DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Introducing a Hierarchical Attention Transformer for document embeddings

Utilizing state-of-the-art word embeddings to generate numerical representations of text documents for classification

VIKTOR KARLSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Introducing a Hierarchical Attention Transformer for document embeddings

VIKTOR KARLSSON

Master in Computer Science
Date: December 8, 2019
Supervisor: Hamid Reza Faragardi
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Swedish title: Introduktion av Hierarchical Attention Transformer för dokumentrepresentationer

Abstract

The field of Natural Language Processing has produced a plethora of algorithms for creating numerical representations of words or subsets thereof. These representations encode the semantics of each unit which for word level tasks enable immediate utilization. Document level tasks on the other hand require special treatment in order for fixed length representations to be generated from varying length documents.

We develop the *Hierarchical Attention Transformer* (HAT), a neural network model which utilizes the hierarchical nature of written text for creating document representations. The network rely entirely on attention which enables interpretability of its inferences and context to be attended from anywhere within the sequence.

We compare our proposed model to current state-of-the-art algorithms in three scenarios: Datasets of documents with an average length (1) less than three paragraphs, (2) greater than an entire page and (3) greater than an entire page with a limited amount of training documents. HAT outperforms its competition in case 1 and 2, reducing the relative error up to 33% and 32.5% for case 1 and 2 respectively. HAT becomes increasingly difficult to optimize in case 3 where it did not perform better than its competitors.

Sammanfattning

Inom fältet *Natural Language Processing* existerar det en uppsjö av algoritmer för att skapa numeriska representationer av ord eller mindre delar. Dessa representationer fångar de semantiska egenskaperna av orden som för problem på ordnivå direkt går att använda. Ett exempel på ett sådant problem är entitetsigenkänning. Problem på dokumentnivå kräver däremot speciella tillvägagångssätt för att möjliggöra skapandet av representationer med bestämd längd även när dokumentlängden varierar.

Detta examensarbete utvecklar algoritmen *Hierarchical Attention Transformer* (HAT), ett neuralt nätverk som tar vara på den hierarkiska strukturen hos dokument för att kombinera informationen på ordnivå till en representation på dokumentnivå. Nätverket är helt och hållet baserat på *uppmärksamhet* vilket möjliggör utnyttjandet av information från hela sekvensen samt förståelse av modellens slutsatser.

HAT jämförs mot de för tillfället bäst presterande dokumentklassificeringsalgoritmerna i tre scenarier: Datasamlingar av dokument med medellängden (1) kortare än tre paragrafer, (2) längre än en hel sida och (3) längre än en hel sida där antalet dokument för träning är begränsat. HAT presterar bättre än konkurrenterna i fall 1 och 2, där felet minskades med upp till 33% och 32.5% för fall 1 respektive 2. Optimeringen av HAT ökade i komplexitet för fall 3, för vilket resultatet inte slog konkurrenterna.

Contents

Chapter 1

Introduction

Written text has for centuries enabled humanity to accumulate knowledge across generations. It also made communication over both distance and time possible. It is reasonable to believe that keeping track of and categorizing these documents never posed a problem during the infancy of this technology. However, it is with the inception of the Internet together with democratization of speech no surprise that this medium has exploded in volume. Categorizing even a fraction of this data would today be an unreasonable task for humans. This has led to many new and interesting problems within the field of Machine Learning (ML) where automatic document categorization is one of them.

Enabling computers to make sense of the infinitely varying and complex structure of written language has been an ongoing research area for more than four decades [1]. Major breakthroughs have been achieved during the last few years [2, 3], enabling new use cases. This is partly due to the ever growing amount of available text data together with cross-pollination of ideas from different research fields within ML. Transfer learning techniques have been brought from the field of Computer Vision to Natural Language Processing (NLP). This has enabled word representation models of millions of parameters to be trained on datasets of billions of words [3] for use in new domains where data sparsity might otherwise prevent their success.

1.1 Background

An early approach for enabling computers to understand text dates back to the mid 1950s. Representing a document during this time was achieved with a list of its word-counts [1]. This technique, which is called a *Bag of words*-model, has some glaring issues but can still produce useful document representations for classification and clustering. The two main shortcomings are its total neglect of word ordering and similarity between words, both of which are invaluable for complete language understanding. These two issues can be solved by training a model to find numerical representation of words which encode their semantics. This approach result in word representations for similar words to be close in the high dimensional embedding space, thus preserving their semantic relatedness [2].

The growing interest in deep neural networks has enabled researchers to construct multi-million parameter models able to create even richer encodings for words. The performance of these models further close the gap to human performance in many different tasks, from question and answering to document classification [3].

A research area born from these achievements has focused on how to use this low-level information provided for each word to create meaningful representation of sentences, paragraphs or even documents. A lot of work has gone in to clever weighting schemes [4, 5, 6] and neural models, both linear [7] and recurrent structures [8], with the common goal of generating *document* embeddings. There are, to the best of our knowledge, still particularly interesting techniques yet to be examined which is what this thesis will explore.

1.2 Research question

This thesis will investigate a novel technique for creating document representations from pre-trained embeddings of the words within. We will to examine this will answer the following research questions:

- *Is there merit in using state-of-the-art word embeddings together with a neural network model only relying on attention to create document embeddings for use in a classification problem?*

- *How does the amount of training data in the case of longer documents affect the performance of our proposed model compared to current document embedding algorithms?*

The merit of our attention model can only be evaluated when compared with state-of-the-art document embedding algorithms. These will be described in later chapters. Further, we consider *longer* documents ones spanning more than 500 words. This usually is the amount that fits on a single A4-page, which closely coincides with the 512 token limit of Delvin et al.'s algorithm [3].

1.2.1 Delimitation

This thesis will only evaluate the proposed model together with the embedding algorithm presented in [3] even though it is possible to use other embedding algorithms. The performance is expected to vary with other models, but this will not be studied. Further, this work is limited to study the effect of varying the amount of training data for one particular dataset.

1.2.2 Relevancy and business value

The area of ML in general and NLP in particular have during the last couple of years experienced major breakthroughs. This has enabled computers to reduce the difference between algorithm- and human performance even further. These new developments have opened up many new areas of research which previously were out of reach. Our research questions are worth exploring since they fall into one of these categories. Further, the business value of automating tasks which previous required skilled knowledge workers should not be undermined. The tools these new algorithms can provide enable far more efficient use of these employees' time.

1.3 Research methodology

The research methodology this project follows is outlined below.

1. Identify and define the research goal through answering: *What is the problem we want to examine and find a solution to?* This is only possible through a literature review which will reveal what previously has been done. The research goal will serve as a northern star during the steps that follow.

2. Specify one or more research questions to restrict the research goal into a quantifiable research question. This should be done in conjunction with a study of the related works to ensure that a unique angle of the problem is addressed. The study of related works can also give new insights into what alternative research goals or question could be studied.

3. Evaluate and select the appropriate method through which the question(s) can be addressed. The experimental design need to take validity concerns into account to ensure that both internal and external validity are considered.

4. Present a solution which can answer the research question(s) with quantitative metrics. Implementing the algorithm might present technical challenges that require alterations to the initial solution. The process can from here fall back to step 3 and is therefore iterative in this regard.

5. The proposed solution is evaluated and the results are compared to the found benchmark references. This might result in new question presenting themselves which to be answered need new experiments to be defined. The process can from here be reiterated from step 2.

1.4 Ethics, sustainability and societal impact

With new technologies aimed at solving harmless issues, such as automatically categorizing and summarizing text, there will always be malicious actors using it to cause harm. The kind of damage the technologies described in this paper is capable of is far from physical, as would be the case for automated weapon systems. It can however be argued to have the ability to cause psychological harm.

The ability for machines to generate text of high enough quality to convince its reader it is written by another human is no longer Sci-Fi but rather reality [9]. It is, despite the fact that the researches of [9] have decided to keep their models out of the public hand, only a matter of time before malicious actors have created their own. One of the possibilities enabled through this technology is the automated spread of misinformation. This can have societal impacts far greater than what is covered in the generated articles themselves. Multiple conflicting sources makes it many times harder for the population to know what or whom to trust. In a society where trust is given to political candidates for representing ones beliefs and where the articles published in the media are assumed to be true, trust is everything. Without trust the entire system could collapse.

This thesis does not produce a generative model and will therefore not contribute to the problem described above. What is presented, on the other hand, is a discriminative model that can be trained to discern the class of a document. It is clear that such contribution has the ability to automate some information retrieval tasks, which could put the employment of a small set of knowledge workers at risk. However, it can be argued that people with these tasks are capable of spending their time on more valuable ones, such as fact- or reference checking. Our contribution should therefore not be regarded as their replacement but rather a tool they can utilize to empower their work.

Other benefits the technology this paper present and touch upon are automatic text summarization, question answering, document similarity assessment and improvements to information retrieval tasks. One example of such is semantic query expansion which leverage the word embeddings presented in this paper to provide richer search results. Tools like these, that enable society to more effectively leverage the shared knowledge, already provide great benefit and can only be assumed to continue to do so.

1.5 Outline

This work will be structured as follows. Chapter 2 will present the necessary ML knowledge together with the extensions of how to work with text data in this setting. This chapter is closed with a description of our contribution to the field of NLP. Chapter 3 is dedicated to contrasting our proposed algorithm to the already existing body of research within this field. Chapter 4 outline the methods which evaluate our proposed solutions in order for the research questions to be answered. This chapter also describes the technical details of the experimental procedure. The findings are shown in chapter 5 which are then discussed in chapter 6. The thesis is closed with concluding remarks and possible future extensions to this work in chapter 7.

Chapter 2

Background

Before diving into the details of state-of-the-art algorithms it is appropriate to give the reader an introduction to the foundation of these topics. This chapter will therefore provide a theoretical background of areas related to Machine Learning (ML) and Natural Language Processing (NLP).

The fundamental structure of neural networks will be presented first in section 2.1, which this is built upon in sections 2.3 and 2.4. These sections are key for understanding the benefits and architecture of the contribution of this thesis presented in section 2.7. Section 2.5 introduces the domain of NLP, which then is further expanded in section 2.6. These will introduce the reader to the techniques through with text data can be processed in a numerical fashion, which is the other key aspect of the contributions of this work.

2.1 Neural networks

Researchers within computer science have on the quest of enabling machines to learn taken inspiration from the inner workings of the human brain. The first breakthrough in this field was *the perceptron* introduced by Rosenblatt [10]. Neural networks have from this simple linear classifier evolved to the multi-million parameter, deep neural networks of today.

2.1.1 Architecture

The architecture of a neural network is something that often is depicted as a graph. The edges in these graphs are mathematical operations while nodes are either input-, intermediary- or output values, ordered into layers. This graph is therefore, in essence, a visual representation of a function constructed through

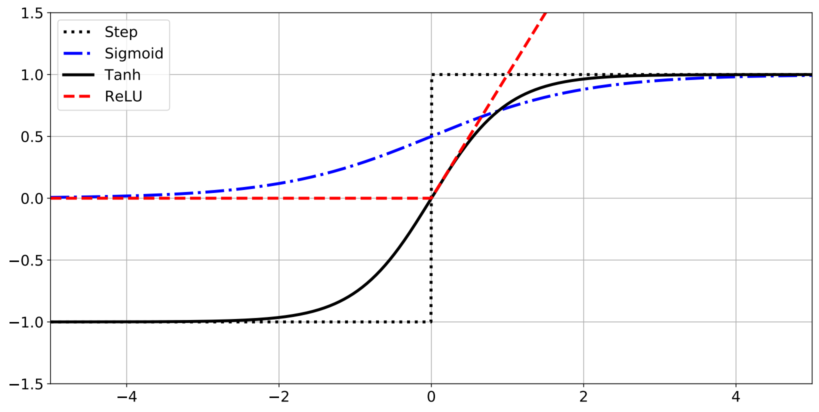


Figure 2.1: Illustration of activation functions.

repeated application of basic mathematical operations. An example of a two layered neural network is shown in equation (2.1).

$$f(\mathbf{x}) = f_2(\mathbf{W}_2 f_1(\mathbf{W}_1\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) \tag{2.1}$$

In this formulation is $\mathbf{x}$ the input and $\mathbf{W}_i$ and $\mathbf{b}_i$ parameters. The functions $f_i(\cdot)$ determine the behaviour and capabilities of the model. Some of the commonly used ones are shown in figure 2.1 with definitions listed below.

Step

$$\sigma(x) = \frac{x}{|x|} \tag{2.2}$$

Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

Tanh

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Rectified Linear Unit

$$\sigma(x) = \text{ReLU}(x) = \max(0, x) \tag{2.5}$$

Another activation function which often is used at the last layer of a model is the *softmax function*. The motivation behind using softmax at the last layer is as follows. Assume $\mathbf{y} = f(\mathbf{x}) = \sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}))$ is the vector of scores a model assigns to each of the y_i output classes. It would to perform classification be beneficial to transform these scores into probabilities. This requires that each element is non-negative and that their sum equals one. This is what the softmax function achieves, defined in equation (2.6).[1]

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.6}$$

2.2 Training techniques

The network example shown in equation (2.1) has weight matrices $\mathbf{W}_i$ and bias vectors $\mathbf{b}_i$. These have to be tuned for the function $f(\mathbf{x})$ to produce valuable inferences. This section will outline the main techniques for achieving such a model state by first describing how to manage the dataset of examples.

2.2.1 Dataset management

For a model to be able to perform a task it needs to be shown examples of connections between inputs and outputs. The goal is for it to learn to *infer* the output of previously unseen inputs. To simulate this scenario is it common practice to divide the dataset of inputs and outputs into three distinct sets; *training-*, *validation-* and *test dataset.* How these datasets are used and the reasoning behind creating them can be found below.

1. *Train* the model using the training set by showing it the input and output pairs. A wisely chosen model will be able to correlate the outputs to the underlying features characterizing each input. Problems that can occur if this is not the case, and how to avoid them, will be discussed later.

2. *Validate* the performance of the model on the validation set to simulate the scenario of performing inference on instances not observed during training. The results found here can be used as a guide to answer whether or not the training procedure is successful. They can also be used to compare performance between different models during the development phase.

3. When satisfactory results are achieved on the validation set can the model be *tested* on the test dataset. This set does most frequently only exist for

[1]In the binary classification setting will softmax reduce to sigmoid in equation (2.3).

benchmark datasets and are in those cases predefined. This guarantees that the evaluation of all models is fair.

2.2.2 Back-propagation

To *train a model* describe the process of updating its tunable parameters so that the loss on the training set decreases. The goal is for this to translate to a lower loss on the validation and test sets as well. The method through which this is achieved is called *back-propagation*. This is a process of calculating the gradients of the loss function with respect to each parameter in the model. These can then be used to update the parameters in such a way as to decrease the loss. There are multiple different ways of using these gradients when updating the weights, some of which are described below.

Gradient decent

The most rudimentary approach for updating the weights using the gradients is simply to change the parameters in the direction which result in the greatest decrease of the loss function. This update is given by

$$w \leftarrow w - \eta \frac{\partial L(w)}{\partial w} \tag{2.7}$$

where η is the *learning rate*. A too small value might result in slow convergence while a large one can result in divergent behaviour. The reason for this is that the gradient $\frac{\partial L(w)}{\partial w}$ only gives local direction of where to move the parameter values to decrease the loss. A too large step in the suggested direction might therefore break this assumption, resulting in increased loss.

Calculating the exact gradients of L using the entire dataset can be computationally expensive and is seldom necessary. Fortunately, it is possible to use a random subset of the training instances instead for the loss and gradient calculations to achieve an approximation of the true gradient. Each such random subset is called a mini-batch or simply a batch. This enables more updates to be calculated for the same computational cost, which often is more valuable to the optimization than that each step is perfect [11]. This optimization technique is called Stochastic Gradient Decent (SGD).

Adam optimizer

There are many additions that can be made to increase the performance of SGD. Notable are the additions of *momentum* and *learning rate adaptation*.

Momentum reduces the variance of the updates by keeping a fictional momentum between each update. Learning rate adaptation on the other hand address' the fact that each dimension of the loss function might have drastically different magnitudes. This makes it hard, or even impossible, to find a suitable learning rate since it is applied across all dimensions at once. Learning rate adaptation therefore scales each dimension so that these differences are reduced.

One of the more advanced optimization algorithms, which include both above mentioned additions, is Adam [2]. The update rules are described below [12].

$$m \leftarrow \beta_1 m + (1-\beta_1)\frac{\partial L(w)}{\partial w} \tag{2.8}$$

$$s \leftarrow \beta_2 s + (1-\beta_2)\left(\frac{\partial L(w)}{\partial w}\right)^2 \tag{2.9}$$

$$m \leftarrow \frac{m}{1-\beta_1^T} \tag{2.10}$$

$$s \leftarrow \frac{s}{1-\beta_2^T} \tag{2.11}$$

$$w \leftarrow w - \eta\frac{m}{\sqrt{s+\epsilon}} \tag{2.12}$$

Equation (2.8) introduces the above mentioned momentum while equation (2.9) provides the scaling factor. Both these term are then boosted for the early update steps in equations (2.10) and (2.11) to essentially jump-start the optimization. The weight updates are then applied in equation (2.12) where η again represents the learning rate. These equations allow for faster convergence when compared to other iterative optimization algorithms [12].

2.2.3 Techniques for improving model performance

It can become increasingly difficult to train a neural network model to perform their intended tasks as its number of parameters grows. Issues that can arise are, just to mention a few; vanishing gradients, exploding gradients, getting stuck in local minima of the loss function, overfitting, underfitting and/or slow convergence. This has essentially strongarmed researchers into developing techniques that can be applied to make the training procedure more efficient. The techniques which later will be used in this work are presented here.

[2]The name *Adam* is derived from *adaptive momentum estimator* and is therefore not an abbreviation.

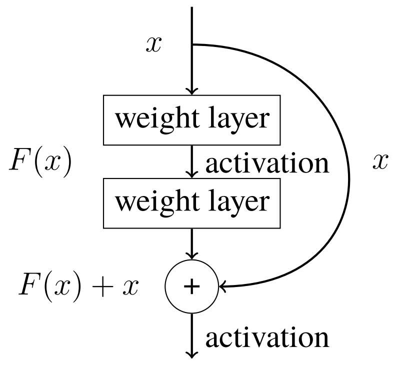


Figure 2.2: Residual learning building block [13].

Residual connections

The idea behind residual connection originates from observing that the performance of neural network models increase when more layers are added until this trend suddenly reverses and the performance declines. This is counter intuitive since the smaller models theoretically are *contained* within the larger network. The solution would be for all additional layers in the larger model to perform the *identity mapping*. He et al. therefore conclude that this mapping is difficult for the model to learn [13].

He et al. introduce *residual connections* in their network as a way to aid the network to learn the identity mapping. The residual connection creates a skip-connection which bypasses two or more feed forward layers. This connection is combined with the output from the skipped layers through addition which allows the network to learn the *residual* between input and output. Because of this reason can the network more easily learn the identity mapping. See figure 2.2 for a graphical depiction of this connection.

He et al. were with this technique able to construct many times deeper networks than previously possible. This also resulted in significant improvements to, as of publishing their paper, state-of-the-art performance in image related tasks [13]. Another interesting effect it had was that the loss surface for a model with these residual connections appeared much smoother compared to the same network without them [14]. This implied that it would be easier to find and converge to a good minima.

Dropout

With more complex network structures does the risk of *overfitting* these models to the training data increase. The problem was previously solved by aggregating inferences from multiple trained models to then average their results, a bootstrapping technique called *bagging*. This is however an unreasonable and wasteful method when models can take hours or even days to train.

The solution is to use *dropout* which randomly removes a portion of the connections in the network during each batch of training. This results in an exponential number of slightly thinned networks to be trained simultaneously, which achieve a similar effect to what bagging networks would [15].

Layer normalization

Multi-layer neural network models suffer from a problem of internal covariate shift occur during training. The reason for this is the fact that the input to each layer depends on the output of the previous one. If the weights of a layer would change, then the output from this layer would also change. This then requires each subsequent layer to adapt to a new *distribution* of inputs at each training step which increases the training complexity [16].

A solution is to apply *normalization* of the activations after each layer to keep the input distribution approximately normal throughout the training [17]. One therefore calculates the mean and standard deviation of the activations in a layer according to equations (2.13) and (2.14). These values would then be used to normalize the layer's output.

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \tag{2.13}$$

$$\sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H} \left(a_i^l - \mu^l\right)^2} \tag{2.14}$$

2.3 Recurrent neural networks

The type of neural networks that have been described above, often referred to as *feed-forward neural networks* due to the information being fed forward through its layers, are designed for processing static data. There are however many cases where a *sequential* aspect of the data is one of its key characteristics. Examples of such are stock-market data, weather data or written text. Each piece of information for this kind of data can be related to what has come before it and possibly also after, in a causal way. Processing this kind of data requires a model designed with memory so that it can store information from what previously has been observed. This is the the key characteristic of a *Recurrent Neural Network* (RNN).

A common feature of RNNs will here be highlighted to frame the reason for the development of the Transformer which is described in section 2.4. RNN models have a *hidden state* $\mathbf{h}_t$ which acts as its memory of the inputs $\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}...$ previously observed. This hidden state is updated when new inputs are observed though

$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \mathbf{x}_{t+1})$$

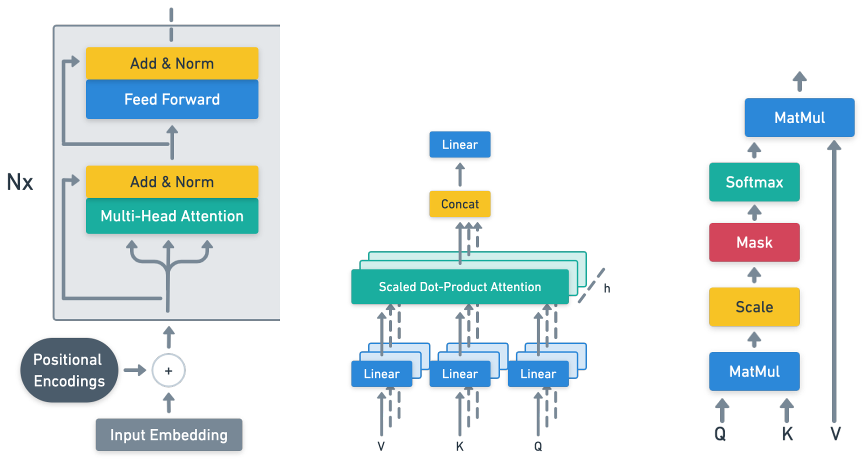
where there are many different ways of designing the function $f(\cdot)$. Two well known ways of defining this function can be found within the *Long Short-Term Memory* (LSTM) and *Gated Recurrent Unit* (GRU) networks. While the details of the function definition is not important for this work, the implied drawbacks are. It is difficult to train these memory units to update their internal representation so that it can connect information separated by many time-steps. The further the separation the harder this becomes due to the issue of *vanishing gradients* [18]. Further, to use information from both previously observed inputs and future inputs is not possible because of the models' sequential nature. This would however be beneficial for text related tasks [3]. A possible workaround for RNNs is to simultaneously train two models, one observing the sequence left-to-right and another right-to-left [19]. Such a shallow *bi-directional* model is believed to be strictly less powerful than a deep bidirectional one [3].

2.4 Transformer

Vaswani et al. recognize the shortcomings of RNNs and their poor utilization of the parallelization capabilities of modern computer hardware [20]. The authors' solution to these problems was a network named the Transformer which entirely relies on *attention* mechanisms for drawing the global dependencies between input and outputs. What attention is and how it integrates into the Transformer is explained below.

2.4.1 Overview

The Transformer consists of an encoder- and a decoder stack. The purpose of this architecture is to enable sequence-to-sequence modelling which for translation problems is required [20]. The encoder would for such problem



(a) Representational overview of a Transformer encoder block.

(b) Multi-head attention architecture of the Transformer encoder.

(c) Scaled dot-product attention.

Figure 2.3: A schematic overview of the different parts within the Transformer encoder, adapted from [20].

first processes the input to create a sequence of hidden states that the decoder then could use to generate the translated sequence. Both encoder- and decoder blocks are similar in construction which allow for this description to be limited to the encoder. Further, the decoder stack is of less interest for this work.

2.4.2 Encoder

Please study figure 2.3a before continuing to aid the understanding of the explanation that follows. Assume vector representations for tokens are given, see section 2.5 for how these are generated, and stored as rows of matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. The first step in *encoding* $\mathbf{X}$ is to add position dependent encodings to these representations. The reason for this addition is to enrich the token representations with the positional information each token has. This is for recurrent models implicitly implied through the order which tokens are fed into the model but has to be explicitly added for the Transformer. The positional encodings are defined by equation (2.15) where i indicates the token position and $j \in [1, d]$ represents the dimension of the feature vector.

$$p_{ij} = \begin{cases} \sin\left(\frac{i}{10000^{j/d}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{(j-1)/d}}\right) & \text{if } j \text{ is odd} \end{cases} \tag{2.15}$$

The reason for this choice of positional encoding is motivated by (1) it is hypothesized that they enable the model to learn to attend by the relative position of two tokens and (2) may allow the model to extrapolate to longer sequences than seen during training since the encoding at any fixed offset k can be represented as a linear function of observed encoding [20]. The positional information is added to each row $\mathbf{x}_i, i \in (1, 2, ..., n)$ of $\mathbf{X}$ through equation (2.16)

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{p}_i \tag{2.16}$$

The positionally enriched embeddings $\tilde{\mathbf{X}}$ are fed forward into the *transformer encoder block* where they enter the *multi-head attention* section, see figure 2.3b. A *head* in this context refers to one set of *scaled dot-product attention* calculation, see figure 2.3c. Let us focus our attention to one of these heads which will be denoted with a subscript i.

Attention

Attention is a mechanism that enables the network to attend information from anywhere in the sequence of inputs at each position. This should be contrasted to how RNNs process sequences, which is limited to only attend previously processed tokens. An illustrating example of this is how to create a hidden representation for the word *it* in the following sentence: *the animal did not cross the road because **it** was too tired.* With attention the network can more easily learn that *it* in this situation refers to *animal* and not *road* even though road is closer than animal in the sequence. Vaswani et al. enable this capability through their attention heads, which with three matrices, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V \in \mathbb{R}^{d \times p}$, calculate *queries*, *keys* and *values* through matrix multiplication with $\tilde{\mathbf{X}}$.

$$\begin{aligned} \mathbf{Q}_i &= \tilde{\mathbf{X}}\mathbf{W}_i^Q \\ \mathbf{K}_i &= \tilde{\mathbf{X}}\mathbf{W}_i^K \\ \mathbf{V}_i &= \tilde{\mathbf{X}}\mathbf{W}_i^V \end{aligned} \tag{2.17}$$

Matrices $\mathbf{Q}_i$ and $\mathbf{K}_i$ are used to measure the *compatibility* between tokens in the sequence through $\mathbf{Q}_i\mathbf{K}_i^\top$. This assigns a *score* to each query-key pair through a scaled softmax operation which describe the relative importance of each token to every other one. These scores are used to weigh how much each tokens' value vector in $\mathbf{V}_i$ should be fed forward from each head at each position in the sequence, which is then stored in $\mathbf{Z}_i$. These calculations are summarized in equation (2.18).

$$\mathbf{Z}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{p}}\right)\mathbf{V}_i \tag{2.18}$$

There are as mentioned earlier multiple attention heads, eight to be exact, where at each the above calculations in equation (2.17) and (2.18) are performed, resulting in eight $\mathbf{Z}_i$ matrices [20]. These are condensed into one matrix $\mathbf{Z}$ through the multiplication with $\mathbf{W}^O \in \mathbb{R}^{8p\times d}$.

$$\mathbf{Z} = [\mathbf{Z}_1, ..., \mathbf{Z}_8]\mathbf{W}^O \tag{2.19}$$

Before propagating $\mathbf{Z}$ to the *feed forward* network within the encoder block, Vaswani et al. applied a residual connection with $\mathbf{X}$ and then performed layer normalization as in equation (2.20).

$$\tilde{\mathbf{Z}} = \text{LayerNorm}(\mathbf{X} + \mathbf{Z}) \tag{2.20}$$

Position-wise Feed-Forward Network Each row $\tilde{z}_i, i = 1, ..., n$ of $\tilde{\mathbf{Z}}$ is fed through a two-layer fully connected feed-forward network using the ReLU activation function where these activations are again combined with a residual connection as well as a layer normalization, all described position-wise in equation (2.21).

$$\begin{aligned} \text{FFN}(\tilde{\mathbf{z}}_i) &= \max(0, \tilde{\mathbf{z}}_i\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \\ \mathbf{H} &= \text{LayerNorm}(\tilde{\mathbf{Z}} + [\text{FFN}(\tilde{\mathbf{z}}_1), ..., \text{FFN}(\tilde{\mathbf{z}}_n)]) \end{aligned} \tag{2.21}$$

$\mathbf{H}$ is the output from the first encoder-block which is fed to the preceding one, where equations (2.17)-(2.21) are repeated with $\mathbf{H}$ in place of $\mathbf{X}$. Stacking these attention layers enables richer representations to be generated.

2.5 Natural Language Processing

The common aspect of all ML models is the fact that they rely on applying mathematical computations to the input in order to learn. Because of this reason does much of the work within the field of Natural Language Processing (NLP) revolve around how to turn text into numerical features. We will in this section describe the fundamentals of how this transformation, from text into numbers has evolved.

2.5.1 Bag of words

The most rudimentary approach of turning text into numerical vectors is to count each unique word in the given text and use it as a feature. This idea was introduced by Harris et al. [21] and, since it shuffles word ordering, is called *Bag Of Words* (BOW). See the example below for an illustration of how a dataset of two documents is transformed into numerical vectors.

$$\begin{aligned} \mathcal{D} &= \{(\text{The movie was not that bad}), (\text{I liked that movie})\} \\ \mathcal{V} &= [\text{the, that, movie, was, not, bad, liked, i}] \\ BOW(\mathcal{D}_1) = \mathbf{x}_1 &= [1, 1, 1, 1, 1, 1, 0, 0] \\ BOW(\mathcal{D}_2) = \mathbf{x}_2 &= [0, 1, 1, 0, 0, 0, 1, 1] \end{aligned}$$

Here $\mathcal{D}$ represents a dataset of two sentences and $\mathcal{V}$ the vocabulary of all unique words found in $\mathcal{D}$. The last two rows above are the BOW representations for the documents in the dataset.

There are a few drawbacks to BOW models which limit their usability caused by their simplicity. First of all, the word order is lost. This means that two sentences that imply the opposite of each other can be assigned the same vector, as long as they use the same set of words [3]. Secondly, BOW-vectors of different words lack any form of semantic relatedness. This stem from the fact that all word vectors are orthogonal to each other in the embedding space. To be able to better understand a language, knowing which words have similar meaning and their order should not be underestimated.

Despite these shortcomings are BOW models able to perform unexpectedly well in some cases. This is especially true when more advanced weighting schemes than simple raw counts are applied to the word-features. One such scheme is described below.

2.5.2 Tf-idf

Intuition tells us that if a word occurs in every document in a corpus, it is probably not important for defining the unique characteristics of a particular document. It is in contrast reasonable to believe that a word which occurs frequently in a few documents, but not in others, are of importance for describing their characteristics.

[3] An illustrating example is *"The movie was not good, it was bad"* and *"The movie was not bad, it was good"* which would be assigned the same BOW vector

A method applying such a weighting scheme, which decrease the value of uninformative words while boosting words that are thought to be important, is *term frequency, inverse document frequency (tf-idf)* [4]. Instead of assigning the corresponding entry in the feature vector for each word with its raw count for each document D_i, we assign it its *tf-idf* score. The score is a function of the word w_j, the document D_i and the entire dataset D, see equation (2.22).

$$\text{tfidf}(w_j, D_i, D) = tf(w_j, D_i) * idf(w_j, D) \tag{2.22}$$

There are multiple ways to define each of the two functions $tf(\cdot)$ and $idf(\cdot)$. This work only use the following definitions:

$$tf(w_j, D_i) = \text{ \# of occurances of word } w_i \text{ in document } D_i \tag{2.23}$$

$$idf(w_j, D) = \frac{1}{\text{\# of documents in dataset } D \text{ word } w_j \text{ occur in.}} \tag{2.24}$$

2.5.3 Language modelling

The step beyond sparse representations is achieved through *language modelling*. Language modelling is the process of estimating the probability of linguistic units such as words, sentences or contiguous subsets thereof in the context they appear. This enables models to be trained to create *token representations* which capture similarity and relatedness between other tokens. Tokens are the smallest unit of text a model processes. This can be characters, n contiguous characters, entire words or other more elaborate ways of splitting text into pieces.

Sparse representations

The initial step in language modelling was the ability to estimate the probability of a certain sequence of words. Chen et al. and Kneser et al. applied count based approaches to this problem [22, 23]. The main idea was to factorize the probability estimation of the entire sequence into predicting the probability of one word at a time, given the *context* of previous words in the sequence. See equation (2.25).

$$P(w_1, w_2, ..., w_m) = P(w_1) \prod_{i=2}^{m} P(w_i | w_{i-1}, w_{i-2}, ..., w_1) \tag{2.25}$$

It is often a fair assumption to only need the n most resent words as context to efficiently predict the succeeding one. This is especially true when the text stretches over multiple sentences or even paragraphs. This is called the *n-gram* constraint. Using this limitation enables equation (2.25) to be rewritten into its approximative, n-gram counterpart shown in equation (2.26).

$$P(w_1, w_2, ..., w_m) \approx P(w_1) \prod_{i=2}^{m} P(w_i | w_{i-1}, w_{i-2}, ..., w_{i-n}) \tag{2.26}$$

where both [22] and [23] used count based techniques to assign each sequence of words the probability given in equation (2.27).

$$P(w_i | w_{i-1}, w_{i-2}, ..., w_{i-n}) = \frac{count(w_i, w_{i-1}, w_{i-2}, ..., w_{i-n})}{\sum_{w \in \mathcal{V}} count(w, w_{i-1}, w_{i-2}, ..., w_{i-n+1})} \tag{2.27}$$

The probability of each sequence of words is thus estimated by their relative frequency of word w_i preceded by the current context. This results in zero probability being assigned to any sequence not observed in the training corpus during inference time. A solution for this problem using this strategy does not exist, even though both [22] and [23] improve on this limitation.

Distributed representations

Bengio et. al. [24] took note of this issue and that of sparsity for the word vectors and provided a solution to address them both. The proposed solution was to learn *distributed representations* for each word in the vocabulary, which was the first contribution of its kind. Benigo et. al. formulates the model as a simple neural network with an embedding layer $\mathcal{C}$. This maps each word w_i to a distributed feature vector $\mathcal{C}(w_i) \in \mathbb{R}^m$. These vectors would then be used as input to a simple neural model defined in equation (2.28).

$$\begin{aligned} P(w_i| \, w_{i-1}, w_{i-2}, ..., w_1) \approx & P(w_i| \, w_{i-1}, w_{i-2}, ..., w_{i-n}) = \\ & = \text{softmax}(\mathbf{b} + \mathbf{W}\mathbf{x} + \mathbf{U} \tanh(\mathbf{d} + \mathbf{H}\mathbf{x})) \end{aligned} \tag{2.28}$$

Here $\mathbf{x} = [\mathcal{C}(w_{i-1}), \mathcal{C}(w_{i-2}), ..., \mathcal{C}(w_{i-n})]$ is the concatenation of all feature vectors for the context words and $\mathbf{b}, \mathbf{d}, \mathbf{W}, \mathbf{U}, \mathbf{H}$ are parameters of the network.

Despite the impressive results compared to previous attempts at creating robust language models [24] does the above method lack a certain characteristic. The semantics and syntactic relations among words are lost at the embedding layer. The learnt representations cannot guarantee that, for example, $\mathcal{C}(paris)$ is any *closer*[4] to $\mathcal{C}(france)$ than it is to $\mathcal{C}(quantum\ physics)$.

A model aimed at capturing this characteristic, preserving the word semantics during embedding, was presented by Mikolov et al. [2]. The authors also highlighted another key finding they addressed; simple models perform excellent when there is lots of available data, but l ose their advantage to more complex models when data is sparse. The two models Mikolov et al. presented are described below as well as shown in figure 2.4.

Continuous bag-of-words The continuous bag-of-word model (CBOW) is a word embedding layer trained to predict the centre word given its surrounding context. The model only relies on the average of the context word embeddings [2]. Because the word embeddings are *continuous* vectors and that their positional information is lost during the averaging operation, just as for the BOW algorithms, this method is given its name.

Skip-gram The skip-gram model consist of a word embedding layer trained to predict the surrounding words given the centre one (the reverse to CBOW). The more words this model is asked to predict both before and after the centre word the better embeddings become [2]. Mikolov et al. did however need to balance this with the increased computational complexity.

These training schemes gave words occurring in similar contexts similar word embeddings. In fact, this characteristic was what Mikolov et al. used for measuring the quality of their embeddings. An interesting implication of these embeddings was the ability to perform algebraic operations with the learnt vector representations to find, for example, the word that for *small* has the same relation that *biggest* has to *big*[5]. The answer to such question could be found through $\mathbf{x} = vec('biggest') - vec('big') + vec('small')$, where the closest word-vector to $\mathbf{x}$ would be the best answer.

These models, CBOW and Skip-gram, can be pre-trained on a task-agnostic dataset such as news articles from Google to be used as an embedding layer. This layer transform word into dense vector representations and has since been referred to as Word2Vec.

[4]Closer in terms of distance measured using a similarity measure as *cosine similarity*

[5]The answer is *smallest*

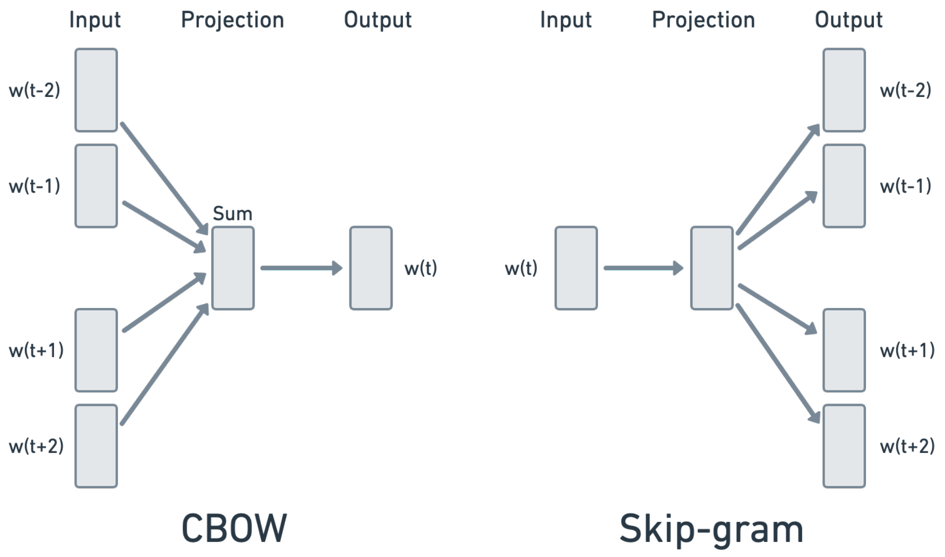


Figure 2.4: Illustration of the two models presented in [2]. Figure adapted from [2] .

GloVe Yet another advancement within dense word representations was the ideas of combining global information given by the word co-occurrence statistic with the local information of the context words [25]. Pennington et al. formulated this as a least square regression problem with the loss to be minimized defined in equation (2.29).

$$L = \sum_{i,j=1}^{V} f(X_{ij}) \left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}) \right)^2 \tag{2.29}$$

$f(\cdot)$ is a weighting function to reduce the impact of rare co-occurrences, X_{ij} the matrix with entries corresponding to the number of times words i and j occur in the same context, w_i the word vector and $\tilde{w}_j$ a context vector. b_i and $\tilde{b}_j$ are bias terms. Minimizing this loss result in predefined embeddings for all words in the vocabulary.

The main benefit of the three models described above is the fact that they can be trained *once* on large datasets to capture the semantics of that language in the learnt word embeddings. These can then be used for other NLP tasks, where data might be limited. These contribution has since their invention set the standard for how to convert text into numerical features and has therefore played an irreplaceable role for advancing the developments within NLP.

2.6 Contextualized word embeddings

Despite the wide adoption of distributed word embeddings [2, 25], one key characteristic of these still troubled researchers: a word can have different meanings depending on its context but is still assigned the same vector representation. An example highlighting this issue is what *bank* refers to in the sentences *the boat crossed the river and arrived at the other **bank*** and *I deposited my money at the **bank***. It therefore seems to be important to consider the context the word occurs in at embedding time when creating its vectorization, rather than giving it a predefined one.

2.6.1 Context vectors

The first work that acknowledged this fact and provided an attempt at solving it can be found in [26]. McCann et al. commented on the recent advancements within the field of computer science accelerated by the use of *transfer learning* CNN's on ImageNet [27]. The authors also believed that the same should be possible within NLP through using translation as the pre-training task.

McCann et al. created a bi-directional Machine Translation LSTM model (MT-LSTM) to encode sequential input of GloVe based word-vectors to hidden representations. These were used as part of the input to a decoder, based on an unidirectional LSTM model, to produce the translated sentences. Transfer learning was introduced into the model by using the output from the MT-LSTM as *context vectors* to enrich the fixed word representations generated by GloVe. Each word was instead represented by concatenation of the two vectors. Given a sequence of words w_i in $\mathbf{w}$, then

$$\begin{aligned} \text{CoVe}(\mathbf{w}) &= \text{MT-LSTM}(\text{GloVe}(\mathbf{w})) \\ \tilde{w_i} &= [\text{GloVe}(w_i), \text{CoVe}(w_i)] \end{aligned} \tag{2.30}$$

The addition of context achieved state-of-the-art results compared to previous methods relying on either Word2Vec or GloVe [26]. It therefore became clear that context aware models were both possible and worthwhile pursuing.

2.6.2 Bidirectional Encoder Representations from Transformers

With the introduction of the Transformer (see section 2.4) a more capable architecture, not relying on a shallow concatenation of left-to-right and right-to-left models, was now possible [3]. Devlin et. al. introduced the Bidirectional

Encoder Representations from Transformers (BERT) which presented novel techniques for training a deep bidirectional language model. This model was, even without any task-specific architecture modification, able to achieve state-of-the-art results on eleven NLP benchmark tasks [3]. This includes all nine tasks of the General Language Understanding Evaluation (GLUE) benchmark [28], a Named Entity Recognition (NER) task posed by the CoNLL 2003 NER dataset and grounded common sense inference posed by the Situations With Adversarial Generations (SWAG) dataset [29].These results indicated that the embeddings BERT created and used for inference encode the content in a powerful way, lending itself to be used as an embedding architecture.

Architecture

Input BERT processes the text input with a tokenizer algorithm called WordPiece. It splits words into a limited number of common word units. Such method provides a balance between the efficiency of word-level models and the flexibility of character-level ones [30]. An example highlights its characteristics: the sentence *I am playing* is tokenized to [I, am, play, ##ing]. The ## characters are used to indicate the presence of a sub-word unit.

Layers The inner workings of BERT is effectively identical to the already existing OpenAI Transformer [31]. The architecture is a stack of Transformer encoders which has been described in section 2.4. Two models are presented, $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, with architectural parameters specified in table 2.1.

Because of the fact that an attention mechanism is present at each head in every layer of the model is each token considered as information to every other token during the forward pass. The number of key-query evaluations does therefore grow quadratically with the length of the input which makes it infeasible to allow for arbitrary length inputs. Devlin et al. did therefore limit the length of an input to 512 tokens. For the datasets used to benchmark BERT did this limitation seldom come into play since most documents fall well below this threshold.

Pre-training

Devlin et al. highlighted the fact that there, as of writing their paper, were no successful attempts at creating a *deep bi-directional* language model. The authors argued that such a model should be strictly more powerful than a shallow concatenation of a left-to-right and a right-to-left language model, which

Parameter	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{LARGE}}$
Transformer layers	12	24
Attention heads	12	16
Hidden dimension	768	1024
Total parameters	110M	340M

Table 2.1: Parameters for the two BERT-models presented in [3]

previous state-of-the-art algorithms were constructed as [26, 19]. The reason was that there had not existed an efficient training procedure for such model due to the problem of bi-directional models allowing the token to be predicted *see* itself. A workaround for this issue was one of the novel contributions in [3], which Devlin et al. named Masked Language Modelling.

Masked Language Modelling Language modelling, as described in section 2.5.3 consists of finding the conditional distribution for the next word given its left context as in equation (2.25) [24]. This does however limit the power of the model since there are many situations where right context can aid in predicting the current word. Devlin et al. introduced Masked Language Modelling (MLM) to be able train a model that can take both history and future words into account. MLM draws inspiration from the Cloze Task [32] and randomly selects 15% of the input tokens and applies one of the following action:

- 80% of the time the word is replaced with the `[MASK]` token,
- 10% of the time it is replaced with another random word from the vocabulary,
- 10% of the time the word is left unchanged.

The reasoning and purpose of these three kind of replacements was first to mitigate some of the mismatch in token distributions during pre-training and fine-tuneing. Since the `[MASK]` token will not be observed during the latter process is it helpful to bias the model towards normal word-tokens. Secondly, replacing a token with another random one, where the original word is then to be predicted, forces the model to keep a distribution over every token. Finally, leaving the correct word unchanged biases the model towards this token [3].

Next Sentence Prediction In addition to the above described MLM training did Devlin et al. also propose a next-sentence prediction task which the network is trained to perform. The reason is that MLM does not capture the relation between two adjacent sentences, which especially for the Question and Answering, and Natural Language Inference tasks are crucial [3]. When choosing the sentences A and B for pre-training, B is with 50% probability either the true next sentence or a random one sampled from the corpus.

The model is pre-trained with these two tasks on a concatenation of BooksCorpus [33] and English Wikipedia, which add up to 3.3 billion words in total [3]. With a model pre-trained in this way is it possible to fine-tune this model to perform other tasks where classification is an example of such.

Key findings

Devlin et al. presented results from using BERT as a feature extractor, which generated embeddings for each of the input tokens similarly to [25] and [2]. Using the embeddings from the four to last hidden layers (combined through summation) and fed through a two-layered BiLSTM model Devlin et al. were able to achieve 95.9 F1 on the CoNLL 2003 NER. This should be compared to the 96.4 F1 they achieve when fine-tuning the entire BERT architecture on the NER task [3]. This strengthens the hypothesis that the embeddings created by the pre-trained model are suitable for use as input to another model.

2.7 Our contribution

With the key concepts introduced in the previous sections is it now appropriate to describe the contributions of this thesis. We decided to include it here, at the end of the background chapter, in order for the content subsequent one *(Related works)* to be more easily understood.

The contribution of our work two-fold. First we describe how we utilized BERT for extracting embeddings for documents longer than the limit of 512 tokens. Secondly we present the, to the best of our knowledge, novel network architecture we propose for document classification.

2.7.1 Extracting features from BERT for documents

Our first contribution is a strategy for extracting token representations for documents longer than the limit of 512 tokens BERT enforces. The idea is not new and known by the authors [6], but the implementation is unique.

One of the hypothesised reasons for why BERT is able to create such rich token representations is because of its ability to attend information from anywhere in the sequence. The context that each token has is therefore key to its representation capability. We applied a sliding-window approach for deciding which tokens to be feed to BERT at each iteration in order to maximize the context for each token. Within this window we extract embeddings for half of the tokens and use the remaining tokens as context, chosen in such a way to always include as much right and left context as possible. For all our experiments we used a window size of 256 resulting in 128 embedded tokens in each iteration.

2.7.2 The Hierarchical Attention Transformer

This section describes our proposed architecture, which utilize embeddings generated by BERT to create document embeddings. The network will make use of the *hierarchical* nature of documents by combining word embeddings into sentence representations and from these construct the document vector. Much of the inspiration for this architecture stems from [8], but will be implemented without the need of recurrence. Instead, our proposed network only relies on *attention* through the Transformer. The network will for these reasons called the *Hierarchical Attention Transformer* (HAT).

Architecture

Given word embeddings $\mathbf{h}_{it} \in \mathbb{R}^d$ generated by BERT the first step of HAT is to combine these into sentence embeddings. Each sentence is generated by the words within, which is assumed to be bounded by the characters ".", "!" or "?". Let $[\mathbf{h}_{i1}, ..., \mathbf{h}_{iT_i}]$ represent all T_i word embeddings of the i:th sentence in a document. The sentence embedding $\mathbf{s}_i$ is created through an attention calculation shown in equation (2.31).

$$\begin{aligned}
\mathbf{u}_{it} &= \tanh(\mathbf{W}_w \mathbf{h}_{it} + \mathbf{b}_w) \\
\alpha_{it} &= \text{softmax}(\mathbf{u}_{it}^\top \mathbf{c}_w) \\
\mathbf{s}_i &= \sum_{t=1}^{T_i} \alpha_{it} \mathbf{h}_{it}
\end{aligned} \tag{2.31}$$

[6]A question asked regarding how to handle longer sequences in the official github repository, with the answer presented https://github.com/google-research/bert/issues/66#issuecomment-436378461.

where $\mathbf{W}_w \in \mathbb{R}^{u\times d}$, $\mathbf{b}_w \in \mathbb{R}^u$ and $\mathbf{c}_w \in \mathbb{R}^u$ are parameters. These equations project the word embeddings from $\mathbb{R}^d$ to $\mathbb{R}^u$ where $u > d$, measure the compatibility of this projection with a vector $\mathbf{c}_w$ through dot product and assign each word a softmax score based on the result. The sentence vector $\mathbf{s}_i$ is then calculated through a weighted sum using the scores from the previous step as the weight for each individual word. This calculation lends itself to insightful visualizations of which tokens the model learns to focus on or *attend to* in each sentence.

The positional encodings proposed by [20] are applied to each sentence in order for the Transformer to make use of their positional information. These positional embeddings are explained in detail in section 2.4 and shown in equation (2.32).

$$p_{ij} = \begin{cases} \sin\left(\frac{i}{10000^{j/d}}\right) & \text{if } j \text{ is even} \\ \cos\left(\frac{i}{10000^{(j-1)/d}}\right) & \text{if } j \text{ is odd} \end{cases} \tag{2.32}$$

The positional information is added to the sentence embeddings

$$\tilde{\mathbf{s}}_i = \mathbf{s}_i + \mathbf{p}_i \tag{2.33}$$

These vectors are then fed to the Transformer encoder as described in section 2.4. The embeddings are through this operation transformed into new embeddings for each sentence $\mathbf{s}_i^*$, which are used to construct the final document embedding. Here, the same attention calculation as performed to create the sentence embeddings from its words is applied to create the document embedding. See equation (2.34).

$$\begin{aligned} \mathbf{v}_i &= \tanh(\mathbf{W}_s\mathbf{s}_i^* + \mathbf{b}_s) \\ \beta_i &= \text{softmax}(\mathbf{v}_i^\top \mathbf{c}_s) \\ \mathbf{d} &= \sum_{t=1}^{T_i} \beta_i \mathbf{s}_i^* \end{aligned} \tag{2.34}$$

where $\mathbf{W}_s \in \mathbb{R}^{v\times d}$, $\mathbf{b}_s \in \mathbb{R}^v$ and $\mathbf{c}_s \in \mathbb{R}^v$ are the parameters to learn. $\mathbf{d}$ is the document embedding of fix length d which can be used as input for many ML classifiers.

Classification

A projection with softmax activations is attached to the output of the above described architecture to enable classification with the generated document embeddings. This layer is trained in conjunction with the entire HAT network for minimizing the cross-entropy loss.

Chapter 3

Related works

The proposed algorithm presented in section 2.7 is far from an idea in isolation. There are many different approaches in finding solutions to the problem of creating embeddings for documents. This chapter describe these approaches and contrast them to HAT in order for a more thorough analysis of either sides strengths and weaknesses.

Distributed word embeddings has since [24] been an integral part of many NLP tasks, where more recently [34] and [25] have been considered the standard starting point for any downstream tasks. An active research field has focused on how to utilize these low-level word embeddings to create high-level sentence, paragraph or even document embeddings. There are many different approaches to this problem which all should be compared to the new ideas presented in this work.

3.1 Smooth Inverse Frequency

Arora et al. introduced a language model partly driven by a discourse vector [35]. This discourse vector $\mathbf{c}_t \in \mathbb{R}^d$ was used to quantify the probability of a word $\mathbf{v}_i \in \mathbb{R}^d$ being observed in the current context. The authors also enable this discourse vector to change over time, through driving it with a random walk process. $\mathbf{c}_{t+1}$ would therefore be given by the previous discourse $\mathbf{c}_t$ and a small displacement vector, where t indicate the word position. Nearby words would then be generated by similar discourse, which is a fair assumption. This was modelled by a log-linear word production model given by equation (3.1).

$$p(w_i|c_t) \propto \exp(\mathbf{c}_t^\top \mathbf{v}_i) \tag{3.1}$$

This model learned word representations similarly to how Word2Vec and GloVe did, where co-occurring words were given similar vectors [5]. However, this model would still not produce representations for documents. Arora et al. tackled the problem through further building on the ideas presented in their previous work [5]. The new model replaced the time-dependent discourse $\mathbf{c}_t$ with a fix vector $\mathbf{c}_d$ representing the discourse throughout the sentence or document. Arora et al. argued that such modification was fair due to the small changes of $\mathbf{c}_t$ over smaller sections of text. Calculating this vector amounted to finding the maximum a poseteriori (MAP) estimate of a slightly modified equation (3.1). The reason being that the MAP estimate of $\mathbf{c}_d$ using equation (3.1) is proportional to the average over all word vectors in a section of text [5], which is undesirable.

The modification to (3.1) included two types of smoothing terms which, similar to the intuition behind tf-idf weighting, were meant to account for (1) words that occur out of context and (2) words that occur regardless of the context [5]. The weighting scheme used is shown in (3.2). The algorithm is because of these reasons referred to as Smooth Inverse Frequency (SIF).

$$\begin{gathered} p(w_i|\mathbf{c}_d) = \alpha p(w_i) + (1-\alpha)\frac{\exp(\tilde{\mathbf{c}}_d^\top \mathbf{v}_{w_i})}{Z_{\tilde{\mathbf{c}}_d}}, \\ \tilde{\mathbf{c}}_d = \beta \mathbf{c}_0 + (1-\beta)\mathbf{c}_d, \quad \mathbf{c}_0 \perp \mathbf{c}_d \end{gathered} \tag{3.2}$$

$p(w_i)$ is added to account for the probability of words appearing out of context and $\tilde{\mathbf{c}}_d$ is introduced to enable the common discourse vector $\mathbf{c}_0$ to be removed from the document specific vector $\mathbf{c}_d$. This accounts for the second type of smoothing. Further, α and β are scalar hyper-parameters and $Z_{\tilde{\mathbf{c}}_d} = \sum_{w \in \mathcal{V}} \exp(\tilde{\mathbf{c}}_d^\top \mathbf{v}_{w_i})$ the normalizing constant.

The likelihood for observing the document d is now given by equation (3.3).

$$p(d|\mathbf{c}_d) = \prod_{w_i \in d} p(w_i|\mathbf{c}_d) = \prod_{w_i \in d} \left[\alpha p(w_i) + (1-\alpha)\frac{\exp(\tilde{\mathbf{c}}_d^\top \mathbf{v}_{w_i})}{Z_{\tilde{\mathbf{c}}_d}}\right] \tag{3.3}$$

To find the embedding $\mathbf{c}_d$ we maximize the probability $p(d|\mathbf{c}_d)$, which through standard procedures amounts to

$$\begin{gathered} \mathbf{c}_d = \underset{\mathbf{c}_d \in \mathbb{R}^d}{\arg\max} \log p(d|\mathbf{c}_d) \propto \frac{1}{|d|}\sum_{w \in d} \frac{a}{p(w)+a}\mathbf{v}_{w_i} \\ \text{where} \quad a = \frac{1-\alpha}{\alpha Z_{\tilde{\mathbf{c}}_d}} \end{gathered} \tag{3.4}$$

$Z_{\tilde{\mathbf{c}}_d}$ can be assumed to be roughly the same for all $\tilde{\mathbf{c}}_d$ since word vectors are roughly uniformly dispersed [35]. The now calculated embeddings $\mathbf{c}_d$ still include the common component $\mathbf{c}_0$. This is calculated as the first principal component from the matrix $\mathbf{D}$, where each row is a document embedding $\mathbf{c}_d^i$ from the dataset $\mathcal{D}$. Each document vector can then be updated as

$$\mathbf{c}_d^i \leftarrow \mathbf{c}_d^i - \mathbf{c}_0 \mathbf{c}_0^T \mathbf{c}_d^i \tag{3.5}$$

The embeddings given by (3.5) are, to enable classification, projected and fed through a softmax layer. This is identical to how HAT perform classification from the generated document embeddings. Other differences and similarities are described below.

- The choice of the word embeddings used. SIF uses the context-independent GloVe embeddings while HAT makes use of state-of-the-art contextualized embeddings generated by BERT.
- HAT make explicit use of documents' hierarchical nature through the attention mechanism. This weighting is in contrast to the above described SIF algorithm not bound to be described by the approximated word probabilities $p(w)$ and is instead calculated using the local context.
- A parallel could be drawn between how the discourse vector in SIF and the context vectors in HAT are used fo measure word- and sentence importance.

3.2 Paragraph Vector

Distributed Memory of Paragraph Vector (PV-DM) and *Distributed Bag of Words version of Paragraph Vector* (PV-DBOW) was proposed by Le et al. [7]. These algorithms build on work by Mikolov et al. [2] through extending the algorithm used for creating word embeddings to generate ones for documents as well. The paragraph vectors, or document embeddings, were learnt in parallel with the word embeddings for both algorithms. The training procedure for these followed the one found in [2] closely. Mikolov et al. did in this work train word embeddings, stored as the columns of a matrix $\mathbf{W}$, to maximize the average log probability of the word over the entire corpus

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t | w_{t-k}, ..., w_{t+k}) \tag{3.6}$$

where the probability is modelled by a softmax classifier.

$$p(w_t|w_{t-k},...,w_{t+k}\ ) = \frac{e^{y_{w_t}}}{\sum_i e^{y_{w_i}}} \tag{3.7}$$

Here, y_{w_i} represent the un-normalized log-probability for each word i which is calculated through a forward pass of the one-layer network in equation (3.8).

$$y_{w_i} = b + \mathbf{U}h(w_{t-k},...,w_{t+k},\mathbf{W}) \tag{3.8}$$

$\mathbf{U}$ and b are parameters of the model while $h(\cdot)$ is either a concatenation or averaging of the context words embeddings from $\mathbf{W}$.

Le et al. extended these equations to enable training of document embeddings. The PW-DM algorithm trained a document embedding matrix $\mathbf{D}$ in which each document was represented by a column vector. This vector was concatenated alongside the context words in equation (3.8) to predict the centre word. This provided information from outside the scope of the context words [7]. The algorithm shown in equations (3.6)-(3.8) were thus modified by replacing $h(w_{t-k},...,w_{t+k},\mathbf{W})$ with $h(w_{t-k},...,w_{t+k},\mathbf{W},\mathbf{D})$.

PV-DBOW was a simplification of PV-DM where the document vector from $\mathbf{D}$ was the only context used to predict randomly sampled words from the document. Equations (3.6)-(3.8) would therefore be modified as follows

$$\frac{1}{|\mathcal{R}|}\sum_{w_i\in\mathcal{R}} \log p(w_i|\mathbf{D}) \tag{3.9}$$

$$p(w_i|\mathbf{D}) = \frac{e^{y_{w_i}}}{\sum_i e^{y_{w_i}}} \tag{3.10}$$

$$y_{w_i} = b + \mathbf{U}h(w_i,\mathbf{D}) \tag{3.11}$$

$\mathcal{R}$ represents the set of randomly selected words from the document in question.

One of the downsides with both these approaches is the computational complexity of creating a document embedding at inference time. The process first freezes $\mathbf{W}$, $\mathbf{U}$ and b while finding the best linear combination of columns in $\mathbf{D}$ through gradient decent to represent the current document. This is one of the differences between these models and HAT, which is further outlined below.

- Both PV models require an additional optimization step during inference for creating the document embeddings. HAT on the other hand can infer the document embeddings with a single forward propagation through the network.

- HAT utilize a pre-training procedure of its word embeddings through BERT where as both PV models include word representations in the training step.

3.3 Document Vector Through Corruption

The success of previous works in preserving syntactic and semantic relations even after addition or subtraction of word vectors suggest that efficient document vectors can be achieved through averaging the learnt word embeddings [6]. Chen et al. argued that such averaging should be performed from a *corrupted* document, where some of the word embeddings are closer to zero than others.

This was achieved through an unbiased, dropout corrupted average of the word vectors from each document and the local context. These were together used to perform the task of predicting the next word [6]. Let w_t be the target word from the vocabulary V and $\mathbf{U}$ be the word embedding matrix where column i represent the embedding for w_i. Also, let $\mathbf{c}_t$ be the local context and $\tilde{\mathbf{x}}$ the corrupted global context. The latter two are BOW vectors which encode the occurrence for word j in either context by setting element j equal to 1, for example $c_j = 1$. $(\mathbf{v}_{w_t}$ is the output projection for word w_t. The algorithm is then trained to maximize the conditional probability in equation (3.12).

$$p(w_t|\mathbf{c}_t, \tilde{\mathbf{x}}) = \frac{\exp\left(\mathbf{v}_{w_t}^\top(\mathbf{U}\mathbf{c}_t + \frac{1}{T}\mathbf{U}\tilde{\mathbf{x}})\right)}{\sum_{w' \in V} \exp(\mathbf{v}_{w'}^\top(\mathbf{U}\mathbf{c}_t + \frac{1}{T}\mathbf{U}\tilde{\mathbf{x}}))} \tag{3.12}$$

Adding this corrupted global context acts as a regularization such that both common and rare words, those that are thought to bring less information, are learnt representations close to zero [6]. This enables document embeddings to be generated through averaging over the learnt representations for all words in the document D, as in equation (3.13).

$$\mathbf{d} = \frac{1}{T}\sum_{w \in D} \mathbf{u}_w \tag{3.13}$$

$\mathbf{u}_w$ is the learnt word representation for w found as a column of $\mathbf{U}$.

Comparing Doc2VecC with our proposed HAT yield both similarities and differences. The points below highlight these.

- Both Doc2VecC and HAT create document representations through a sum of all word embeddings within a document. However, HAT regard information at two levels of hierarchy and use this to assign the relative importance of each word and sentence. Doc2VecC on the other hand encode the importance of each word directly into the magnitude of the vector representation.

- The learnt word representations for Doc2VecC are specialized for the dataset while HAT utilize task agnostic embeddings from BERT. Further, these word embeddings are pre-trained on large task-agnostic datasets.

3.4 Hierarchical Attention Network

It can be beneficial to utilize the hierarchical nature of text when creating document embeddings [8]. Yang et al. introduced a network that creates *sentence embeddings* from words within each sentence and then combine these embeddings into document representations. This network was named the Hierarchical Attention Network (HAN) not to be confused with HAT. It achieved significantly better results than previous methods [8].

HAN generate document embeddings through the following process. First, words $w_{it}, t \in [0, T_i]$ in sentence i are encoded as word embeddings $\mathbf{x}_{ij}$ by an embedding matrix $\mathbf{W}_e$. These word vectors then interacts with their context (words within the same sentence) using a bi-directional Gated Recurrent Unit (GRU). This gives a hidden representation $\mathbf{h}_{it}$ of each word. Here an attention mechanism is applied to decide the importance of each word in sentence i. These calculations are shown in equations (3.14)-(3.16).

$$\mathbf{u}_{it} = \tanh(\mathbf{W}_w \mathbf{h}_{it} + \mathbf{b}_w) \tag{3.14}$$

$$\alpha_{it} = \frac{\exp(\mathbf{u}_{it}^\top \mathbf{u}_w)}{\sum_j \exp(\mathbf{u}_{ij}^\top \mathbf{u}_w)} \tag{3.15}$$

$$\mathbf{s}_i = \sum_t \alpha_{it} \mathbf{h}_{it} \tag{3.16}$$

$\mathbf{W}_w$, $\mathbf{b}_w$ and $\mathbf{u}_w$ are parameters of the network. $\mathbf{u}_w$ can be thought of as a vector that has an *understanding* of which hidden word-representations $\mathbf{u}_{it}$ are of importance to the information in the sentence. Equation (3.16) shows that the sentence embedding $\mathbf{s}_i$ is a weighted average of the hidden word representations. The weight for each word is assigned through the compatibility between $\mathbf{u}_w$ and the word's hidden representation.

The above GRU and attention calculations are repeated for the second level of hierarchy. In place of the word representations are now sentence embeddings $\mathbf{s}_i$ used instead. Equations (3.14)-(3.16) are thus repeated with new parameters $\mathbf{W}_s$, $\mathbf{b}_s$ and $\mathbf{u}_s$. This results in a single document embedding $\mathbf{d}$. This vector is linearly projected to the output space where a softmax activation is applied so that classification can be performed, see equation (3.17).

$$p = \text{softmax}(\mathbf{W}_c\mathbf{d} + \mathbf{b}_c) \tag{3.17}$$

The architecture of HAT is largely inspired by HAN where some key features have been changed.

- HAT (our contribution) rely on pre-trained embeddings from BERT rather than including an embedding layer into the model as found in HAN. It is because of this reason possible for HAT to rely on a single block of recurrence rather than applying it at both levels of hierarchy.
- The networks' architecture used to process the sequence of sentence embeddings differ. HAT rely on the Transformer while HAN use the recurrent structure of GRU in its place.

3.5 Word Mover's Embeddings

The document embedding algorithm presented by Wu et al. [36] builds on Word Mover Distance (WMD) [37]. The WMD measure the distance between two documents in the Word2Vec vector space. The contribution from Wu et al. was combining the WMD measure with a method for calculating a positive-definite kernel from a distance measure (D2KE) [38]. This enabled creation of document embeddings which exhibited significantly reduced computational cost while it also improved performance in classification tasks [36]. The mathematical details for creating these document embeddings are presented below.

3.5.1 Word Mover's Distance

WMD is a special case of the Earth Mover's Distance which describes the distance between two documents [37]. This distance is for two documents d_1 and d_2 found through taking word-distance into consideration. Let $|d_1|$ and $|d_2|$ be the number of unique words in each document. $\mathbf{f}_{d_1}$ and $\mathbf{f}_{d_2}$ are introduced, in close parallel to how a BOW vector is defined, as the *normalized frequency vectors* for documents d_1 and d_2. These enable the WMD to be defined as follows.

$$\begin{aligned} \text{WMD}(d_1, d_2) \equiv \min_{F \in \mathbb{R}_+^{|d_1| \times |d_2|}} \langle C, F \rangle \\ s.t. \quad F\mathbf{1} = \mathbf{f}_{d_1}, F^\top \mathbf{1} = \mathbf{f}_{d_2} \end{aligned} \tag{3.18}$$

C_{ij} is the *cost* of travelling between the i:th word $w_{i,1}$ in d_1 and the j:th word $w_{j,2}$ in d_2, defined as the distance between words in the word-embedding space, $C_{ij} = |\mathbf{v}_{w_{i,1}}, \mathbf{v}_{w_{j,2}}|$. F is the transportation flow matrix where each element F_{ij} denotes the amount of flow travelling from $w_{i,1}$ to $w_{j,2}$ [37].

3.5.2 Word Mover's Embeddings

The word mover kernel, from which the document embeddings are retrieved, is defined by

$$\begin{aligned} k(d_1, d_2) &\equiv \int p(r)\phi_r(d_1)\phi_r(d_2)dr \\ \text{where} \quad \phi_r(d) &= \exp(-\gamma \text{WMD}(d, r)) \end{aligned} \tag{3.19}$$

r can in these equations be interpreted as a document of randomly sampled words $\{w_j\}_{j=1}^{D}$. These *random* documents are sampled from $p(r)$ which govern their probability. Since these documents consist of vectors in the Word2Vec space is it this space that $p(r)$ has to model in order for the random documents to consist of a meaningful sets of words. It has been found that the word vectors in this space are approximately uniformly dispersed [35, 36]. This enables $p(r)$ to be modelled as a uniform distribution in each dimension. A random word $\mathbf{u} \in \mathbb{R}^d$ can therefore be sampled from $u_j \sim [v_{min}, v_{max}]$ for $j \in [1, d]$ where $v_{min/max}$ are constants.

Wu et al. argue that it is favourable for the number of words per random document, D, to be small because this variable represents the number of hidden global topics within each document. D is therefore for each random document sampled from $D \sim [1, D_{max}]$ for some constant D_{max}.

Calculating the exact kernel in equation (3.19) is computationally expensive. It can however be approximated through a Monte Carlo Approximation [36].

$$k(d_1, d_2) \approx \langle \mathbf{Z}(d_1), \mathbf{Z}(d_2) \rangle = \frac{1}{R}\sum_{i=1}^{R} \phi_{r_i}(d_1)\phi_{r_i}(d_2) \tag{3.20}$$

Here, the set of random documents $\{r_i\}_{i=1}^{R}$ are sampled from $p(r)$ which takes the random length of each document D_i into account. Wu et al. defined $\mathbf{Z}(d) \equiv \frac{1}{\sqrt{R}}[\phi_{r_1}(d), \phi_{r_2}(d), ..., \phi_{r_R}(d)]$ to represent the embedding for document x.

To provide some intuition behind this expression one can think of each $Z_i(d) = \phi_{r_i}(d)$ as measuring the compatibility between d and a set of random global topics contained in r_i. This intuition tells us that a larger R would create a more diverse comparison of d thus providing a richer description of the document.

Comparing WME with HAT yields both similarities and differences. Some of these are listed below:

- Both algorithms rely on pre-trained word-embeddings. WME utilize the context independent Word2Vec embeddings while HAT use context dependent word representations from BERT.
- WME does not utilize the sequential- nor hierarchical nature of the documents. Both these features are used by HAT.

Chapter 4

Methods

With a grasp of previous works within the field of document embeddings it is now possible to revisit the research question first stated in section 1.2. This section will also outline how to evaluate these questions as well as give an introduction to the datasets used for this comparison.

4.1 Research questions

To reiterate, the questions are formulated as follows:

- *Is there merit in using current state-of-the-art word embeddings presented in [3] together with a neural network model only relying on attention for creating document embeddings for use in classification problems?*
- *How does the amount of training data in the case of longer documents affect the performance of our proposed model compared to current document embedding algorithms?*

4.2 Method

These research questions pose three problems to be evaluated. First, a baseline comparison between our algorithm and existing ones on standardized datasets used in previous literature. We will refer to this experiment as the *baseline comparison* and outline the details in section 4.2.1. Secondly, evaluate how the best performing models from the baseline comparison perform when the average length of each document exceeds one page. These results are then

Dataset	Classes	Train	Test	Avg #words
BBCSPORT	5	517	220	344
OHSUMED	10	3999	5153	168
IMDB-long-n	2	n	1000	711

Table 4.1: Dataset statistics [36] for baseline comparisons together with our subsampled IMDB dataset. The average length of the IMDB-long datasets is reported for the case when $n = 1000$. The average length is consistent enough for all tested n to not change the characteristic of the experiment.

compared to the ones found when addressing the third and final aspect of the research question; how performance varies with number of training documents. The details of how we address these final two questions are outlined in sections 4.2.2 and 4.2.3.

4.2.1 Baseline comparison

The baseline comparison measured the performance of HAT and that of the related works using two already existing datasets, BBC Sport and OHSUMED. The characteristics of these two datasets are shown in table 4.1. These are such as to focus the analysis on the domain where the length of each document is short. Short does in this context refer to an average length of three paragraphs or shorter, which in our experience equals 360 words. [1] Wu et al. presented results for the related works on these datasets [36] and was used as the basis for our comparison.

BBC Sport

This datasets consists of sport articles posted by BBC from five different sport categories: athletics, cricket, football, rugby and tennis [39]. The classes are not balanced but have been sampled between train and test set in such a way as to keep the same relative proportions. The average length of a document is 344 words.

[1]The keen observer will realize that BBC Sport and OHSUMED are only a subset of the datasets presented in [36]. The decision for this subset is that the average number of words for these datasets were among the higher ones while simultaneously keeping the number of training instances relatively low. This enables better insights into the performance of our proposed model in that particular scenario.

OHSUMED

OHSUMED is a dataset containing abstracts from medical research papers which in [36] have been sub-sampled to only include the first 10 classes. This dataset does not have balanced classes, but has equal relative proportion between test and train splits for each class. The average number of words per document is 168.

Algorithms

The algorithms that were used for the baseline comparison are listed below. We also highlight some features but refer to chapter 3 for the detailed explanation.

Smooth inverse frequency (SIF) Utilizing pre-trained GloVe word representations for creating document embeddings through clever weighted averages. See section 3.1 for a thorough description of the algorithm.

Word2Vec + tf-idf A weighted average of Word2Vec embeddings using tf-idf weights for each word vector. See section 2.5.2 for details on the weighting scheme and 2.5.3 for how the word embeddings are created.

PV-DBOW Distributed Bag of Word model of Paragraph Vectors extend the local context with a global document for predicting a hidden word. Section 3.2 goes into details of this algorithm.

PV-DM Distributed Memory of Paragraph Vectors only rely on a global document vector to predict a hidden word. The entire description can be found in section 3.2.

Doc2VecC Document Vectors through Corruption learns word representations for which the importance is encoded in their magnitude. The document embeddings can then be created through summation of all word vectors in the document. See section 3.3 for the exact details on how the weights are defined.

Word Mover's Embeddings Word Mover's Embeddings combines a distance measure for words vectors with a kernel function. The latter can be approximated which enables calculation of document embeddings. Details are presented in section 3.5.

4.2.2 Classification of page-long documents

The second research question address how the performance of HAT perform when the documents are longer than one page. We decided to limit the comparison to the two best performing algorithms in [36] (SIF and WME). We also decided to create a dataset of the specific characteristic required for this experiment through extracting documents from the IMDB-review dataset [40]. This dataset poses a binary classification problem of the sentiment of user-written movie reviews.

We extracted the *longest* 1000 reviews from both train and test splits, balanced between the two classes. We will refer to this dataset as *IMDB-long-1000*. Length was measured as number of sentences, counted as occurrences of any character in {".", "?", "!"} followed by a *space*. This might seem unnecessary when counting the number of words is a more direct approach, actually yielding the longest documents. Our reasoning is as follows. We observed reviews of poor grammatical quality and overall sentence structure when simply counting the number of words. These issues were not present when using the other length measurement, where we instead observed structured paragraphs and correct grammar. IMDB-long-1000 has 711 words per document on average, which is considerably more than the average of 500 that fit on a single A4-page.

4.2.3 Limiting the number of training instances

The third problem posed by the research questions was addressed through examining the effect limiting the number of training instances had on performance. We therefore further reduced the size of IMDB-long-1000 to subsets of 500, 250 and 100 documents. These datasets will be referred to as IMDB-long-n where n is the number of training documents. Each of these datasets have balanced classes. The performance was evaluated against the test set of size 1000.

4.3 Training procedure

For repeatability of our experiments will we here describe the training procedure and any parameter decisions, both for HAT as well as SIF and WME. We did, to enable a fair comparison, use the official implementations of each baseline algorithm and thoroughly followed the respective authors suggestions of how to optimize their algorithm. We also, where an architectural dimensionality choice could be made, kept the algorithms as similar as possible to further increase the fairness of the comparison.

4.3.1 Hierarchical Attention Transformer

We used the uncased BERT$_{\text{BASE}}$ model to generate token embeddings of dimension 768 by the sliding window approach described in section 2.7. The output from the embedding layer was the sum of the token embeddings from the last four layers of BERT. Our reasoning behind this choice was that it achieved the second best performance in Devlin et al.'s experiments [3]. It only fell behind concatenation of the last four layers, which we decided to disregard as it would increase the dimension of the embeddings by a factor of four only for a small increase in performance.

We used the Adam optimizer for training the HAT network with standard momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was found through a technique inspired by a fast.ai related forum post [2] which exponentially increase the learning rate while observing the loss. The learning rate with the steepest loss decrease is usually a good initial choice. We would if this choice still resulted in divergent behaviour reduce the learning rate by a factor of at least two, depending on the observed behaviour, and train the model again.

The architectural parameters of the HAT network were chosen to reflect the design of the original Transformer encoder. Thus, we used 8 attention heads [20] and decided to use 4 encoder layers throughout our analysis. The dropout probability was chosen through guidance from [20] and set in the range 0.25-0.3.

The validation loss was monitored during training and used as a guidance for when optimal performance was reached. At this point the parameters of the model were frozen and used to evaluate its performance on the test set.

4.3.2 Smooth Inverse Frequency

With the official implementation[3] we trained the SIF algorithm for all IMDB-long-n datasets. An initial parameter search for the parameters governing the weighting for the probability of out of context words α and the relative impact

[2]The article can be found here https://www.jeremyjordan.me/nn-learning-rate/. The evidence for why such a schedule would be more efficient for training is mostly anecdotal, but since neural network training in and of its self could be regarded an art form is this enough of a reason to apply it.

[3]The github repository can be found here https://github.com/PrincetonML/SIF

of the common discourse vector β was performed on IMDB-long-1000. These values were then kept fix for the other n. [4] The parameter a found in equation (3.4) did not affect the performance of the algorithm enough to be optimized [5] and was therefore set within the suggested range [5]. The models was trained until the validation loss increased, and the performance was evaluated over multiple restarts.

4.3.3 Word Mover's Embeddings

Using the official WME implementation[5] we performed a 10-fold cross-validation for finding L-SVM parameters as well as model hyper-parameters. Each model was during training, as described in section 3.5, evaluated for increasing values of R (the number of random documents generated) to ensure that the highest possible performance was achieved with the current dataset.

4.3.4 Dataset management and model evaluation

All datasets were split 90% / 10% between training and evaluation data in order to enable model evaluation during training and hyper-parameter search. Some parameters were left out of this search where authors of the respective papers provided good settings. Worth mentioning is the fact that the test set was never used for either parameter- nor model selection to reduce bias in our experiments. All model evaluations are averages of multiple initializations, where model selection was entirely based on validation data performance.

We applied early stopping during training of the neural network based algorithms through monitoring of the validation loss and halted training when it did not improve for 1000 steps of training. The batch size varied from 2 to 16 between the different datasets in order to maximize the GPU utilization without exceeding its memory capacity.

[4]This is defended by (1) giving the algorithm a best case scenario and (2) reduced the need of thorough parameter search for each dataset and thus more efficient experimentation.

[5]The github repository can be found here: https://github.com/IBM/WordMoversEmbeddings

Chapter 5

Results

5.1 Baseline comparison

The baseline comparison results are found in table 5.1. Our algorithm, Hierarchical Attention Transformer (HAT), outperformed all other ones on both datasets. BBCSPORT present a document classification task of articles from five different sports (athletics, cricket, football, rugby and tennis) published by BBC. OHSUMED is another document classification problem where the task is to predict the category of medical abstract from ten possible ones.

Dataset	SIF(GloVe)	Doc2VecC	PV-DBOW	PV-DM	WME	HAT
BBCSPORT	97.3 ± 1.2	90.5 ± 1.7	97.2 ± 0.7	97.9 ± 1.3	98.2 ± 0.6	$\mathbf{98.8 \pm 0.3}$
OHSUMED	67.1	63.4	55.9	59.8	64.5	**67.6**

Table 5.1: Accuracy on test set for each algorithm and dataset presented in [36] as well as for our algorithm HAT. The highest performance is shown in bold font.

5.2 IMDB-long-n

The results from the binary, sentiment classification task posed by IMDB-long-n for the three algorithms used in this comparison are shown in table 5.2. HAT achieve significantly higher accuracy on the two larger datasets while falling behind on the smaller ones.

Dataset	SIF(GloVe)	WME	HAT
IMDB-long-1000	83.0 ± 0.4	84.6	**89.6 ± 0.4**
IMDB-long-500	80.2 ± 1.4	82.5	**87.9 ± 1.1**
IMDB-long-250	77.0 ± 1.2	**80.9**	77.8 ± 3.5
IMDB-long-100	73.1 ± 0.3	**76.5**	72.1 ± 4.1

Table 5.2: Accuracy on test set for each algorithm and dataset. The best result for each dataset is shown in bold font.

Chapter 6

Discussion

This chapter will discuss the results in light of the research questions. It will also provide a reflection on the validity of the experiments, both from an internal- and external validity point of view.

6.1 Baseline comparison

It becomes apparent from table 5.1 that the Hierarchical Attention Transformer (HAT) has merit. HAT outperform all baseline algorithms on both datasets with a relative error improvement of 33% for BBC Sport and 1.5% for OHSUMED. We reflect on this disparity in the sections that follow.

6.1.1 BBC Sport

We argue that the reason for why BERT embeddings in combination with HAT was able to achieve the observed performance is twofold. First of all are the embeddings created by BERT pre-trained on English Wikipedia [3] which already include many of the sport related words and phrases that might appear in BBC articles. This gives the embedding layer greater probability of creating high quality token embeddings since most tokens are already known from the pre-training step. An example highlighting this fact is shown in figure 6.1. There, the attention weights are visualized for a *cricket* article from the BBC sports dataset. We find that the word *cricket* is the most important one and is therefore the embedding that will characterize this document. We also see that *Shaun* and *Ashley Giles*, which are two famous players of the sport, are regarded as important features of the article.

Secondly, the hierarchical nature of HAT enables it to efficiently disregard much of a document's content. This is also apparent in figure 6.1 where generic sport sentences are assigned an exceedingly small attention weight. Such sentences could reflect on any sport in the dataset and is therefore not of importance for finding the key characteristics of a class. This example highlight the ability of HAT to filter through large bodies of text and focus only on sentences and words of importance. This differentiates it from the algorithms we compared it to in a fundamental way, which we argue support the merit of this algorithm.

To further strengthen the conclusions drawn from the observations in figure 6.1 are more such examples included in appendix A.

6.1.2 OHSUMED

Moving our attention to OHSUMED we observe a relative error improvement more than one order of magnitude smaller compared to that achieved for BBC Sport. The absolute error improvement is however of comparable size (0.5% vs 0.6%).

A possible reason for this results is the difference between the vocabularies present during pre-training and testing. The pre-training phase of the word embedding algorithms used (BERT, GloVe and Word2Vec) enable semantic relations between similar words to be established within the embeddings space. Rare words therefore suffer from lower quality embeddings. It would in this particular case result in specific medical terms, that are essential to the characteristic of the class, not being understood during embedding time. This flaw propagates through each of the algorithms which results in the poor performance. HAT is despite this fact able to achieve new state-of-the-art performance, taking the spot from SIF utilizing GloVe embeddings.

6.2 IMDB-long-n

Table 5.2 shows that HAT outperforms its competition by a significant margin for the larger datasets in our proposed set IMDB-long-n. HAT was able to achieve relative error improvements for IMDB-long-1000 and IMDB-long-500 of 32.5% and 30.8% respectively. We attribute this gain in performance to its ability to focus on word embeddings within each sentence and sentence embeddings within the entire document in such a way to ignore noise. We have already highlighted this with an example in figure 6.1. This ability differs from how SIF filter out the word embeddings that are thought to be important

- rain holds up england fifth test , centurion .
- south africa v england start delayed , rain england lead the series 2 - 1 with just this final test remaining so south africa desperately need to win .
- the officials had hoped play could get under way at noon local time but a further downpour led to an early lunch being taken by the players .
- for england , simon jones is expected to replace james anderson while the hosts play both andrew hall and andre nel , with batsman boeta dippenaar dropped .
- dale steyn had already been released from the squad for centurion and it means south africa go in with five seamers and one spin bowler .
- on friday , england captain michael vaughan had said it was important his team targeted victory and not simply a draw .
- he said : " you can ' t set out over five days to draw a game .
- we have to try to win .
- if you go out in a mindset to draw you don ' t have the right attitude .
- " we ' ll just set out in a positive manner and try to put the south africans under a lot of pressure .
- " england , who have not won a series in south africa for 40 years , will play on a green pitch which has endured a long spell of wet and humid weather .
- the weather forecast for saturday onwards is fine , and early starts on all four days from then could ensure there is plenty of time to achieve a result .
- middlesex opener andrew strauss , with a stunning aggregate of 612 runs in the four tests played to date , said south africa had every incentive to throw caution to the wind .
- he said : " it ' s been a great series for me .
- it ' s been very satisfying .
- " but i think we all realise that this last game is very important and we can ' t afford to take our foot off the pedal now .
- " sometimes when you ' ve got a lead it ' s quite hard to keep yourself motivated .
- " we are pretty sure that the south africans will come back hard at us .
- they have a lot of motivation to do well against us this week .
- " south africa captain graeme smith said the colour of the pitch was probably misleading .
- he said : " the pitch has played pretty well all year .
- it might look a bit juicy but will probably play pretty good .
- " he added that his team had the capability of bouncing back .
- " we came back in new zealand from 1 - 0 down to level the series in wellington so we ' ve been in this situation before .
- " we ' ve just got to play some really good cricket for four to five days .
- we ' ve had had a lot of iffy performances so far .
- " with a bit of luck here or there or playing some better cricket at certain times it could have easily been the other way round .
- " it ' s been that sort of series , ebb and flow , and i ' m sure it ' s going to be that way again .
- " graeme smith (capt .
-) , herschelle gibbs , jacques rudolph , jacques kallis , ab de villiers , mark boucher (wkt) , andrew hall , nicky boje , shaun pollock , makhaya ntini , andre nel .
- marcus trescothick , andrew strauss , robert key , michael vaughan (capt .
-) , graham thorpe , andrew flintoff , geraint jones (wkt) , ashley giles , matthew hoggard , steve harmison , simon jones .

Figure 6.1: Attention visualization for a randomly selected test article from the BBC Sport dataset.The article was correctly predicted to be from the *cricket* class. Blue highlight at the start of each sentence indicate its attention weight, while red highlight correspond to the assigned attention weight for each word.

for the representation of each document, which is based on an estimate of the probability for each word.

6.3 Validity discussion

It is important to discuss experimental design decisions and data management practices in light of their validity [41]. This section is devoted to reflect on the validity from both an internal and external point of view.

6.3.1 Internal validity

The discussion of internal validity reflects the quality of the data analysis presented in the work. We here have to argue that the cause and effect relationship between the variables attributed the change in outcome really did cause this change. This is to avoid confounding of the experiment. Only when the risk of confounding is low can the internal validity be high [41].

In our performance comparison between different models is the model architecture together with the word embeddings used the variable that should describe the change in observed performance. To eliminate variations in performance caused by variance in either training or testing data we have ensure the same dataset splits were used throughout. We also kept the test set hidden from the models during training so that the reported performance measures the models generalizability rather than memorizing capabilities.

Our reported results are in addition averages over multiple random parameter initializations between which training data order also was shuffled. This reduces the variance in performance to better reflect the actual achieved metric.

6.3.2 External validity

External validity refers to the applicability of the conclusions drawn from our experiments in other environments [41]. Particularly important here is how general the performance of the models are when applied to other datasets.

We have in our experiments reported accuracies for the algorithms using three distinct datasets. They vary both in vocabulary, length and number of classes. We went further than this to also report the performance on the IMDB-long dataset using different amounts of training data. These factors support the generalizability of our findings to a broader range of problems. However, due to the extreme variations that exist between datasets make prediction of performance in other scenarios from our findings irrelevant.

During construction of the IMDB-long-n dataset we intentionally made sure to sub-sample the original IMDB dataset [40] to yield grammatically correct and well structured reviews. This resulted in documents being more comparable to ones that can be addressed in the future.

Chapter 7

Conclusions

We have in this work introduced a novel architecture, the Hierarchical Attention Transformer (HAT), for creating powerful document embeddings. It outperforms state-of-the-art baseline algorithms on two interesting datasets.

Examining the performance of HAT in the special case of longer documents, more than one page of written text, we found that it excels when there was enough training data available. With 1000 documents used for training from our generated IMDB-long dataset HAT achieved 89.6 % accuracy which represent a relative error improvement of 32.5 % compared to its best competitor. However, the performance of HAT was lower than its competition when less than 500 documents were used for training. We believe that a more thorough fine-tuning process of HAT's hyper-parameters than these experiments performed could outperform its competition in this case too.

7.1 Future work

We were able to achieve highly competitive performance without any explicit hyper-parameters tuneing or weight regularization. This does not imply that such practices can be overlooked, but rather the strength of HAT. We expect that even higher performance could be achieved through a more careful optimization process and will outline some of these in this section.

Regularization techniques such as dropout and *weight decay* could improve the convergence process when the number of training documents are limited. A careful hyper-parameter search would be necessary for finding optimal parameters in this case. Parameters such as the number of layers and attention heads could also be experimented with to reduce the model complexity, hence decreasing the model variance, which could strike a better balance between it and the bias.

We also believe experiments with the optimization algorithm could yield improved performance. We would in particular want to experiment with Momentum SGD as the optimizer, which can achieve higher performance *if* the learning rate is tuned carefully [42]. There is also evidence that learning rate scheduling improves convergence which would be interesting to examine [43].

Bibliography

[1] Gerard Salton and Christopher Buckley. “Term-weighting Approaches in Automatic Text Retrieval”. In: *Inf. Process. Manage.* 24.5 (Aug. 1988), pp. 513–523. ISSN: 0306-4573. DOI: 10.1016/0306-4573(88)90021-0. URL: http://dx.doi.org/10.1016/0306-4573(88)90021-0.

[2] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* abs/1301.3781 (2013).

[3] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: (Oct. 2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[4] S. E. Robertson and S. Walker. “Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval”. In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’94. Dublin, Ireland: Springer-Verlag New York, Inc., 1994, pp. 232–241. ISBN: 0-387-19889-X. URL: http://dl.acm.org/citation.cfm?id=188490.188561.

[5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. “A Simple but Tough-to-Beat Baseline for Sentence Embeddings”. In: (2017).

[6] Minmin Chen. “Efficient Vector Representation for Documents through Corruption”. In: *CoRR* abs/1707.02377 (2017).

[7] Quoc V. Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *ICML*. 2014.

[8] Zichao Yang et al. “Hierarchical attention networks for document classification”. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 1480–1489.

[9] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: (2019).

[10] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[11] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407.

[12] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2015).

[13] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). DOI: `10.1109/cvpr.2016.90`. URL: `http://dx.doi.org/10.1109/CVPR.2016.90`.

[14] Hao Li et al. “Visualizing the loss landscape of neural nets”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 6389–6399.

[15] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[16] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[17] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[18] Sepp Hochreiter et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[19] Matthew E Peters et al. “Deep contextualized word representations”. In: *arXiv preprint arXiv:1802.05365* (2018).

[20] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[21] Zellig S Harris. “Distributional structure”. In: *Word* 10.2-3 (1954), pp. 146–162.

[22] Reinhard Kneser and Hermann Ney. “Improved backing-off for m-gram language modeling”. In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 1995, pp. 181–184.

[23] Stanley F Chen and Joshua Goodman. “An empirical study of smoothing techniques for language modeling”. In: *Computer Speech & Language* 13.4 (1999), pp. 359–394.

[24] Yoshua Bengio et al. “A Neural Probabilistic Language Model”. In: *Journal of Machine Learning Research* 3.Feb (2003), pp. 1137–1155. ISSN: ISSN 1533-7928. URL: `http://www.jmlr.org/papers/v3/bengio03a.html`.

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

[26] Bryan McCann et al. “Learned in Translation: Contextualized Word Vectors”. In: *NIPS*. 2017.

[27] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[28] Alex Wang et al. “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *BlackboxNLP@EMNLP*. 2018.

[29] Rowan Zellers et al. “SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference”. In: *EMNLP*. 2018.

[30] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).

[31] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[32] Wilson L Taylor. ““Cloze procedure”: A new tool for measuring readability”. In: *Journalism Bulletin* 30.4 (1953), pp. 415–433.

[33] Yukun Zhu et al. “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (Dec. 2015). DOI: `10.1109/iccv.2015.11`. URL: `http://dx.doi.org/10.1109/ICCV.2015.11`.

[34] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. Tech. rep. URL: `https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

[35] Sanjeev Arora et al. “A latent variable model approach to pmi-based word embeddings”. In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 385–399.

[36] Lingfei Wu et al. “Word Mover’s Embedding: From Word2Vec to Document Embedding”. In: *EMNLP*. 2017.

[37] Matt Kusner et al. “From word embeddings to document distances”. In: *International Conference on Machine Learning*. 2015, pp. 957–966.

[38] Lingfei Wu et al. “Word Mover’s Embedding: From Word2Vec to Document Embedding”. In: *arXiv preprint arXiv:1811.01713* (2018).

[39] Derek Greene and Pádraig Cunningham. “Practical solutions to the problem of diagonal dominance in kernel document clustering”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 377–384.

[40] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: `http://www.aclweb.org/anthology/P11-1015`.

[41] Hamid Reza Faragardi. “Optimizing timing-critical cloud resources in a smart factory”. PhD thesis. Mälardalen University, 2018.

[42] Ashia C Wilson et al. “The marginal value of adaptive gradient methods in machine learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4148–4158.

[43] Martin Popel and Ondřej Bojar. “Training tips for the transformer model”. In: *The Prague Bulletin of Mathematical Linguistics* 110.1 (2018), pp. 43–70.

Appendix A

Attention examples

This appendix is dedicated to showcase examples of the attention visualization the HAT model can produce. We have included examples from all examined datasets.

- after the lead actress of the opera is killed in a car accident , her young understudy , betty , is brought to the forefront .
- that ' s very lucky for her , with one problem : she has an admirer that has decided he will kill all her friends and make her watch .
- what is his connection to the opera , and what is his fascination with betty ?
- i love dario argento with every part of my body .
- and i ' m not an orthodox fan , i think .
- many people , particularly critics , praise his earlier work (" suspiria " and " deep red ") but really frown on later films , such as " sleepless " , which i liked .
- my favorite , " phenomena " , is usually vastly underrated .
- " opera " tends to fall somewhere in between .
- some consider it one of his last great films , others see it as part of his so - called decline .
- i loved it .
- the picture is crisp , the music is great (unlike other critics , i love the metal soundtrack) , the female lead is someone i can feel for (not unlike jennifer connelly from " phenomena ") .
- and the imagery .
- .
- .
- wonderful .
- great cinematography , and some amazing kill scenes .
- the concept of taping needles to a person ' s eyes so they cannot blink .
- .
- .
- brilliant .
- my assistant tina thinks this looked fake , but even if it does , the idea is more than enough to pay off .
- and some great effects , like a knife blade coming up inside a man ' s mouth ?
- awesome .
- jim harper calls the film " stunning " and calls attention to the " innovative cinematography , well - constructed shots and exceptionally violent murders .
- " i agree with this completely - - one shot follows the camera through winding tunnels , and there is a very interesting visual use of crows throughout the story .
- mike mayo likewise calls it " visually fascinating eye - candy " and lauds the " crisp editing and flowing camera - work " .
- it ' s really a wonder that this is not one of argento ' s more highly - praised works .
- argento returned to the opera with " phantom of the opera " , which was a bit of a failure despite the casting of his daughter asia and julian sands .
- even more interesting , this same year offered the release of michele soavi ' s " stagefright " , which (like " opera ") has a killer loose inside a theater killing off the people involved with the presentation .
- both are great films , with soavi ' s more on the slasher side .
- (soavi actually served as second unit director on " opera " .
- .
- .
- you can make your own conclusions .
-) my only complaint with this film is the length and pacing .
- while it is very beautifully shot and the kill scenes are glorious , they are not as frequent as they should be .
- the first one takes over a half hour , and then we get down times between them .
- the lead actress should be in constant terror , but she is given time between kills to calm down as if everything is normal again .
- not cool , dario .
- we need to keep the suspense low and the intensity high .

Figure A.1: Attention visualization for a randomly selected test article from the IMDB-long dataset. Blue highlight at the start of each sentence indicate its attention weight, while red highlight correspond to each words assigned attention weight.

- gardener wins double in glasgow britain ’ s jason gardener enjoyed a double 60m success in glasgow in his first competitive outing since he won 100m relay gold at the athens olympics .
- gardener cruised home ahead of scot nick smith to win the invitational race at the norwich union international .
- he then recovered from a poor start in the second race to beat swede daniel persson and italy ’ s luca verdecchia .
- his times of 6 .
- 61 and 6 .
- 62 seconds were well short of american maurice greene ’ s 60m world record of 6 .
- 39secs from 1998 .
- ” it ’ s a very hard record to break , but i believe i ’ ve trained very well , ” said the world indoor champion , who hopes to get closer to the mark this season .
- ” it was important to come out and make sure i got maximum points .
- my last race was the olympic final and there was a lot of expectation .
- ” this was just what i needed to sharpen up and get some race fitness .
- i ’ m very excited about the next couple of months .
- ” double olympic champion marked her first appearance on home soil since winning 1500m and 800m gold in athens with a victory .
- there was a third success for britain when edged out russia ’ s olga fedorova and sweden ’ s jenny kallur to win the women ’ s 60m race in 7 .
- 23secs .
- maduaka was unable to repeat the feat in the 200m , finishing down in fourth as took the win for russia .
- and the 31 - year - old also missed out on a podium place in the 4x200m relay as the british quartet came in fourth , with russia setting a new world indoor record .
- there was a setback for jade johnson as she suffered a recurrence of her back injury in the long jump .
- russia won the meeting with a final total of 63 points , with britain second on 48 and france one point behind in third .
- led the way for russia by producing a major shock in the high jump as he beat olympic champion stefan holm into second place to end the swede ’ s 22 - event unbeaten record .
- won the triple jump with a leap of 16 .
- 87m , with britain ’ s tosin oke fourth in 15 .
- 80m .
- won the men ’ s pole vault competition with a clearance of 5 .
- 65m , with britain ’ s nick buckfield 51cm adrift of his personal best in third .
- and won the women ’ s 800m , with britain ’ s jenny meadows third .
- there was yet another russian victory in the women ’ s 400m as finished well clear of britain ’ s catherine murphy .
- chris lambert had to settle for fourth after fading in the closing stages of the men ’ s 200m race as sweden ’ s held off leslie djhone of france .
- france ’ s won the men ’ s 400m , with brett rund fourth for britain .
- took victory for sweden in the women ’ s 60m hurdles ahead of russia ’ s irina shevchenko and britain ’ s sarah claxton , who set a new personal best .
- italy grabbed their first victory in the men ’ s 1500m as kicked over the last 200 metres to hold off britain ’ s james thie and france ’ s alexis abraham .
- a botched changeover in the 4x200m relay cost britain ’ s men the chance to add further points as france claimed victory .

Figure A.2: Attention visualization for a randomly selected test article from the BBC Sport dataset. The article is from the *athletics* class. Blue highlight at the start of each sentence indicate its attention weight, while red highlight correspond to each words assigned attention weight.

- coronary bypass surgery : is the operation different today ?
- patients undergoing coronary bypass grafting have undergone an evolution in recent years .
- to document this change , we analyzed two groups of patients in 1981 (n = 1586) and 1987 (n = 1513) to document preoperative and postoperative variables important in determining immediate morbidity and mortality after isolated coronary bypass .
- between 1981 and 1987 , patients were found to be older (greater than or equal to 70 years , 8 .
- 7 % versus 21 .
- 8 % , p less than 0 .
- 0001) , more often diabetic (15 % versus 24 % , p less than 0 .
- 0001) , have a greater prevalence of triple vessel disease (14 .
- 5 % versus 46 .
- 1 % , p less than 0 .
- 0001) , and have more left ventricular dysfunction (ejection fraction 0 .
- 60 + / - 14 versus 0 .
- 54 + / - 13 , p less than 0 .
- 0001) .
- to facilitate analysis and because of overlap between subgroups , we subdivided patients into three subgroups for statistical comparison of the years 1981 and 1987 : subgroup i , no prior procedure (n = 1546 in 1981 and 1396 in 1987) ; subgroup ii , optimal group (n = 503 in 1981 and 292 in 1987 , and defined as no prior procedure , ejection fraction greater than or equal to 0 .
- 50 and age less than 65 years) ; subgroup iii , patients having reoperations (n = 40 in 1981 and 117 in 1987) .
- internal mammary artery grafting was infrequently used in 1981 but was used in 72 .
- 1 % in 1987 .
- major postoperative morbidity between the 2 years for the total population increased significantly : need for intraaortic balloon pumping , 1 .
- 4 % versus 4 .
- 7 % , p less than 0 .
- 0001 ; myocardial infarction 3 .
- 5 % versus 5 .
- 5 % , p less than 0 .
- 008 ; stroke , 1 .
- 4 % versus 2 .
- 8 % , p less than 0 .
- 008 ; and wound infection , 1 .
- 0 % versus 3 .
- 0 % , p less than 0 .
- 001 .
- wound infection (all types) in 1987 was increased sevenfold in patients having a perioperative myocardial infarction (0 .
- 7 % versus 5 % , p less than 0 .
- 0001) .
- for young patients with good left ventricular function (subgroup ii) , there was no increase in these morbid events between 1981 and 1987 .
- hospital mortality in the total population increased significantly between 1981 and 1987 from 1 .
- 2 % to 3 .
- 1 % (p less than 0 .
- 0002) , respectively .
- it was lowest for the patients in optimal condition (subgroup ii) in both years , 0 .
- 8 % versus 1 .
- 1 % , and highest for reoperative patients , 5 .
- 3 % versus 4 .
- 3 % .
- in 1981 , 58 % of patients (503 / 870) were in the optimal group compared with 35 % (292 / 828) in 1987 (p less than 0 .
- 0001) .
- the last six years have seen a progressive trend in surgically treating older , sicker patients who have more complex disease , with a significant reduction in the best candidate group .
- (abstract truncated at 400 words) .

Figure A.3: Attention visualization for a randomly selected test article from the OHSUMED dataset. Blue highlight at the start of each sentence indicate its attention weight, while red highlight correspond to each words assigned attention weight.

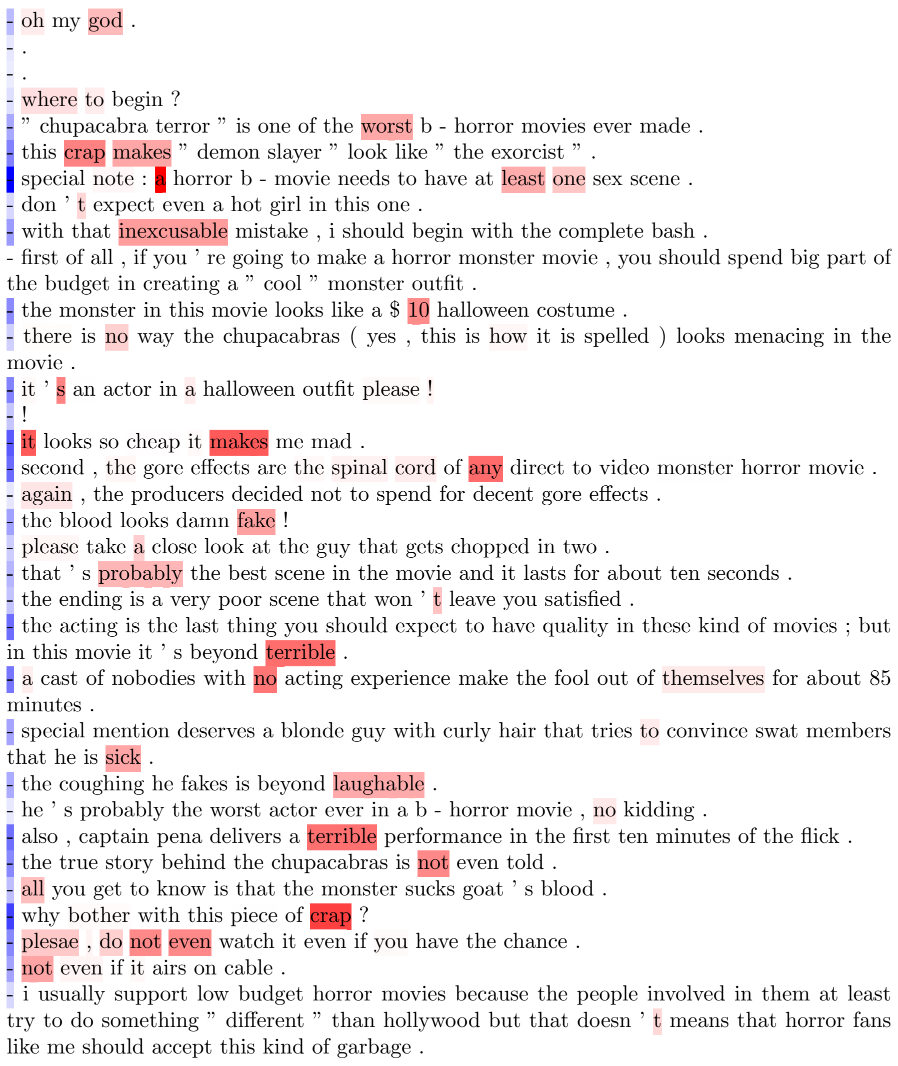

- oh my god .
- .
- .
- where to begin ?
- " chupacabra terror " is one of the worst b - horror movies ever made .
- this crap makes " demon slayer " look like " the exorcist " .
- special note : a horror b - movie needs to have at least one sex scene .
- don ' t expect even a hot girl in this one .
- with that inexcusable mistake , i should begin with the complete bash .
- first of all , if you ' re going to make a horror monster movie , you should spend big part of the budget in creating a " cool " monster outfit .
- the monster in this movie looks like a $ 10 halloween costume .
- there is no way the chupacabras (yes , this is how it is spelled) looks menacing in the movie .
- it ' s an actor in a halloween outfit please !
- !
- it looks so cheap it makes me mad .
- second , the gore effects are the spinal cord of any direct to video monster horror movie .
- again , the producers decided not to spend for decent gore effects .
- the blood looks damn fake !
- please take a close look at the guy that gets chopped in two .
- that ' s probably the best scene in the movie and it lasts for about ten seconds .
- the ending is a very poor scene that won ' t leave you satisfied .
- the acting is the last thing you should expect to have quality in these kind of movies ; but in this movie it ' s beyond terrible .
- a cast of nobodies with no acting experience make the fool out of themselves for about 85 minutes .
- special mention deserves a blonde guy with curly hair that tries to convince swat members that he is sick .
- the coughing he fakes is beyond laughable .
- he ' s probably the worst actor ever in a b - horror movie , no kidding .
- also , captain pena delivers a terrible performance in the first ten minutes of the flick .
- the true story behind the chupacabras is not even told .
- all you get to know is that the monster sucks goat ' s blood .
- why bother with this piece of crap ?
- plesae , do not even watch it even if you have the chance .
- not even if it airs on cable .
- i usually support low budget horror movies because the people involved in them at least try to do something " different " than hollywood but that doesn ' t means that horror fans like me should accept this kind of garbage .

Figure A.4: Attention visualization for a randomly selected test article from the IMDB-long dataset. Blue highlight at the start of each sentence indicate its attention weight, while red highlight correspond to each words assigned attention weight.

TRITA EECS-EX-2019:817