focus on the support model and the rationale extractor components.

FRESH architecture uses BERT as the support model. Averaged self-attention scores from the $[CLS]$ token to other tokens in the penultimate layer are used as token importance scores. In order to discretise the rationales, a Top-k heuristic is used, which selects k% of tokens with the highest scores, where k is set the percentage of annotations in the development dataset.

It is important to note that the extracted rationales are not necessarily faithful to the predictions of the *support model*, but they are faithful to predictions made by the *classifier*, as these are the only tokens the model sees. This setup allows for the quantitative evaluation of plausability of rationales against human annotators, similar to the work of Fomicheva et al. [65] in evaluating the plausibility of extracted rationale from Translation Error Prediction models.

Jain et al. [19] define the obtained rationales as *explainability* and we follow their definition of faithful and plausible rationales being equivalent to explainability. In particular, this work focuses on designing support models and rationale extractors for long documents that provide plausible rationales that could then be quantitatively evaluated against human annotations.

2.3.3 Sequence labellers as rationale extractors

We highlight that Rei and Søgaard [21] evaluated soft attention models in the context of zero-shot sequence labelling, without investigating the architecture as a rationale extractor. However, we note that Pruthi et al. [66] treats the problem of plausible rationale extraction for sentiment analysis as a sequence labelling task that is evaluated using the human annotations on the token-level. Similarly, Aly et al. [67] approached the problem of extracting rationale from fact verification systems as a sequence labelling task. We therefore assume that the 2 tasks of sequence labelling and rationale extraction are equivalent. That indicates that measuring the plausibility of the extracted rationale is synonymous to evaluating the sequence labelling performance against the human annotations.

2.4 Ethical Discussion

In our experiments, we use large pre-trained Transformers language models. It has been shown that training one of those models can emit approximately $626,000$lbs of carbon dioxide, nearly 5× as much as an average American car during its lifetime [68]. However, we note that we do not perform any pre-training ourselves and use a downloadable pre-trained version of those models that is shared by the whole NLP community. The fine-tuning of these pre-trained models does not require anywhere near as much energy. By exploring how to effectively use already existing standard-size Transformers, we also aim to find an architecture that does not require custom pre-training in order to be applied to long documents, thus aiming to reduce carbon emissions from pre-training the long document Transformers.

We acknowledge the debate about potential misuse of large language models that ensued following the publication of GPT-2. These models have been shown to have capacity to generate biased and abusive texts, therefore only a simplified version of them was released [69]. As the models that our research is based on are already available online via the HuggingFace library [47] and we focus on document classification rather than generation, we do not foresee any additional risks in using those or extending them to large documents.

Lastly, we note that the explanations our models provide are not unique or necessarily faithful to the model's actual behaviour. This might cause legal disputes following the 'Right To Explanation' clause included in General Data Protection Regulation (GDPR) [70]. We note the work of Wachter et al. [71], where they claim that GDPR only mandates the right to a *meaningful information* about the *logic involved* and the *significance* in the automated decision making. They argue that the clause is equivalent to providing a general system description (e.g. a Transformer document classifier), rather than providing specific explanation to the individual decision that was made. Given this reasoning, we believe that our system is not at risk of legal challenges related to the GDPR, as it does not provide an 'explanation' in the context of the said regulation.

Chapter 3

Soft Attention Architectures

In this chapter, we aim to adapt the Weighted Soft Attention architecture proposed by Bujel et al. [15] to long text binary classifiers. We propose a family of soft attention architectures that are supervised on the document level and provide zero-shot token-level scores $\widetilde{a_i}$ without any token-level annotations. These token scores could then serve as rationale for the task of document classification. To the best of our knowledge, this is a first attempt to evaluate soft attention architectures in combination with long text Transfomers at the document-level. We also propose a novel Compositional Soft Attention system that sequentially applies a standard Transformer to individual sentences to obtain token-level contextual embeddings which are then composed by a soft attention layer to build document-level representation.

The main contributions in this chapter are as follows:

- We adapt the Weighted Soft Attention from token-level rationale extraction for sentence classifiers to long text classification.
- We introduce a novel Compositional Soft Attention system that composes token-level representations of isolated sentences to build document representations.
- We evaluate the soft attention architectures in combination with long text Transformers on both the document and token-level.
- We propose several modifications to the Weighted Soft Attention model to allow for more supervision signal on the token-level and improve the quality of the extracted rationale.

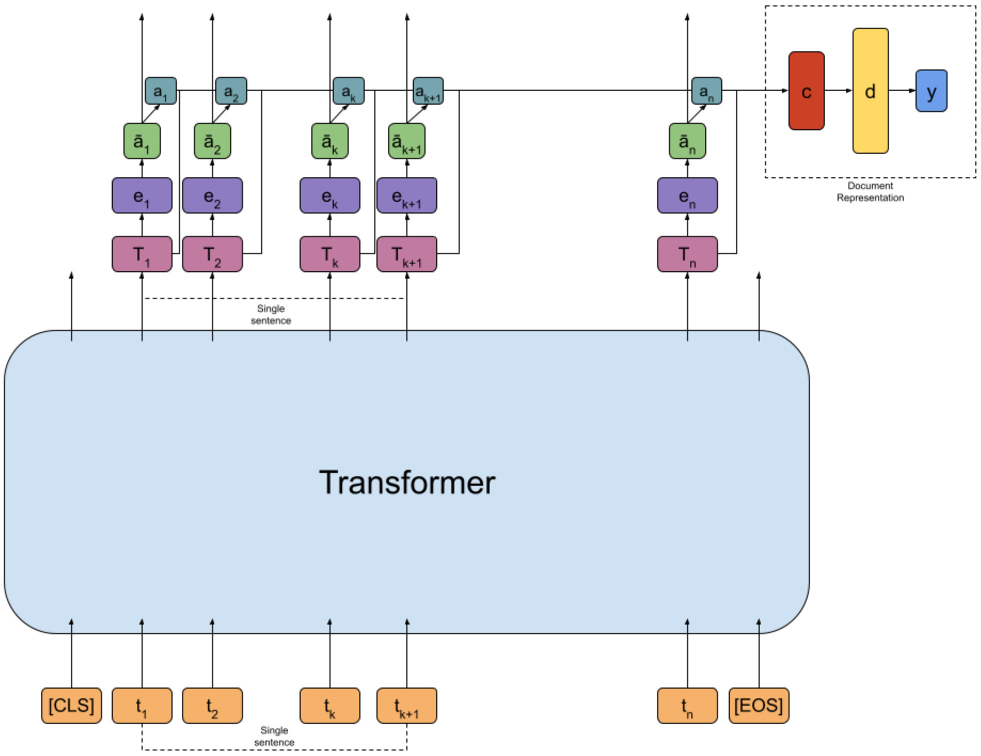


Figure 3.1: Weighted Soft Attention architecture for long text classification. t_i and T_i indicate flattened input tokens and token contextual embeddings respectively. All other variables are as described in Section 3.1. $[CLS]$ and $[EOS]$ token outputs are ignored.

3.1 Weighted Soft Attention

Rei and Søgaard [21] introduced a soft attention architecture for biLSTM sentence classifiers, which Bujel et al. [15] adapted to Transformers by introducing **Weighted Soft Attention** (§2.2.8). To the best of our knowledge, this system had not been evaluated before on the document-level. We therefore decided to implement it as our baseline.

We follow the work of Bujel et al. [15]. In order to adapt this sentence classifier to perform document classification, the input document is flattened to a single sequence of tokens. We use the Transformer output tokens T_i as input for the soft attention layer:

$$e_i = \tanh(W_e T_i + b_e) \tag{3.1}$$
$$\widetilde{a_i} = \sigma(\widetilde{e_i}) \quad \widetilde{e_i} = W_{\widetilde{e}} e_i + b_{\widetilde{e}} \tag{3.2}$$

where e_i is a vector, $\widetilde{e_i}$ is a single scalar value and $\widetilde{a_i}$ is the token attention score. We can convert these scores to normalised attention weights a_i and in combination with token contextual embeddings build a document representation c that is used to predict a document-level label y:

$$a_i = \frac{\widetilde{a_i}^{\beta}}{\sum_{j=1}^{N} \widetilde{a_j}^{\beta}} \qquad c = \sum_{i=1}^{N} a_i T_i \tag{3.3}$$

$$d = \tanh(W_d c + b_d) \quad y = \sigma(W_y d + b_y) \tag{3.4}$$

For each document j, we obtain document-level predictions $y^{(j)} \in [0,1]$ and the token-level scores $0 \le \widetilde{a_i} \le 1$. These token-level scores can be converted to predictions using a natural threshold of 0.50. The token-level predictions can be interepreted as token-level rationale (§2.3.3). We evaluate the plausability of the extracted rationale the against the human annotations. Figure 3.1 represents the architecture.

We recall that the soft attention architecture, uses the following loss functions:

$$L_1 = \sum_j (y^{(j)} - \tilde{y}^{(j)})^2 \tag{3.5}$$
$$L_2 = \sum_j (min(\widetilde{a_i}) - 0)^2 \tag{3.6}$$
$$L_3 = \sum_j (max(\widetilde{a_i}) - \tilde{y}^{(j)})^2 \tag{3.7}$$
$$L = L_1 + \gamma(L_2 + L_3) \tag{3.8}$$

where L_1 optimises the document-level performance, L_2 ensures the minimum attention score is close to 0 and L_3 optimises the maximum attention score to be close to the document label $\widetilde{y}^{(j)}$. γ is a hyperparameter that sets the importance of the token-level objectives.

3.2 Regularised Soft Attention

In our initial experiments, we found that the Weighted Soft Attention architecture does not work well for long documents (Figure 6.1). Additionally, we discovered that most token scores stay close to 1, even for negative documents. We suspect that using a loss function that only selects token scores with the maximum or minimum values causes the tokens to not receive enough supervision signal, as only 2 tokens are optimised in each step. This was not an issue for sentence classifiers due to the smaller number of tokens in each sample.

In order to increase the supervision signal, we propose a **Regularised Soft Attention** architecture that modifies the loss function L (Eq. 3.8) to include a new term L_{reg} that acts as an $L2$ regularisation method [72] on the token scores:

$$L_{reg} = \sum_j \sum_i \widetilde{a_i}^2 \tag{3.9}$$
$$L = L_1 + \gamma(L_2 + L_3) + \gamma_{reg} L_{reg} \tag{3.10}$$

where j is a document index, i is the index of a token in the j-th document, γ_{reg} is a hyperparameter to control the weight of the regularisation signal. L_1, L_2, L_3 are defined as before in Eq. 3.5-3.7.

We hope that this method encourages most tokens to stay close to 0 instead of 1 and provides an additional supervision signal to all tokens, instead of optimising the maximum and minimum values. We acknowledge that L_{reg} essentially optimises all values to be 0, but we hope that by finding a balance between γ and γ_{reg} weights, the system learns to extract correct tokens, while keeping most token scores close to 0.

3.3 Mean Soft Attention

We recognise that the main limitation of the Regularised Soft Attention is the assumption that most tokens should be close to 0. The regularisation method allows for one token per optimisation step to be optimised towards the same label as the document, while most will get supervised towards 0. This means that there might still be little signal for all of the positive tokens to be be optimised towards 1 in positive documents.

As an alternative, we propose **Mean Soft Attention**, which replaces the regularisation loss L_{reg} with a mean loss L_{mean} that optimises the mean of the token scores to be close to the document label:

$$L_{mean} = \sum_j \left(\frac{1}{|\widetilde{a}|} \sum_i \widetilde{a_i} \right) - \widetilde{y}^{(j)} \tag{3.11}$$

$$L = L_1 + \gamma(L_2 + L_3) + \gamma_{mean} L_{mean} \tag{3.12}$$

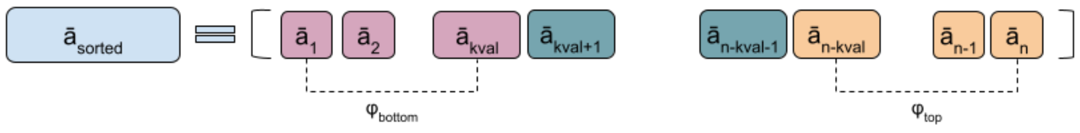
where γ_{mean} is a hyperparameter.

We note that this approach is equivalent to supervising each token individually towards the document label. While not ideal, this is preferable to the Regularised Soft Attention, which imposes an arbitrary condition that most tokens need to be close to 0. Unlike Weighted Soft Attention, the Mean Soft Attention also allows for all tokens to receive some supervision signal in each optimisation step. We hope this encourages various token scores, as opposed to the case of Weighted Soft Attention where all scores are close to 1.

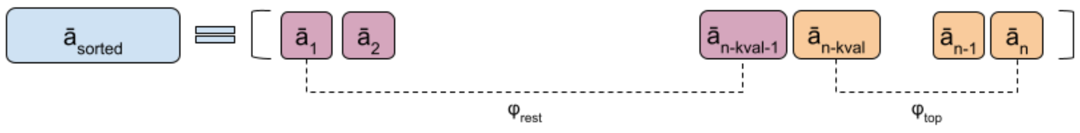
3.4 Top-k Soft Attention

Regularised Soft Attention and Mean Soft Attention attempt to increase the supervision signal available to individual token scores. These architectures propose supervision of all token scores with the same value, which might encourage homogeneity across the token scores. The human annotations on the other hand are binary, with more negative than positive tokens (Table 4.2). Therefore, we suggest that a loss function encouraging a distribution of a portion of token scores towards the document label would perform better.

We propose **Top-k Bottom-k Soft Attention**, an approach where only k% of tokens with the highest scores are optimised to the document label, while the lowest k% tokens are supervised with a 0, as pictured in Figure 3.2(a). This is essentially an extended version of the original Weighted Soft Attention that allows for



(a) *Top-k Bottom-k method.* ϕ_{bottom} contains bottom $k\%$ of tokens, while ϕ_{top} contains top $k\%$ of tokens. All of the not selected tokens do not receive any supervision.



(b) *Top-k Rest-0 method.* ϕ_{top} contains top $k\%$ of tokens, while ϕ_{rest} contains the remaining $100 - k\%$ of tokens. All tokens receive supervision.

Figure 3.2: Comparison of Top-k Bottom-k and Top-k Rest-k Soft Attention methods. Assuming $\widetilde{a}$ is sorted in increasing order. Assuming $kval = \lfloor k\% \times n \rfloor$.

an increased supervision signal instead of supervising the minimum and maximum token scores only.

We define two subsets ϕ_{top} and ϕ_{bottom}, which contain the top $k\%$ and bottom $k\%$ of token attention scores $\widetilde{a_i}$ respectively. We then define the loss function $L_{top/bottom}$ with respect to those subsets:

$$L_{top/bottom} = \sum_j \frac{1}{|\phi_{top}|} \sum_{a_{top} \in \phi_{top}} (a_{top}) - \widetilde{y}^{(j)} + \frac{1}{|\phi_{bottom}|} \sum_{a_{bottom} \in \phi_{bottom}} a_{bottom} \tag{3.13}$$

$$L = L_1 + \gamma(L_2 + L_3) + \gamma_{top/bottom} L_{top/bottom} \tag{3.14}$$

where $\gamma_{top/bottom}$ is a hyperparameter and k is a hyperparameter that needs to be tuned, or can be inferred based on the percentage of annotations in the dataset [19].

We note that the Top-k Bottom-k Soft Attention architecture is still prone to little supervision signal being available to most tokens in documents where the proportion of positive tokens is low. Therefore, we introduce a variation of Top-k Soft Attention that instead supervises top $k\%$ of tokens to be the same as the document label, while supervising the remaining $100\% - k\%$ token scores to be 0. We call this variant **Top-k Rest-0**, with ϕ_{top} and ϕ_{rest} which contain the top $k\%$ and the remaining $1 - k\%$ of token attention scores $\widetilde{a_i}$ respectively, so that $\widetilde{a} = \phi_{top} + \phi_{rest}$. We define its loss function $L_{top/rest}$ as follows:

$$L_{top/rest} = \sum_j \frac{1}{|\phi_{top}|} \sum_{a_{top} \in \phi_{top}} (a_{top}) - \widetilde{y}^{(j)} + \frac{1}{|\phi_{rest}|} \sum_{a_{rest} \in \phi_{rest}} a_{rest} \tag{3.15}$$

$$L = L_1 + \gamma(L_2 + L_3) + \gamma_{top/rest} L_{top/rest} \tag{3.16}$$

where $\gamma_{top/rest}$ is a hyperparameter.

We believe this setup ensures that all of the tokens receive some supervision signal, while still ensuring heterogeneity of the token scores. The differences and similarities in both Top-k approaches are shown in Figure 3.2. In our experiments, we use Top-k Rest-0 Soft Attention architecture, as we found that for each dataset ≈10% of all tokens are marked as evidence, which would leave to 80% of tokens not receiving supervision signal in the Top-k Bottom-k system.

3.5 Compositional Soft Attention

3.5.1 Motivation

In order for the soft attention architectures to support long texts, we had to adapt them to encourage the propagation of the token-level supervision signal to more tokens. Methods known to perform well for Transformer sentence classifiers did not work well for document classifiers based on long text Transformers.

These long text models additionally require large GPUs in order to be pre-trained, with the original Longformer using 48GB Nvidia RTX8000 GPUs [9], which can be prohibitively expensive for many research groups[1]. In our experiments, we encountered issues with fine-tuning the Longformer model on the 11GB Nvidia RTX2080Ti and had to use to a 16GB Nvidia Tesla T4, on which we only managed to fine-tune with a batch size of 1. Additionally, the HuggingFace library does not currently contain the dilated attention version of the Longformer due to the requirements for a custom CUDA kernel. For BigBird, we did not manage to fit the model and data on the 16GB Tesla T4.

The long text Transformers also require more time to be trained and fine-tuned, with the Longformer taking up to 1 day, compared with BERT taking at most a couple of hours [2]. It has been reported that training standard Transformers is not sustainable, given that it emits approximately $626,000$lbs of carbon dioxide[2] [68]. The longer training times and larger memory requirements of long text Transformers suggest a substantial increase in emissions of the training process for those models.

Therefore, it would be preferable to re-use the already pre-trained standard Transformers (e.g. BERT). As our core motivation, we would like to find an architecture that can utilise the already widely-used standard Transformers and exploit their sentence-level performance to represent long documents.

Zhang et al. [13] introduced HIBERT (§2.2.9), one of the first successful approaches to use BERT to build representations of documents beyond BERT's standard maximum sequence length. The core of the proposed architecture was the composition sentence representations obtained from the output $[EOS]$ tokens of BERT. While this approach performed worse than Longformer, it required less GPU memory and outperformed BERT. In order to combine the sentence representations, HIBERT used a custom Transformer layer that did not allow to obtain

[1]USD10,000 per GPU at the time of their release

[2]Nearly 5× as much as an average American car during its lifetime

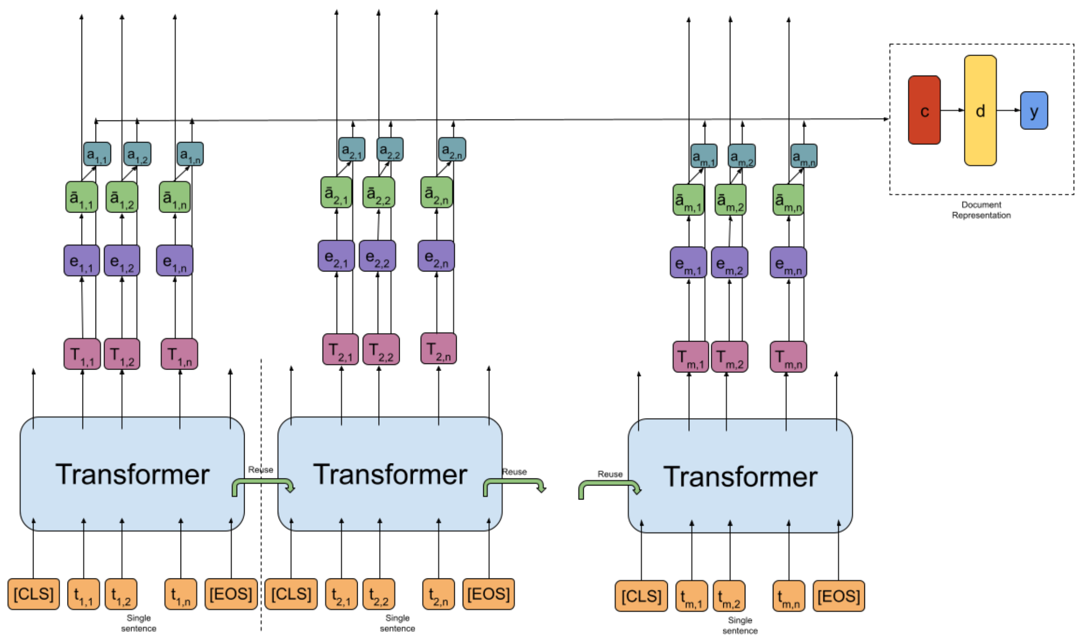


Figure 3.3: Compositional Soft Attention architecture. A Transformer is sequentially applied to each sentence individually to obtain the contextual token embeddings. Then, a soft attention layer is used to compose the token representations across the sentences to build document representation.

any token-level representations other than the self-attention weights.

We believe that the complexity and scale of standard Transformers such as BERT or RoBERTa is sufficient to represent long documents beyond Transformers' standard maximum sequence length. We further claim that in order to represent long documents, it should not be required to attend to individual tokens across the whole documents using multiple layers and heads of attention. We hope to find a system that is able to efficiently compose these individual token representations with a single soft attention layer.

In this chapter, we propose a Compositional Soft Attention architecture, which allows for the composition of token contextual embeddings obtained from a Transformer model applied to each sentence in the document individually to produce a document-level representation.

We hope that this novel architecture can reduce the computational resource consumption of long text classification models, while also maintaining similar performance. We further aim to show that in some scenarios, models based on standard Transformers can perform as well as larger and slower long document Transformers.

3.5.2 Architecture

We introduce **Compositional Soft Attention** architecture that applies a soft attention layer to compose contextual token embeddings of each sentence to build a document-level representation.

We assume a document doc_j that consists of a sequence of m sentences doc_j, where each sentence $s_{j,i}$ is a sequence of tokens $t_{i,k}$:

$$doc_j = [s_{j,1}, ..., s_{j,i}, ..., s_{j,m}] \tag{3.17}$$

$$s_{j,i} = [t_{i,1}, ..., t_{i,k}, ..., t_{i,n}] \tag{3.18}$$

We first use a standard length Transformer to build token contextual embeddings $T_{j,i}$ separately for each sentence.. These token embeddings are packed to create a token-level document representation $T_j = [T_{j,1,1}, ..., T_{j,m,n}]$, with outputs for special tokens $[CLS]$ and $[EOS]$ omitted. We then provide this document representation T_j as input to a soft attention layer, which composes tokens across sentences to obtain a document-level representation and the prediction $y^{(j)}$. We additionally obtain token-level attention scores $\widetilde{a_i}$, which we use to extract rationale for the document classification task. The algorithm is visualised in Figure 3.3 and its pseudocode is in Algorithm 1.

Algorithm 1 Compositional Soft Attention

```
for sentence $s_{j,i}$ in document $doc_j$ do
    $T_{j,i} \leftarrow$ Transformer($s_{j,i}$)
end for
$T_j \leftarrow [T_{j,1}, ..., T_{j,n}]$
$\tilde{y}^{(j)}, \widetilde{a} =$ SoftAttention($T_j$)
```

We note that the proposed Compositional Soft Attention architecture can support any variant of the soft attention presented in Chapter 3. Any Transformer can be used to obtain the contextual embeddings. This architecture can be directly fine-tuned on the task of document classification without the need for any pre-training. We emphasise that during training, both the soft attention and the Transformer components are optimised, similar to previous soft attention architectures.

We hope that using the sentence-level Transformer permits the exploitation of intra-sentence dependencies, while the soft attention layer models and composes the inter-sentence relationships in order to build a document representation. We suspect this approach will exhibit lower runtimes than long text Transformers that contain multiple layers of self-attention over the whole document.

3.5.3 Batching

As part of the Compositional Soft Attention model, there are 2 distinct types of batching we can perform:

1. *Sentence-level batching:* It is possible and preferable to batch multiple sentences together as input to the standard Transformer. That permits us to benefit from the efficiency of GPUs and reduce the runtime by avoiding the application of Transformer sequentially to individual sentences. We call the size of this batch *Compositional Sentence Batch Size*.

2. *Document-level batching:* We can additionally batch multiple documents together. In this case, we collect the contextual token embeddings obtained

from Transformer for multiple documents. Once all of the contextual embeddings are collected, we apply the soft attention layer to all documents at once, calculate the loss and propagate it backwards. We call this *Compositional Document Batch Size*.

For sentence-level batching, we pad the input tokens to the size of the longest sentence in the batch. For document-level batching, we pad contextual embeddings with vectors of zero to the length of the document with most tokens, before providing as input to the soft attention layer. This padding has no impact on the soft attention model, as it uses a masking method where attention scores $\widetilde{a_i}$ are non-zero only for non-padded positions present in the original input.

During our initial experiments, we found that this batching system improved the runtimes of the Compositional Soft Attention by aproximately 10%.

Chapter 4

Datasets

In this chapter, we introduce the requirements posed for the datasets to be used for long document classification and rationale extraction. We overview the particular tasks we decided to focus on, together with the suitable datasets. We further provide details of the necessary preprocessing steps taken in order to bring all datasets to a common format.

4.1 Requirements

4.1.1 Document Classification & Rationale Extraction

The proposed architectures (§3) are designed as binary document classifiers that provide token-level scores that can be interpreted as model rationales. In order to quantitatively evaluate the performance on both the token and document- level, we require document classification datasets that contain long documents of text with binary token-level annotations that are human rationales for the document label.

Following the report by Byrd and Lipton [73] on slower convergence of BERT training for imbalanced datasets, we prefer datasets with balanced document label classes. In order to capture the performance difference between standard and long text Transformers, we also need some proportion of the documents to be longer than 512 tokens, the standard Transformer maximum token length. Datasets satisfying those conditions are scarce, due to the costs of obtaining human annotations for long documents.

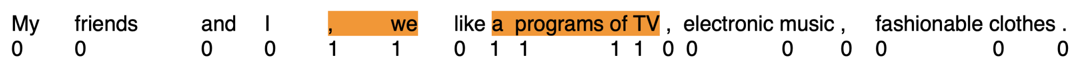
4.2 Tasks

4.2.1 Grammatical Error Detection

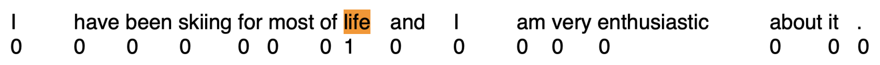
We evaluate our architectures on 2 distinct Grammatical Error Detection (GED) datasets, which identify tokens that are grammatically incorrect, while also containing a document-level label indicating the level of language proficiency of the learner.

Write & Improve [74] dataset was released as part of the **BEA 2019**[1] shared task [75]. It contains English essays written by learners in response to various prompts. It was manually annotated on the token-level with grammatical errors

[1] https://www.cl.cam.ac.uk/research/nl/bea2019st/


My friends and I , we like a programs of TV , electronic music , fashionable clothes .
0 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0 0

(a) An example of a BEA 2019 sentence from an essay with proficiency level *A* (beginner). Numerous grammatical errors highlighted.


I have been skiing for most of life and I am very enthusiastic about it .
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

(b) An example of a BEA 2019 sentence from an essay with a proficiency level *C* (advanced). Contains one small grammatical error of a missing the "my" pronoun.

Figure 4.1: Example sentences from the BEA 2019 dataset. The first row contains the individual words. The second row shows the human annotations, with 1 representing a grammatical error and 0 representing no error. We additionally highlight the words with grammatical errors.

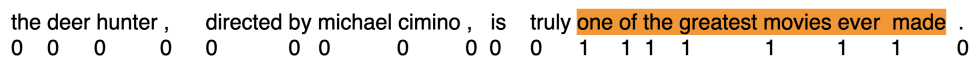
and contains a document-level label to indicate the CEFR proficiency score of the author: *A* (beginner), *B* (intermediate), *C* (advanced). As the publicly available dataset does not contain a test split, we use the development dataset as test and use 10% of the training data for development. In order to perform binary document classification, we select documents with the *A* proficiency score as the positive class and with the proficiency score *C* as the negative class. Example excerpts from positive and negative documents are presented in Figure 4.1.

The First Certificate in English dataset[2] (**FCE**) [76] contains essays written by non-native English learners. Each student wrote 2 essays in response to 2 different prompts and was allocated an overall mark by the examiners. We concatenate the 2 essays into a single data point in order to increase the average sequence length. We convert the data into a balanced binary classification dataset with the essays of lowest and highest marks grouped in separate classes. This is done by thresholding on learner scores s_i, with $0 \leq s_i \leq 26$ for the positive (beginner) class and $30 \leq s_i \leq 40$ for the negative (advanced) class. We decided to omit the essays with scores $27 \leq s_i \leq 29$ to mimic the omission of the proficiency level *B* (intermediate) for the BEA 2019 dataset. We use the train/dev/test splits released by Rei and Yannakoudakis [77].

We suspect that for both BEA 2019 and FCE, the proportion of grammatical errors in human annotations for the beginner learners is higher than for the advanced ones. We verify the assumption in Table 4.1, where we find it to be the case. This allows us to assume that we should encourage the models to attend to more tokens for documents with positive (beginner) label.

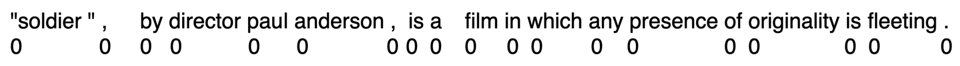
	FCE	BEA 2019
Negative Class (Advanced)	9%	5%
Positive Class (Beginner)	17%	20%

Table 4.1: Proportion of evidence tokens for the development datasets of BEA 2019 and FCE, reported separately for the negative and positive document labels.



```
the deer hunter ,   directed by michael cimino ,  is   truly one of the greatest movies ever  made  .
0   0    0      0   0        0  0       0      0  0  0     1   1  1   1        1      1     1     0
```

(a) An excerpt from a positive review, with the positive tokens highlighted.



```
"soldier " ,  by director paul anderson ,  is a   film in which any presence of originality is fleeting .
0        0 0  0  0        0    0        0  0  0   0    0  0     0   0        0  0           0  0        0
```

(b) An excerpt from a negative review, without any negative tokens annotated, because we want to avoid the model to learn to find negative evidence when classifying positive movie reviews.

Figure 4.2: Example excerpts from the IMDB-Pos dataset that contains evidence for all positive reviews, but no evidence for negative reviews. The first row in each subfigure represents the review sentence, with the second row representing annotations, where 1 determines the evidence words. The evidence words are also highlighted.

4.2.2 Sentiment Analysis

We also evaluate the systems on the Sentiment Analysis **IMDB**[3] dataset, which contains IMDB movie reviews collected by Zaidan et al. [78]. The reviews are labelled as positive and negative and were manually annotated on the token-level with rationale for the document-level label. We further split the dataset into **IMDB-Pos** and **IMDB-Neg**, respectively containing token-level annotations for the positive and negative reviews only. This is to allow the model to learn to attend to either positive or negative tokens separately. Example excerpts of a positive and negative movie reviews from IMDB-Pos are presented in Figure 4.2. We use the dev/train/test splits published by Pruthi et al. [66].

4.3 Statistics

We present detailed statistics for all 4 datasets in Table 4.2. We note the low proportion of texts over 512 words for FCE and BEA 2019, which indicates standard Transformers would also perform well for those datasets. We recognise that the word count is not necessarily equal to the post-tokenisation token count, but decide that reporting word count is sufficient to illustrate general dataset trends. While we acknowledge that there is little benefit to using long text transformers for the GED datasets, we decide to treat them as benchmarks that highlight the performance differences between standard and long text Transformers. We were not able to obtain any better suited datasets.

	FCE	BEA 2019	IMDB-Pos	IMDB-Neg
Number of train samples	722	1120	1200	1200
Number of dev samples	51	280	299	299
Number of test samples	66	200	300	300
Average text length (words)	441	213	686	686
Maximum text length (words)	725	655	1935	1935
% of texts > 512 words	16%	2%	73%	73%
% positive samples	49%	46%	50%	50%
% negative samples	51%	54%	50%	50%
% evidence	13%	9%	8%	8%

Table 4.2: Statistics for the datasets used. All measured on the development datasets.

[2]https://ilexir.co.uk/datasets/index.html
[3]https://github.com/danishpruthi/evidence-extraction/tree/main/datasets

4.4 Preprocessing

Each dataset is available in a different data format, with FCE and BEA 2019 using JavaScript Object Notation (JSON) [79] and IMDB using plaintext TXT files. We therefore decided to introduce our custom schema and convert all of the datasets to it. The schema is based on the JSON file format, where all documents are stored in an array as *Document* JSON objects (Listing 4.1).

Parts of the data processing code are based on the preprocessing performed by Rei and Yannakoudakis [77] for FCE & BEA 2019 and by Pruthi et al. [66] for IMDB.

Listing 4.1: The schema to represent *Document* objects in JSON. It is used to store individual input text. *"id"* represents the original dataset document id, while *"document_label"* indicates the document class. Tokens are represented as *"tokens"*, a nested list of strings, where each list of strings is a sentence. *"token_labels"* similarly contains a numerical label for each token.

```
{
    "id": String,
    "document_label": Integer,
    "tokens": [[String]],
    "token_labels": [[Integer]]
}
```

Chapter 5

Experimental Setup

In this chapter, we introduce our experimental setup, together with the baselines we use to compare the performance of the architectures introduced.

5.1 Baselines

We use **RoBERTa**-base [3] and **Longformer**-base [9] as document classification baselines. For RoBERTa, we use the default 512 maximum sequence length, while for Longformer we set 4096 tokens as the maximum length. Each batch is padded with $[PAD]$ tokens to the length of the longest document in the batch. A text classification setup for Transformers is described in §2.2.7.

Longformer architecture does not include the dilated self-attention, as it is currently not supported by HuggingFace and requires a custom CUDA kernel[1]. We note that the performance differences between the dilated and non-dilated self-attention methods for the original Longformer were minimal [9] and do not expect the omission to have large impact on the final results. We use the local self-attention with the default window size of 512 and also define a global self-attention to the $[CLS]$ token, as that token is used for the document label prediction. We attempted to use BigBird [8], but did not manage to fit it on the 16GB Nvidia Tesla T4.

On the token-level, we use the **FRESH Top-k support model** [19] (§2.3.2) as our baseline for rationale extraction, with Longformer as the support model. This architecture was previously evaluated on the larger version of the IMDB dataset by Pruthi et al. [66], but it is the first time it is evaluated on the FCE and BEA2019 datasets. We follow the previous work and set k% to be equal to the average percentage of evidence in the dataset, rounded to the nearest 10%. The percentage of evidence in the datasets was reported in Table 4.2 and values of k are presented in Table 5.1. We extract the top k% of token scores as rationale, with the rest set to 0, where token scores are the mean attention values from each token to the $[CLS]$ token in the last self-attention layer, averaged across different self-attention heads.

We additionally implement a token-level **random baseline**, where token-level scores are sampled from a standard uniform distribution and tokens with scores

[1] https://huggingface.co/docs/transformers/model_doc/longformer#transformers.LongformerModel

	FCE	BEA 2019	IMDB-Pos	IMDB-Neg
$k\%$	10%	10%	10%	10%

Table 5.1: Percentage of all tokens in the datasets that are marked as evidence. Value of k is chosen by rounding to the nearest 10%. We observe that for all datasets $k = 10\%$, which shows that the percentage of tokens is similar across all datasets.

$\geq 1 - k\%$ are selected as positive. It provides us with means of comparison of the quality of the token-level predictions to a random lower bound.

5.2 Implementation

We implement all architectures in Python 3.9 [80], pyTorch 1.11 [81]. We use the pre-trained versions of RoBERTa-base and Longformer-base available in HuggingFace 4.18 [47]. Some parts of our soft attention implementation are based on the work of Rei and Søgaard [21] and Bujel et al. [15], while the data collation process follows the HuggingFace token classification guide[2].

All models are evaluated on the 11GB Nvidia RTX2080Ti and 16GB Nvidia Tesla T4. All runtimes are measured on the Nvidia Tesla T4.

5.3 Evaluation Metrics

We evaluate the document classification performance using Accuracy (*Doc Acc*) and the F-measure (*Doc* F_1). On the token-level rationale extraction task, we report Accuracy (*Tok Acc*), Precision (*Tok P*), Recall (*Tok R*) and the F-measure (*Tok* F_1), evaluated against the human annotations. We use the following definitions of the metrics:

$$Acc = \frac{1}{n} \sum_j I(y^{(j)} = \tilde{y}^{(j)}) \tag{5.1}$$

$$P = \frac{\sum_j I(y^{(j)} = \tilde{y}^{(j)} \wedge y^{(j)} = 1)}{\sum_j I(\tilde{y}^{(j)} = 1)} \qquad R = \frac{\sum_j I(y^{(j)} = \tilde{y}^{(j)} \wedge y^{(j)} = 1)}{\sum_j I(y^{(j)} = 1)} \tag{5.2}$$

$$F_1 = 2\frac{P \times R}{P + R} \tag{5.3}$$

where I is the indicator function, n is the number of samples, $y^{(j)}$ are the gold labels and $\tilde{y}^{(j)}$ are the predicted labels.

For the soft attention architecture, we use a natural classification boundary of 0.50 on the token attention scores $\widetilde{a_i}$. For the FRESH Top-k Approach, we simply select the $k\%$ of tokens with highest scores as positive.

We perform significance testing using a two-tailed paired t-test and $a = 0.05$. This allows us to assess the statistical significance of the difference in performance.

[2]https://github.com/huggingface/transformers/blob/main/examples/pytorch/token-classification/run_ner.py

5.3.1 Mean Average Precision

We also evaluate the extracted rationale using the Mean Average Precision (MAP). MAP is the mean of Average Precision (AP) achieved for each document, where AP is equivalent to the area under the precision-recall curve:

$$AP_j = \sum_{i}^{n_j} (R_i - R_{i-1}) P_i \tag{5.4}$$

$$MAP = \frac{1}{n} \sum_{j} AP_j \tag{5.5}$$

where AP_j, n_j is the average precision and the number of tokens for document j respectively. P_i and R_i are the precision and recall calculated when token i is selected as the threshold. Average Precision is equivalent to a mean precision value weighted by the increase in recall at each step.

This metric allows us to assess the quality of the token scores irrespective of the chosen threshold or value of k. MAP treats the rationale extraction task as a ranking problem and measures whether the higher scores correspond to positive gold labels rather than the negative ones. For that reason, we evaluate using the pre-thresholding token scores $\widetilde{a_i}$ that allow us to investigate how well the model learns the ranking of tokens. A high MAP score but low F_1 metric might indicate issues with the chosen classification threshold.

5.4 Training Procedure

Following Mosbach et al. [82], we train all models for 20 epochs, with checkpointing after each epoch. We perform early stopping after the loss function has not improved on the development dataset for 5 epochs and a minimum of 10 epochs elapsed, with the best performing checkpoint selected. This setup allows us to ensure that we select the best performing models that did not overfit.

We use the standard Transformers finetuning learning rate $lr = 1e-5$, with a 6% linear warmup and a AdamW optimiser [83]. We use an effective batch size of 32, with gradient accumulation and real batch size of 1 for Longformer and Compositional Soft Attention (document-level batch size) and 8 for RoBERTa. We find optimal values of k for Top-k Soft Attention and of γ for all soft attention models with a hyperparameter search (§6.1). For soft attention architectures, we use $\beta = 2$. All other hyperparameters are reported in Appendix A.

Chapter 6

Evaluation & Discussion

6.1 Development Experiments

In this section, we describe the hyperparameter tuning and other experiments we have performed to initially assess each architecture. Unlike for the final evaluation, we ran the experiments for a single random seed. All of the experiments were performed on the development dataset only.

We also did not perform hyperparameter search on the Longformer-based systems due to the extensive runtimes of more than 5 hours and limited computational resources. We therefore use the same hyperparamaters across both the RoBERTa and Longformer based models and leave further hyperparameter tuning to the follow-up work.

We use these development dataset experiments as an opportunity to validate some of our predictions with relation to the performance of various soft attention architectures. That allowed us to find the best performing ones early on and reduce the number of architectures in the final evaluation and analysis.

6.1.1 Soft Attention

Weighted Soft Attention
We first perform the hyperparameter tuning for the parameter γ in the standard soft attention loss function L (Eq. 2.18). This parameter γ represents the importance given to the token-level objectives of optimising the minimum token scores $\widetilde{a_i}$ to 0 and the maximum token scores $\widetilde{a_i}$ towards the document label. We note that in the previous work for sentence classifiers, Rei and Søgaard [21] found $\gamma = 0.1$ to work best for biLSTMs and Bujel et al. [15] found $\gamma = 1.0$ to be performing best for Transformers. We suspect that the value might be different for document classifiers due to the increased length of the input sequences. We explore the behaviour of the model for $\gamma \in [0.1, 1.0, 10, 100]$.

The results are presented in Table 6.1. We note that the results are stable for various values of γ, with the no substantial differences on the token-level. On the document-level, the performance varies, which we suspect is because the model chooses to first optimise the token-level performance and is unable to improve the document-level predictions further. We choose $\gamma = 10$ as the optimal value mainly due to the stable document and token performance across all datasets.

	FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
γ	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
0.1	**87.50**	22.00	95.45	**17.47**	92.76	6.43	90.66	7.07
1.0	80.95	**22.10**	94.59	17.26	92.36	6.42	90.91	7.07
10	86.36	22.07	**96.62**	17.27	91.56	6.43	**91.03**	**7.09**
100	86.36	22.01	91.91	17.20	**93.25**	**6.44**	88.97	7.06

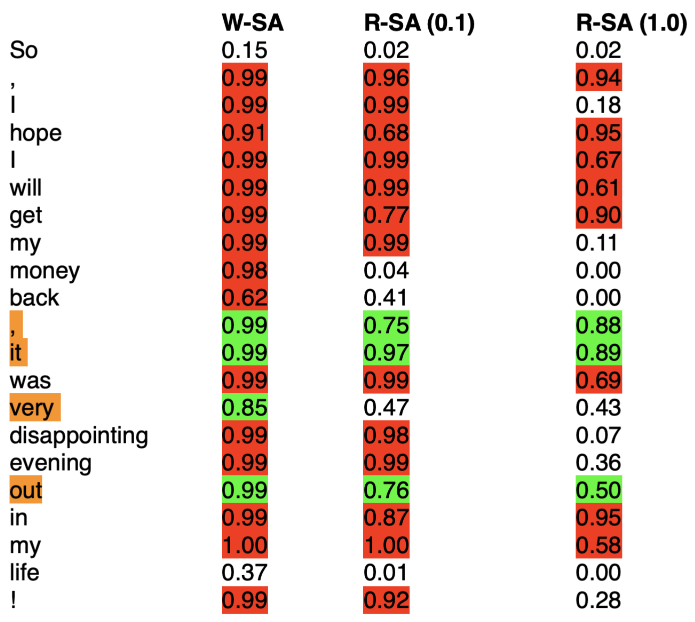
Table 6.1: Hyperparameter tuning of the token-level loss importance γ for Weighted Soft Attention with RoBERTa. All results are for the development datasets. We find no clear best performing value and choose $\gamma = 10$.

Regularised Soft Attention

We also explored different values of the weight γ_{reg} of L_{reg} (Eq. 3.9) for the Regularised Soft Attention Architecture. The aim of the added regularisation was to increase the supervision signal available to each token, while also encouraging a small number of tokens to be close to the document label. For that reason, we explored small values $\gamma_{reg} \in [0.1, 1.0]$ for $\gamma = 10$, following the experiments for the Weighted Soft Attention.

	FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
γ_{reg}	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
0.0	**86.36**	**22.07**	**96.62**	**17.27**	**91.56**	6.43	**91.03**	7.09
0.1	77.27	21.79	95.00	13.21	90.45	**10.40**	89.35	**15.43**
1.0	80.95	7.79	94.43	13.05	90.10	7.85	90.68	12.16

Table 6.2: Hyperparameter tuning of the regularisation token-level loss importance γ_{reg} for Regularised Soft Attention with RoBERTa. All results are for the development datasets. We find that regularisation does not improve the results substantially and decided to pick $\gamma_{reg} = 0.0$ (no regularisation). All experiments are for $\gamma = 10$.



	W-SA	R-SA (0.1)	R-SA (1.0)
So	0.15	0.02	0.02
,	0.99	0.96	0.94
I	0.99	0.99	0.18
hope	0.91	0.68	0.95
I	0.99	0.99	0.67
will	0.99	0.99	0.61
get	0.99	0.77	0.90
my	0.99	0.99	0.11
money	0.98	0.04	0.00
back	0.62	0.41	0.00
,	0.99	0.75	0.88
it	0.99	0.97	0.89
was	0.99	0.99	0.69
very	0.85	0.47	0.43
disappointing	0.99	0.98	0.07
evening	0.99	0.99	0.36
out	0.99	0.76	0.50
in	0.99	0.87	0.95
my	1.00	1.00	0.58
life	0.37	0.01	0.00
!	0.99	0.92	0.28

Figure 6.1: Token scores for a sentence in the FCE dataset for a beginner learner. *W-SA* represents Weighted Soft Attention, *R-SA (0.1)* and *R-SA (1.0)* represent Regularised Soft Attention with $\gamma_{reg} = 0.1$ and $\gamma_{reg} = 1.0$ respectively. orange represents human annotations, red represents false positive token-level predictions and green represents true positives. All models made the correct prediction that this learner is a beginner.

We present the results in Table 6.2. We find that $\gamma_{reg} = 0.1$ performs substantially better on the token-level than γ_{reg}, which we suspect is because the latter provides too much signal that encourages most tokens to be close to 0, causing a reduction in recall. This can be observed in Figure 6.1, where we compare the outputs of the Weighted and Regularised Soft Attention models for the same excerpt. We note that while the Regularised Soft Attention decreases the false positives, it also reduces the number of true positives. We note that the overall performance of Regularised Soft Attention is worse than Weighted Soft Attention ($\gamma_{reg} = 0.0$) for FCE and BEA 2019. Therefore, we decide to omit this architecture in the final evaluation.

Mean Soft Attention

We also evaluate the Mean Soft Attention architecture and try to find optimal values of γ and γ_{mean} (Eq. 3.12). As before, γ represents the importance given to the L_2 (minimum token score) and L_3 (maximum token score) loss objectives and γ_{mean} is the importance given to the mean objective, where the mean of token scores is optimised towards the document label.

We run a grid search hyperparameter search for $\gamma \in [0.1, 1.0, 10, 100]$ and $\gamma_{mean} \in [0.1, 1.0, 10]$. Similarly to the Regularised Soft Attention, we hope that the additional mean loss objective and the resulting increased token-level signal will lead to better results than the Weighted Soft Attention on the token-level, while maintaining similar document-level performance.

		FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
γ_{mean}	γ	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
0.1	0.1	81.63	21.98	96.62	24.20	90.16	8.16	92.57	8.92
	1.0	78.05	21.95	96.47	24.50	90.79	7.98	90.97	9.04
	10	78.05	22.20	95.71	27.97	91.64	7.96	90.78	8.02
	100	83.72	22.02	82.62	13.38	**92.65**	6.82	92.73	8.33
1.0	0.1	86.96	21.96	**97.23**	27.35	91.72	10.36	**93.02**	12.71
	1.0	**91.30**	22.00	93.08	30.96	89.73	11.96	92.98	12.52
	10	76.60	21.18	92.31	**32.18**	92.26	12.27	91.86	14.18
	100	83.72	25.11	80.37	27.03	90.79	**13.22**	90.60	**16.32**
10	0.1	80.95	**28.17**	97.07	30.75	91.69	10.55	92.26	12.81
	1.0	86.96	25.79	96.68	30.72	89.42	11.48	90.78	13.49
	10	81.82	26.26	89.12	30.00	88.15	10.71	90.78	14.63
	100	80.00	22.94	87.46	31.84	89.52	12.53	88.81	15.41

Table 6.3: Hyperparameter tuning of the token-level loss importance for the mean loss (γ_{mean}) and the min-max loss (γ) for Mean Soft Attention with RoBERTa. All results are for the development datasets. We report large variance of results based on the chosen hyperparameter values, noting that $\gamma = 1.0$ tends to perform best. We select $\gamma = 1.0$ and $\gamma_{reg} = 100$.

The results are presented in Table 6.3. We note the variance in results, both on the token-level and the document-level. There is no clear best performing setup, although it seems that $\gamma_{mean} = 1.0$ usually leads to the best results. We choose $\gamma_{mean} = 1.0$ together with $\gamma = 100$.

6.1.2 Top-k Soft Attention

Top-k Rest-0 Soft Attention

We experiment with different values of the hyperparameter k% that determines the percentage of tokens optimised towards the document label. We focus on the Top-k Rest-0 method, where the remaining $1-k\%$ of tokens are all optimised towards 0, maximising the supervision signal available to the tokens. All of the experiments are for $\gamma_{top/rest} = 10$ (Eq. 3.16), following the findings of optimal γ values for Weighted and Mean Soft Attention architectures.

	FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
k	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
10%	68.42	10.38	92.82	15.14	89.54	11.00	90.97	11.78
20%	79.07	16.34	94.32	22.86	**90.45**	10.71	89.63	12.82
30%	83.72	19.16	94.07	19.66	88.44	11.09	88.52	13.20
40%	**93.62**	**24.67**	94.00	**27.79**	89.17	**11.61**	89.19	14.48
50%	83.72	21.44	95.04	25.54	90.00	11.50	**91.03**	**15.02**
60%	88.89	23.54	**96.83**	26.53	90.10	11.45	90.55	14.63

Table 6.4: $k\%$ tuning for the Top-k Rest-0 Soft Attention, where $k\%$ is the percentage of tokens supervised towards the document label. We find that the optimal value varies and pick $k = 50\%$ due to its stable performance and intuition - supervising half of the tokens towards 0 and the other half towards the document label. All experiments are for $\gamma_{top/rest} = 10.0$ and include RoBERTa as a Transformer.

The results are presented in Table 6.4. We note that there is no clear best performing value of k, but that we obtain good results for most values of $k >= 20\%$. We decided to choose $k = 50\%$ as the optimal value, as it delivers good results and also has an intuitive justification of optimising the top half of tokens towards the value of the document label, while the bottom half is optimised towards 0.

For the value $k = 50\%$, we now perform an extra step of finding the optimal hyperparameter $\gamma_{top/rest}$, which is the importance of the Top-k Rest-0 objective in the loss function. We explore $\gamma_{top/rest} \in [0.1, 1.0, 10, 100]$.

	FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
$\gamma_{top/rest}$	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
0.1	**87.50**	21.13	94.21	17.10	90.73	8.23	91.47	10.28
1.0	83.72	21.44	**95.04**	25.54	90.00	**11.50**	91.03	**15.02**
10	86.96	**23.73**	93.64	**30.35**	90.43	10.44	91.03	13.55
100	82.61	23.63	94.46	28.87	**91.61**	10.64	**92.11**	13.12

Table 6.5: Hyperparameter tuning for the token-level loss objective weight $\gamma_{top/rest}$ of the Top-k Rest-0 Soft Attention. We use $k = 50\%$, RoBERTa as a base Transformer and report results on the development dataset. We pick $\gamma_{top/rest} = 10.0$.

The results can be found in Table 6.5. We observe that results do not yield a clearly best performing value of $\gamma_{top/rest}$ and decide to pick $\gamma_{top/rest} = 10$ as the hyperparameter value we use, in line with values of γ for other soft attention architectures.

6.1.3 RoBERTa vs Longformer

In the final evaluation, we use Longformer for all soft attention architectures. In this section, we compare the performance of Longformer and RoBERTa on all datasets to explore how much benefit does using a model supporting longer texts brings. We use the hyperparameters we found in the previous sections for both RoBERTa and Longformer based models. The results are presented in Table 6.6.

	FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
Weighted Soft Attention RoBERTa	86.72	25.18	94.34	**16.85**	86.54	5.39	86.76	7.31
Weighted Soft Attention Longformer	**88.96**	**25.66**	**95.66**	16.30	**94.89**	**6.35**	**93.67**	**7.73**
Mean Soft Attention RoBERTa	**90.09**	27.02	86.29	**24.86**	87.86	10.91	90.74	15.13
Mean Soft Attention Longformer	87.86	**27.65**	**93.91**	22.64	**91.58**	**15.77**	91.80	**16.70**
Top-k Rest-0 Soft Attention RoBERTa	**88.86**	25.90	94.85	**29.85**	89.10	9.13	87.31	13.37
Top-k Rest-0 Soft Attention Longformer	88.33	**27.79**	**95.13**	27.32	**94.21**	**11.82**	**93.91**	**13.80**

Table 6.6: Comparison of performance of RoBERTa and Longformer based soft attention architectures on the development datasets. We find that apart from BEA 2019, which contains substantially shorter documents, Longformer-based models perform better on the token-level and often on the document-level. We use optimal hyperparameters for all architectures.

We note that on the token-level F_1, Longformer-based models perform better than RoBERTa-based ones. This is with the exception of BEA 2019, where RoBERTa-based models perform better on the token-level. We suspect it is likely due to the low percentage of long texts in this dataset. We also note substantial performance improvement on the document-level classification for Longformer-based systems on the IMDB datasets, which are the longest.

6.1.4 Compositional Soft Attention

Given the development dataset results presented above, we decided to assess the Compositional Soft Attention architecture in combination with the Top-k Rest-0 Soft Attention, with $\gamma_{top/rest} = 10$ and $k = 50\%$, following the previous experiments on RoBERTa.

FCE		BEA 2019		IMDB-Pos		IMDB-Neg	
Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1	Doc F_1	Tok F_1
68.18%	27.72%	90.33%	25.86%	82.89%	14.86%	87.16%	16.39%

Table 6.7: Development dataset performance of the Compositional Soft Attention system in combination with Top-k Rest-0 Soft Attention. We use $\gamma_{top/rest} = 10.0$ and $k = 50\%$, following the results for standard Top-k Rest-0 Soft Attention.

Table 6.7 contains the development dataset results of the Compositional Soft Attention that allowed us to verify that the system performs well. We find the performance on the document level to be substantially worse than other non-compositional soft attention architectures, but also notice the high token-level performance, making this method a good candidate for a rationale extractor. We suspect the performance of this model could benefit individual hyperparameter tuning, but leave this work to the follow-up research.

6.2 Final Experiments

For our final experiments, we report the average results for 3 different random seed values. We also report standard deviation for the document-level and token-level F-measure and the token-level MAP to quantify the variance. All the reported results are for the test dataset. For all soft attention models, we use Longformer as the base Transformer. Compositional Soft Attention uses a RoBERTa model.

	Doc F_1	Tok F_1	Tok P	Tok R	Tok MAP
Random Baseline	-	11.96 ± 0.28	14.67	10.10	15.86 ± 0.31
Document-level RoBERTa	$\mathbf{90.13} \pm 1.55$	-	-	-	-
Document-level Longformer	88.85 ± 1.69	-	-	-	-
FRESH Top-k (support model)	88.85 ± 1.69	7.28 ± 0.35	13.39	5.01	14.67 ± 0.02
Weighted Soft Attention	88.96 ± 0.68	25.66 ± 0.11	15.07	**88.63**	16.86 ± 0.30
Mean Soft Attention	87.86 ± 1.52	27.65 ± 2.26	17.82	61.72	18.98 ± 1.81
Top-k Rest-0 Soft Attention	88.33 ± 3.35	27.79 ± 0.56	20.81	42.30	18.25 ± 0.13
Compositional Soft Attention	88.04 ± 2.29	$\mathbf{31.07} \pm 0.76$	**24.82**	41.62	$\mathbf{21.93} \pm 0.14$

Table 6.8: Final results on the FCE dataset for the test split. Reported value is an average over 3 different random seeds, with standard deviation reported for key metrics. We find the Compositional Soft Attention to perform best on the token-level rationale extraction, while a RoBERTa document classifier performs best on the document-level. We suspect the good RoBERTa performance is caused by the dataset not containing enough long documents.

Grammatical Error Detection

We report the results on the Grammatical Error Detection datasets in Tables 6.8 and 6.9 for FCE and BEA 2019 respectively. On the document-level, we find both RoBERTa and Longformer to display similar performance. We suspect it is due to the lower proportion of long texts in these datasets, which allows standard Transformers to build robust document-level representations.

We find that the direct application of Weighted Soft Attention does not perform well for long documents on the token-level. We explore individual predicted token scores in Figure 6.2 and find that a majority of the token scores are high and close to 1, irrespective of the document-level prediction. We suspect that is likely due to the token-level loss function only optimising minimum and maximum values, preventing most tokens in the document from having their values optimised after the initialisation.

We note that Mean Soft Attention improves the token-level predictions significantly over the Weighted Soft Attention, with token-level F_1 absolute improvement of 1.99% on FCE and 6.34% on BEA 2019. We suspect that the mean loss

	Doc F_1	Tok F_1	Tok P	Tok R	Tok MAP
Random Baseline	-	11.07 ± 0.47	12.07	10.23	16.32 ± 0.42
Document-level RoBERTa	95.25 ± 1.12	-	-	-	-
Document-level Longformer	95.45 ± 0.48	-	-	-	-
FRESH Top-k (support model)	95.45 ± 0.48	13.85 ± 1.90	17.22	11.59	18.76 ± 0.27
Weighted Soft Attention	$\mathbf{95.66} \pm 1.18$	16.30 ± 0.85	9.05	**82.34**	22.88 ± 0.63
Mean Soft Attention	93.91 ± 3.59	22.64 ± 4.97	15.78	61.28	24.90 ± 0.33
Top-k Rest-0 Soft Attention	95.13 ± 1.47	$\mathbf{27.32} \pm 1.15$	**19.04**	48.46	22.45 ± 0.76
Compositional Soft Attention	88.06 ± 1.71	26.86 ± 1.39	17.75	55.62	$\mathbf{25.41} \pm 1.19$

Table 6.9: Final results on the BEA 2019 dataset for the test split. Reported value is an average over 3 different random seeds, with standard deviation reported for key metrics. We find the Top-k Rest-0 Soft Attention to perform best at the token-level classification, while Compositional Soft Attention is better at retrieving the ranks of tokens, as evidenced by the higher token-level MAP.

	W-SA	M-SA	TOP-SA	C-SA
To	0.99	0.81	0.00	0.22
finish	0.99	0.48	0.00	0.14
this	0.99	0.61	0.00	0.25
letter	0.00	0.98	0.00	0.00
with	0.96	0.02	0.00	0.04
I	0.99	0.24	0.00	0.71
would	0.99	0.00	0.00	0.00
like	0.99	0.00	0.00	0.02
to	0.73	0.88	0.00	0.30
give	0.99	0.96	0.00	0.06
you	0.94	0.84	0.00	0.06
a	0.00	0.99	0.00	0.00
good	0.99	0.97	0.00	0.00
idea	0.00	0.04	0.00	0.14
for	0.99	0.01	0.00	0.00
next	0.99	0.14	0.00	0.01
year	0.99	0.00	0.03	0.20
.	0.99	0.99	0.00	0.09

(a) FCE negative document (advanced learner), sentence with errors. Weighted Soft Attention assigns high token scores to most tokens despite supervision of the minimum and maximum towards 0 for negative documents. Mean Soft Attention reduces the number of tokens with high scores, but still assigns many high token scores despite the supervision to 0 for negative documents. Top-k Rest-0 Soft Attention correctly learns to assign scores of 0 to all tokens. Compositional Soft Attention learns to pick up incorrect tokens in the sentence, despite the document label being negative.

	W-SA	M-SA	TOP-SA	C-SA
So	0.15	0.98	0.34	0.29
,	0.99	0.02	0.02	0.00
I	0.99	0.99	0.00	0.00
hope	0.91	0.99	0.76	0.00
I	0.99	0.99	0.91	0.02
will	0.99	0.99	0.99	0.00
get	0.99	0.98	0.37	0.00
my	0.99	0.30	0.99	0.00
money	0.98	0.12	0.98	0.00
back	0.62	0.96	0.00	0.04
,	0.99	0.99	0.98	0.00
it	0.99	0.99	0.99	0.00
was	0.99	0.99	0.99	0.12
very	0.85	0.99	0.99	0.00
disappointing	0.99	0.99	0.99	0.00
evening	0.99	0.99	0.99	0.00
out	0.99	0.99	0.99	0.00
in	0.99	0.99	0.96	0.00
my	1.00	0.99	0.03	0.00
life	0.37	0.02	0.00	0.03
!	0.99	0.54	0.00	0.36

(b) FCE positive document (beginner learner), sentence with errors. Weighted Soft Attention assigns token scores close to 1 for most tokens. Mean Soft Attention gives high scores to less tokens than Weighted Soft Attention. Top-k Rest-0 correctly finds all incorrect tokens, while also giving high scores to the least number of tokens. For this sentence, Compositional Soft Attention does not find any incorrect tokens.

	W-SA	M-SA	TOP-SA	C-SA
In	0.99	0.93	0.00	0.70
the	0.99	0.99	0.00	0.72
end	0.11	0.99	0.01	0.51
,	0.91	0.93	0.00	0.40
the	0.99	0.56	0.00	0.52
restaurant	0.89	0.99	0.00	0.46
was	0.99	0.99	0.00	0.53
closed	0.97	0.99	0.00	0.14
because	0.99	0.57	0.01	0.27
you	0.75	0.99	0.18	0.28
did	0.04	0.99	0.00	0.26
n't	0.94	0.99	0.14	0.09
have	0.85	0.00	0.00	0.18
enough	0.27	0.43	0.00	0.28
staffs	0.99	0.99	0.00	0.27
,	0.15	0.99	0.00	0.56
this	0.99	0.99	0.00	0.39
evening	0.99	0.98	0.00	0.33
!	0.98	0.00	0.00	0.30

(c) FCE positive document (beginner learner), sentence with errors. Weighted Soft Attention assigns high token scores to most tokens, but fails to attend to ",". Mean Soft Attention similarly attends to most tokens, but detects ",". Top-k Rest-0 approach fails to detect any incorrect tokens. Compositional Soft Attention correctly detects one incorrect token, but does not attend to its neighbours.

	W-SA	M-SA	TOP-SA	C-SA
You	0.99	0.99	0.05	0.02
said	0.99	0.97	0.86	0.05
that	0.99	0.99	0.84	0.04
it	0.99	0.99	0.97	0.13
was	0.99	0.94	0.99	0.05
a	0.99	0.99	0.99	0.10
perfect	0.08	0.99	0.99	0.00
evening	0.66	0.95	0.49	0.05
out	0.99	0.76	0.00	0.18
but	0.99	0.99	0.13	0.00
it	0.58	0.99	0.98	0.17
was	0.97	0.99	0.00	0.57
n't	0.99	1.00	0.76	0.00
like	0.16	0.77	0.68	0.01
that	0.99	0.06	0.86	0.44
.	0.99	0.99	0.07	0.08

(d) FCE positive document (beginner learner), sentence without errors. Weighted Soft Attention and Mean Soft Attention both incorrectly assign high scores to most tokens in the sentence. Top-k Rest-0 similarly assign high scores to some tokens, despite the sentence containing no errors. Compositional Soft Attention does not assign high scores to any token but 1. Upon further inspection, it appears that the token selected by the Compositional Soft Attention was mislabeled as correct when it is in fact incorrect.

Figure 6.2: Example token-level predictions on the development dataset for Grammatical Error Prediction using the FCE dataset. *W-SA* represents Weighted Soft Attention Longformer, while *M-SA* indicates Mean Soft Attention Longformer. Top-k Rest-0 Longformer and Compositional Soft Attention are denoted *TOP-SA* and *C-SA* respectively. orange represents human annotations, red represents false positive token-level prediction and green represents a true positive one.

function increased the supervision signal available to tokens, leading to a better distribution of token scores. We notice that the system learnt to attend to less tokens and focus on the more important ones, as indicated by the decrease in recall and increase in precision. This is especially observed in the example presented in Figure 6.2(a), where Mean Soft Attention visibly attends to less tokens. However, we point out that Mean Soft Attention suffers from a high standard deviation on the token-level, which we believe is due to the extensive hyperparameter tuning required for this method to perform well.

We find the Top-k Rest-0 Soft Attention to perform similarly to the Mean Soft Attention, but also exhibit lower standard deviation on the token-level F_1, indicating that the method is more stable. As seen in Figure 6.2, while it misses out on some incorrect tokens, it is overall more precise and avoids attending to as many tokens as Weighted Soft Attention and Mean Soft Attention. This behaviour is also confirmed by the lower recall and higher precision on the token-level.

On the token-level, the Compositional Soft Attention architecture performs significantly better than other methods on the FCE dataset (3.28% absolute improvement). On the BEA 2019 dataset, the Compositional Soft Attention achieves substantially better results on the token-level MAP, but does not perform as well on the token-level F_1 score. This indicates that the model correctly learns the ranking of tokens, but struggles to correctly scale the token scores.

Upon the inspection of example token-level predictions in Figure 6.2, we note that while the Compositional Soft Attention learns to recognise some positive tokens, it doesn't correctly attend to neighbouring incorrect tokens (Figures 6.2(a), 6.2(c)). We point out that Compositional Soft Attention token scores of true positives seem lower than for other architectures. This further confirms our suspicion that the model fails to learn a correct distribution of the token scores, but successfully learns how to rank tokens well. As a way to mitigate it, we suggest a more appropriate choice of a loss function for the future research.

We note the significantly worse performance of the Compositional model on the task of document-level classification. We suspect this is due to the suboptimal hyperparameter choice, with too much weight assigned to the token-level loss objectives. In order to alleviate the issue in future research, we also suggest early stopping based on the document-level performance rather than the overall loss value.

Sentiment Analysis

We present the results for the Sentiment Analysis datasets IMDB-Pos and IMDB-Neg in Tables 6.10 and 6.11 respectively. We note that, unlike for Grammatical Error Detection, the Longformer model performs significantly better on the document-level than RoBERTa, with an absolute F_1 improvement of 5.52% and 6.46% on IMDB-Pos and IMDB-Neg respectively. We suspect it is due to the fact that IMDB datasets are longer in size and hence standard size Transformers do not have enough capacity to infer correct document labels based on only a part of the document.

	Doc F_1	Tok F_1	Tok P	Tok R	Tok MAP
Random Baseline	-	5.20 ± 0.09	3.54	9.80	8.44 ± 0.07
Document-level RoBERTa	88.62 ± 1.28	-	-	-	-
Document-level Longformer	94.14 ± 1.39	-	-	-	-
FRESH Top-k (support model)	94.14 ± 1.39	8.69 ± 3.62	6.70	12.38	7.83 ± 0.62
Weighted Soft Attention	**94.89** ± 0.31	6.35 ± 0.24	3.30	**84.36**	7.72 ± 0.29
Mean Soft Attention	91.58 ± 3.78	**15.77** ± 1.10	**9.15**	59.13	11.60 ± 1.36
Top-k Rest-0 Soft Attention	94.21 ± 1.68	11.82 ± 1.25	6.63	54.47	8.66 ± 0.72
Compositional Soft Attention	83.31 ± 1.96	12.73 ± 0.40	7.05	65.87	**14.81** ± 0.54

Table 6.10: Final results on the IMDB-Pos dataset for the test split. Reported value is an average over 3 different random seeds, with standard deviation reported for key metrics. We find that the Mean Soft Attention performs significantly better than Compositional Soft Attention on the token classification, despite Compositional Soft Attention ranking the tokens better. Compositional Soft Attention performs poorly on the document-level.

Similarly to the task of Grammatical Error Detection, a direct application of Weighted Soft Attention does not perform well. As seen in Figure 6.3, the Weighted Soft Attention models assign high scores to all tokens, irrespective of the predicted document label. This indicates yet again that there is not enough supervision signal available to tokens that would encourage a better scoring and ranking.

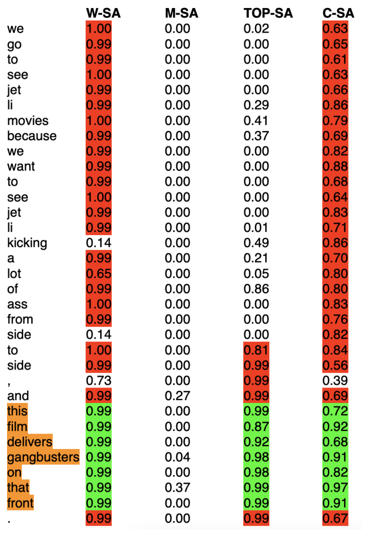
Mean Soft Attention achieves significantly better performance on the token-level F_1 score, with 9.42% and 8.97% absolute improvement for IMDB-Pos and IMDB-Neg respectively. We note that it also displays high standard deviation, indicating that the hyperparameters found might be overfit to a particular random seed.

We find the the Top-k Rest-0 Soft Attention performs significantly better on the token-level than the Weighted Soft Attention, but still substantially worse than the Mean Soft Attention architecture. We observe (Figure 6.3) that Top-k Rest-0 consistently predicts more false positives than the Mean Soft Attention method, indicating that the selected value of k% might be suboptimal.

We note that the Compositional Soft Attention performs significantly better than all architectures on the token-level MAP, with an absolute improvement of 3.21%

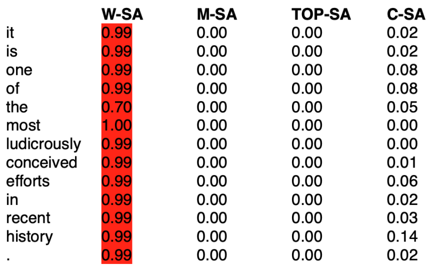
	Doc F_1	Tok F_1	Tok P	Tok R	Tok MAP
Random Baseline	-	5.89 ± 0.34	4.13	10.27	9.81 ± 0.06
Document-level RoBERTa	87.66 ± 0.87	-	-	-	-
Document-level Longformer	**94.12** ± 0.88	-	-	-	-
FRESH Top-k (support model)	**94.12** ± 0.88	8.73 ± 3.02	7.08	11.41	9.14 ± 0.46
Weighted Soft Attention	93.67 ± 0.55	7.73 ± 0.37	4.04	**90.41**	9.83 ± 0.91
Mean Soft Attention	91.80 ± 2.39	**16.70** ± 1.78	**10.56**	40.07	11.45 ± 0.73
Top-k Rest-0 Soft Attention	93.91 ± 1.64	13.80 ± 1.10	7.93	53.39	10.55 ± 0.78
Compositional Soft Attention	88.07 ± 2.22	16.30 ± 0.09	9.42	60.73	**16.29** ± 0.44

Table 6.11: Final results on the IMDB-Neg dataset for the test split. Reported value is an average over 3 different random seeds, with standard deviation reported for key metrics. We find that the Mean Soft Attention performs substantially better than Compositional Soft Attention on the token-level. Compositional Soft Attention performs poorly on the document-level.



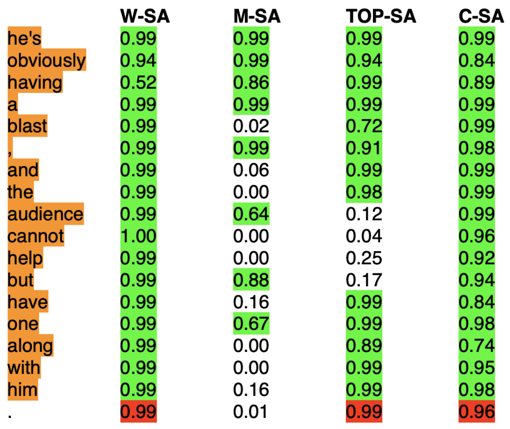
	W-SA	M-SA	TOP-SA	C-SA
we	1.00	0.00	0.02	0.63
go	0.99	0.00	0.00	0.65
to	0.99	0.00	0.00	0.61
see	1.00	0.00	0.00	0.63
jet	0.99	0.00	0.00	0.66
li	0.99	0.00	0.29	0.86
movies	1.00	0.00	0.41	0.79
because	0.99	0.00	0.37	0.69
we	0.99	0.00	0.00	0.82
want	0.99	0.00	0.00	0.88
to	0.99	0.00	0.00	0.68
see	1.00	0.00	0.00	0.64
jet	0.99	0.00	0.00	0.83
li	0.99	0.00	0.01	0.71
kicking	0.14	0.00	0.49	0.86
a	0.99	0.00	0.21	0.70
lot	0.65	0.00	0.05	0.80
of	0.99	0.00	0.86	0.80
ass	1.00	0.00	0.00	0.83
from	0.99	0.00	0.00	0.76
side	0.14	0.00	0.00	0.82
to	1.00	0.00	0.81	0.84
side	0.99	0.00	0.99	0.56
,	0.73	0.00	0.99	0.39
and	0.99	0.27	0.99	0.69
this	0.99	0.00	0.99	0.72
film	0.99	0.00	0.87	0.92
delivers	0.99	0.00	0.92	0.68
gangbusters	0.99	0.04	0.98	0.91
on	0.99	0.00	0.98	0.82
that	0.99	0.37	0.99	0.97
front	0.99	0.00	0.99	0.91
.	0.99	0.00	0.99	0.67

(a) An excerpt from a positive movie review in the IMDB-Pos dataset. We note that Weighted Soft Attention attends to most tokens, while Mean Soft Attention detects no positive sentiment evidence in this sentence. Top-k Rest-0 correctly identifies the evidence span. Compositional Soft Attention also correctly detects the evidence, but also incorrectly predicts most other tokens as evidence. Upon further inspection, we note that Compositional Soft Attention assigns higher scores to true positives than false positives, showing that the model learns to correctly rank tokens, but fails to scale the token scores. All models predicted this sentence as positive.



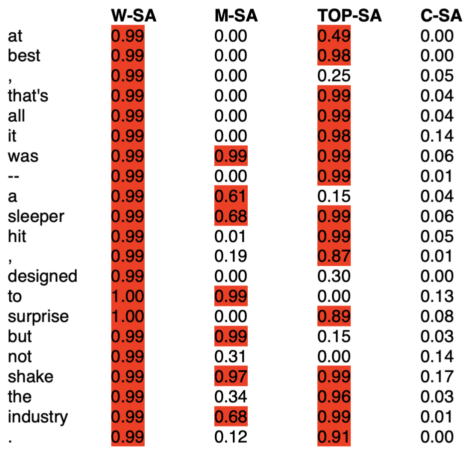
	W-SA	M-SA	TOP-SA	C-SA
it	0.99	0.00	0.00	0.02
is	0.99	0.00	0.00	0.02
one	0.99	0.00	0.00	0.08
of	0.99	0.00	0.00	0.08
the	0.70	0.00	0.00	0.05
most	1.00	0.00	0.00	0.00
ludicrously	0.99	0.00	0.00	0.00
conceived	0.99	0.00	0.00	0.01
efforts	0.99	0.00	0.00	0.06
in	0.99	0.00	0.00	0.02
recent	0.99	0.00	0.00	0.03
history	0.99	0.00	0.00	0.14
.	0.99	0.00	0.00	0.02

(b) An excerpt from a negative moview review in the IMDB-Pos dataset. As this is a negative review in a positive reviews dataset, no evidence is marked. Despite that, Weighted Soft Attention still assigns high scores to all tokens. All other methods correctly assign scores close to 0 for all tokens, not making a single false positive prediction.



	W-SA	M-SA	TOP-SA	C-SA
he's	0.99	0.99	0.99	0.99
obviously	0.94	0.99	0.94	0.84
having	0.52	0.86	0.99	0.89
a	0.99	0.99	0.99	0.99
blast	0.99	0.02	0.72	0.99
,	0.99	0.99	0.91	0.98
and	0.99	0.06	0.99	0.99
the	0.99	0.00	0.98	0.99
audience	0.99	0.64	0.12	0.99
cannot	1.00	0.00	0.04	0.96
help	0.99	0.00	0.25	0.92
but	0.99	0.88	0.17	0.94
have	0.99	0.16	0.99	0.84
one	0.99	0.67	0.99	0.98
along	0.99	0.00	0.89	0.74
with	0.99	0.00	0.99	0.95
him	0.99	0.16	0.99	0.98
.	0.99	0.01	0.99	0.96

(c) An excerpt from a positive movie review in the IMDB-Pos dataset with the whole sentence annotated as evidence. We notice that the Weighted Soft Attention attends to all tokens, unlike Mean Soft Attention or Top-k Rest-0 Soft Attention that only select a subset of tokens. We note that for this sentence Compositional Soft Attention assigned equally high weights to most tokens, indicating that it learnt that all tokens should have a high rank. All models predicted this sentence as positive.



	W-SA	M-SA	TOP-SA	C-SA
at	0.99	0.00	0.49	0.00
best	0.99	0.00	0.98	0.00
,	0.99	0.00	0.25	0.05
that's	0.99	0.00	0.99	0.04
all	0.99	0.00	0.99	0.04
it	0.99	0.00	0.98	0.14
was	0.99	0.99	0.99	0.06
--	0.99	0.00	0.99	0.01
a	0.99	0.61	0.15	0.04
sleeper	0.99	0.68	0.99	0.06
hit	0.99	0.01	0.99	0.05
,	0.99	0.19	0.87	0.01
designed	0.99	0.00	0.30	0.00
to	1.00	0.99	0.00	0.13
surprise	1.00	0.00	0.89	0.08
but	0.99	0.99	0.15	0.03
not	0.99	0.31	0.00	0.14
shake	0.99	0.97	0.99	0.17
the	0.99	0.34	0.96	0.03
industry	0.99	0.68	0.99	0.01
.	0.99	0.12	0.91	0.00

(d) An excerpt from a positive movie review in the IMDB-Pos dataset with no evidence. We note that Weighted Soft Attention incorrectly attends to all tokens. Mean Soft Attention and Top-k Rest-0 Soft Attention reduce the false positive rate of Weighted Soft Attention, with Mean Soft Attention attending to less tokens. Compositional Soft Attention correctly assigns low scores to all tokens. All models predicted this sentence as positive.

Figure 6.3: Example token-level predictions on the IMDB-Pos development dataset for Sentiment Analysis. *W-SA* represents Weighted Soft Attention Longformer, while *M-SA* indicates Mean Soft Attention Longformer. Top-k Rest-0 Longformer and Compositional Soft Attention are denoted *TOP-SA* and *C-SA* respectively. orange represents human annotations, red represents false positive token-level prediction and green represents a true positive one.

for IMDB-Pos and 4.84% for IMDB-Neg. It indicates that the model learns how to rank tokens better than other systems. However, the Compositional approach underperforms on the token-level F_1, suggesting that the model does not learn to scale the token scores. This can be observed in Figure 6.3(a), where we notice that the Compositional Soft Attention assigns higher scores to true positives than false positives. Yet, the false positives still have scores greater than the classification threshold of 0.50, leading to them being misclassified as positive. We found a similar issue with the token scores distribution for the Grammatical Error Detection datasets. We follow the previous suggestion for future work to include an additional loss function that distributes lower ranking tokens towards 0 and higher ranking ones towards the document label.

We report that the Compositional Soft Attention performs significantly worse on the document classification than Longformer-based systems. Upon the exploration of training logs, we found that is likely due to our early stopping criterion which considers the overall trend of the loss function. We experiment with document-level F_1 as the stopping criterion and found it to alleviate the issue. We did not have enough time to perform full experiments to report the results here.

6.3 Computational Efficiency

We report the average training time per epoch of each system in Table 6.12. The results reported are an average of 3 runs on the 16GB Nvidia Tesla T4. As expected, the RoBERTa-based systems take the least amount of time to complete a full epoch, with Longformer-based systems being $3-4\times$ slower.

We emphasise the significantly better performance of our Compositional Soft Attention models over the Longformer-based ones. On the FCE dataset, Compositional Soft Attention is approximately 40% faster, while being 65% faster for BEA 2019 and 30% for both IMDB-Pos and IMDB-Neg datasets. This represents a substantial improvement.

	FCE	BEA 2019	IMDB-Pos	IMDB-Neg
Document-level RoBERTa	99	158	172	171
Document-level Longformer	250	459	673	674
Top-k	264	488	701	701
Weighted Soft Attention	252	465	677	679
Mean Soft Attention	256	479	680	682
Top-k Rest-0 Soft Attention	258	482	685	682
Compositional Soft Attention	154	167	504	505

Table 6.12: Average time per epoch of each model during training on the full training dataset. The reported time is in seconds. All of the models were trained on the same 16GB Nvidia Tesla T4. We find that the Compositional Soft Attention is significantly faster than Longformer-based Soft Attention models.

Chapter 7

Conclusion

In this project, we have explored various architectures for long document binary classification and token-level rationale extraction using long document Transformers. We focused on the tasks of Grammatical Error Detection and Sentiment Analysis, while evaluating the plausibility of the extracted rationale against the token-level human annotations. To the best of our knowledge, this is the first work that quantitatively evaluates the token-level rationale extraction for long document Transformers.

We showed that a direct adaptation of Weighted Soft Attention, used before for Transformer-based zero-shot sequence labelling for sentence classifiers, does not perform well. We explored sample predictions of the model and found most token scores to be close to 1, regardless of the predicted document label. We suspect it is due to not enough supervision signal reaching individual tokens.

We proposed Mean Soft Attention, a modification of Weighted Soft Attention that optimises the mean token score towards the document-label. We found this architecture to significantly improve the token-level F_1 score compared to the Weighted Soft Attention, with 1.99% – 9.42% absolute improvement. Upon investigation, we noticed that the increased token-level supervision signal helped to avoid token scores obtaining similar scores irrespective of the document label. We noted the this method required extensive hyperparameter tuning and displays large standard deviation in the results.

We further introduced a Top-k Rest-0 Soft Attention architecture that supervises top k% of tokens towards the document label, with the rest of tokens supervised to 0. We report this approach to perform substantially better than Mean Soft Attention on the Grammatical Error Detection, while performing worse on the Sentiment Detection.

Finally, we proposed a novel Compositional Soft Attention architecture that does not require a long document Transformer to represent long texts. It uses a standard Transformer to compute token contextual embeddings of tokens in individual sentences, which are then composed into a document-level representation by a soft attention layer. We find the Compositional approach to be be 30% – 65% faster than Longformer-based models. We report substantially better results on the token-level MAP score (0.51% – 4.84% absolute) over Mean Soft Attention, indicating this model better learns how to best rank the individual tokens in the text. We find the token-level F_1 to be worse than for Mean Soft Attention. We note that while the model correctly recovers ranking of tokens, it fails to optimise some of the token scores to be below the classification threshold, leading to numerous false positives. We also report that the Compositional Soft Attention performs significantly worse on the document-level classification, which we suspect is due to suboptimal hyperparameters and an inefficient early stopping criterion.

7.1 Future Work

We note that this project is not fully completed yet. We intend to further investigate the poor performance of the Compositional Soft Attention on the document-level in order to find a solution that performs similarly to the long document Transformers. We also intend to explore a ranked loss function that would encourage high ranking tokens to be assigned substantially higher scores than lower ranking tokens. We hope this leads to improvement on the token-level F_1 score and allows the Compositional Soft Attention to become new state-of-the-art rationale extractor for long texts.

We intend to further evaluate the Compositional Soft Attention in combination with the full FRESH framework, evaluating how the extracted token-level rationales can act as faithful explanations to a document-classification model and improve the document-level performance.

We hope for this work to inform further research of explainable long document Transformers that provide plausible and faithful rationale.

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. pages 4, 12, 13

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019. pages 4, 12, 14, 15, 17, 28

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. pages 4, 12, 14, 36

[4] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China national conference on Chinese computational linguistics*, pages 194–206. Springer, 2019. pages 4, 15

[5] Samuel Bell, Helen Yannakoudakis, and Marek Rei. Context is key: Grammatical error detection with contextual word representations. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 103–115, 2019. pages 4

[6] 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, 2019. pages 4, 12, 20

[7] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32, 2019. pages 4

[8] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020. pages 4, 17, 19, 36

[9] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. pages 4, 17, 18, 28, 36

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. pages 4, 8

[11] Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018. pages 4

[12] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE, 2019. pages 4, 17

[13] Xingxing Zhang, Furu Wei, and Ming Zhou. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, 2019. pages 4, 17, 28

[14] Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, 2020. pages 4, 20

[15] Kamil Bujel, Helen Yannakoudakis, and Marek Rei. Zero-shot sequence labeling for transformer-based sentence classifiers. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 195–205, 2021. pages 4, 5, 7, 15, 16, 20, 23, 24, 37, 39

[16] Tian Shi, Xuchao Zhang, Ping Wang, and Chandan K Reddy. Corpus-level and concept-based explanations for interpretable document classification. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(3):1–17, 2021. pages 4

[17] Malte Feucht, Zhiliang Wu, Sophia Althammer, and Volker Tresp. Description-based label attention classifier for explainable icd-9 classification. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 62–66, 2021. pages 4

[18] Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 122–129, 2021. pages 4

[19] Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.409. URL https://aclanthology.org/2020.acl-main.409. pages 5, 20, 21, 27, 36

[20] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.408. URL https://aclanthology.org/2020.acl-main.408. pages 5, 20

[21] Marek Rei and Anders Søgaard. Zero-shot sequence labeling: Transferring knowledge from sentences to tokens. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302, 2018. pages 5, 7, 10, 11, 16, 19, 20, 21, 24, 37, 39

[22] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. pages 8

[23] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. pages 8, 9

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. pages 8

[25] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. pages 8

[26] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000. pages 8

[27] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010. pages 9

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. pages 9

[29] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015. pages 9

[30] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28:1693–1701, 2015. pages 9

[31] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005. pages 9

[32] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. pages 9, 10, 19

[33] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015. pages 10

[34] Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1070. URL https://aclanthology.org/P17-1070. pages 10

[35] Marek Rei. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2121–2130, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1194. URL https://aclanthology.org/P17-1194. pages 10

[36] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. pages 10

[37] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018. pages 12

[38] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. pages 12

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. pages 12

[40] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010. pages 13

[41] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. pages 13

[42] MV Koroteev. Bert: A review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*, 2021. pages 14

[43] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. pages 14

[44] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. pages 14

[45] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond

a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019. pages 17

[46] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. pages 17

[47] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. pages 18, 19, 22, 37

[48] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019. pages 19

[49] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. pages 19

[50] Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. Long-short transformer: Efficient transformers for language and vision. *Advances in Neural Information Processing Systems*, 34, 2021. pages 19

[51] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Łukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34, 2021. pages 19

[52] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021. pages 19

[53] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Generating token-level explanations for natural language inference. 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 963–969, 2019. pages 19

[54] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. pages 19

[55] Malika Aubakirova and Mohit Bansal. Interpreting neural networks to improve politeness comprehension. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2035–2041, 2016. pages 19

[56] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. pages 19

[57] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, 2018. pages 19

[58] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016. pages 19

[59] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. Investigating the influence of noise and distractors on the interpretation of neural networks. *arXiv preprint arXiv:1611.07270*, 2016. pages 20

[60] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *stat*, 1050:26, 2018. pages 20

[61] Ruiqi Zhong, Steven Shao, and Kathleen McKeown. Fine-grained sentiment analysis with faithful attention. *arXiv preprint arXiv:1908.06870*, 2019. pages 20

[62] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.432. URL https://aclanthology.org/2020.acl-main.432. pages 20

[63] Sarthak Jain and Byron C Wallace. Attention is not explanation. 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 3543–3556, 2019. pages 20

[64] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, 2019. pages 20

[65] Marina Fomicheva, Lucia Specia, and Nikolaos Aletras. Translation error detection as rationale extraction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4148–4159, 2022. pages 21

[66] Danish Pruthi, Bhuwan Dhingra, Graham Neubig, and Zachary C Lipton. Weakly-and semi-supervised evidence extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3965–3970, 2020. pages 21, 34, 35, 36

[67] Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. Feverous: Fact extraction and verification over unstructured and structured information. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. pages 21

[68] Karen Hao. Training a single ai model can emit as much carbon as five cars in their lifetimes. *MIT Technology Review*, 2019. pages 21, 28

[69] Alec Radford. Better Language Models and Their Implications. https://openai.com/blog/better-language-models/, 2019. [Online; accessed 27-January-2022]. pages 22

[70] General Data Protection Regulation. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016. pages 22

[71] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017. pages 22

[72] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. pages 25

[73] Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pages 872–881. PMLR, 2019. pages 32

[74] Helen Yannakoudakis, Øistein E Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31(3):251–267, 2018. pages 32

[75] Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4406. URL https://aclanthology.org/W19-4406. pages 32

[76] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 180–189, 2011. pages 33

[77] Marek Rei and Helen Yannakoudakis. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, 2016. pages 33, 35

[78] Omar Zaidan, Jason Eisner, and Christine Piatko. Using "annotator rationales" to improve machine learning for text categorization. In *Human language technologies 2007: The conference of the North American chapter of the association for computational linguistics; proceedings of the main conference*, pages 260–267, 2007. pages 34

[79] Tim Bray. The javascript object notation (json) data interchange format. Technical report, 2014. pages 35

[80] Guido vanRossum. Python reference manual. *Department of Computer Science [CS]*, (R 9525), 1995. pages 37

[81] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. pages 37

[82] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2020. pages 38

[83] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. pages 38

Appendix A

Hyperparameters

Model	Hyperparameter	Value
Shared		
	soft_attention_dropout	0.10
	soft_attention_evidence_size	100
	soft_attention_hidden_size	300
	optimiser	adamW
	lr	$1e-5$
	opt_eps	$1e-7$
	warmup_ratio	0.06
	dropout	0.10
	initializer_name	normal
RoBERTa		
	train_batch_size	8
	eval_batch_size	16
	gradient_accumulation_steps	4
Longformer		
	train_batch_size	1
	eval_batch_size	1
	gradient_accumulation_steps	32
	Compositional Soft Attention	
	compositional_document_batch_size	1
	gradient_accumulation_steps	32
	compositional_sentence_batch_size	8

Table A.1: Hyperparameters used during training.