MASTER'S THESIS | LUND UNIVERSITY 2018

A study of hierarchical attention networks for text classification with an emphasis on biased news

Max Söderman

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2018-21

A study of hierarchical attention networks for text classification with an emphasis on biased news

Max Söderman
Max.Soderman@gmail.com

December 2, 2018

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Marcus Klang, marcus.klang@cs.lth.se
Examiner: Jacek Malec, Jacek.Malec@cs.lth.se

Abstract

The last few years the world has seen an increase in fake and biased news in the media and on the internet. Due to the pure amount of articles, the fake news are basically impossible to manually sort out from the rest. Automatic classification algorithms based on machine learning have therefore been suggested.

This study examines different hierarchical attention networks for classifying texts. The main focus has been on biased news but the models have also been tested on other data.

The best classifier for fake news was a 3-level hierarchical attention network with FastText embeddings with a f1-score of 0.96, recall of 0.96 and precision of 0.96.

Keywords: NLP, Hierarchical Attention Network, 3HAN, Text Classification, Machine Learning

Acknowledgements

As customary there are a lot of people to acknowledge and some to unacknowledge. Here follows a short summary of the former.

Firstly, I would like to thank my supervisor Marcus for his help, support and for always being able to explain my page-long error messages. A thank you also goes out to all the teachers I have had during the years. Thanks to the BBC for letting me use their articles in my visualization.

Big thanks to 李志瑛 for helping me patch up the mess I called equations. Another big thanks to Maria for finding the flaws in my English and spending hours discussing "fancy" possible titles. So many great titles gone to waste… "Vectorization of Language: Autodetermine Bullshit Websites Automatically" is a personal favourite.

The most important thanks of course goes out to my family. To my mom, my dad and my brother for always supporting me and encouraging my interest in science. To my girlfriend (do not worry, there is more below) and to Maya, the world's only real "gosedjur". Love to you all.

Finally, a big thanks to my 端端 for always being there for me. For your love and support and for filling my life with happiness. Sorry for ruining the amazing idea you had for your own thesis acknowledgements. Hope this at least makes up for parts of the damage.

Contents

Chapter 1
Introduction

1.1 Background and purpose

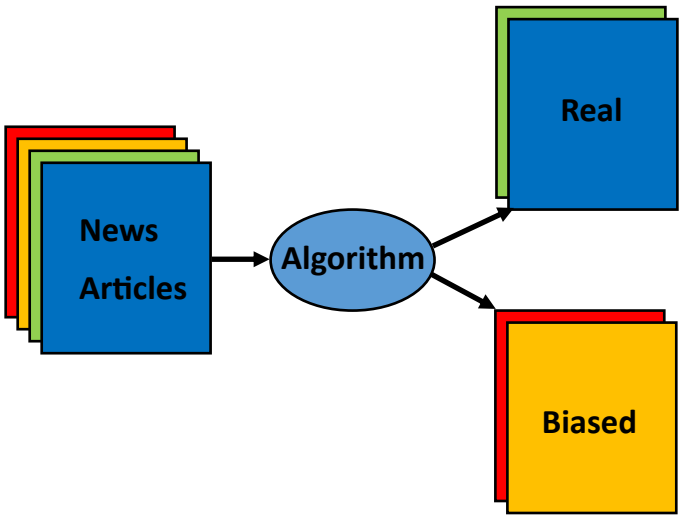

Figure 1.1: A text classification problem where news articles are classified as biased or real

A common task in natural language processing is text classification, sometimes also called text categorization. The task consists of classifying texts based on their content into one of several predefined categories and is described in Figure 1.1. This can be a very time saving application of machine learning since manually classifying big amounts of text can be very time consuming. If the machine learning algorithm is deterministic it also guarantees consistency in the classification as a deployed machine learning algorithm will

classify the same text to the same class every time, while several humans might classify it in different classes.

Text classification has historically been applied to for example research papers, news articles, comments and emails Singhania et al. (2017) Hingmire et al. (2013) Yang et al. (2016). The categories could be different topics for news and research articles, comments containing hate speech or emails containing spam.

Recently, in the wake of the United States presidential election in 2016 and the appearance of fake news a new application of text classification has risen in popularity. Distinguishing fake news from normal news has become a problem for many social networks and news feeds Mihailidis and Viotty (2017). A problem that could potentially be solved with automated text classification.

To solve the problem several methods have been proposed. Some methods such as (Chopra et al., 2017) are based around the idea that comparing the title with the text will give information about the articles authenticity as fake news tend to have titles either unrelated to or exaggerating the content of the articles. Other solutions such as (Yang et al., 2016) are based on classic text classification methods only looking at the content of the article. A combined method was proposed by (Singhania et al., 2017). They suggest a 3-level hierarchical attention network with a structure focusing on each word in a sentence, each sentence in text and finally the headline and the text. The model itself is an expansion of the classical hierarchical attention network model suggested by (Yang et al., 2016).

The objective of this project is to examine and compare the hierarchical attention network and 3-level hierarchical attention network (3HAN) models, primarily on English fake news classification but also on other text classification tasks including a Swedish dataset of speeches from the Swedish parliament (The Swedish Parliament, 2010) and the 20 Newsgroups dataset (Lang, 2008). Hierarchical attention models have to the best of my knowledge not been applied to the Swedish language before. The study also involves looking at potential improvements and variations of the 3HAN model such as different kinds of word embeddings as few such modifications were examined in the original report.

1.2 Contribution

This work aims to give a better understanding of how the 3HAN structure suggested by (Singhania et al., 2017) performs compared to the original hierarchical attention network (Yang et al., 2016). This was done through re-implementing the models. They were then optimized and compared on a news dataset.

In addition to the biased news classification the performance of the hierarchical attention network on a Swedish classification task was tested. This has to the best of my knowledge not been done before and provides new knowledge of the model's possible applications.

1.3 Outline

The Theory chapter introduces the reader to the theory behind text classification with machine learning. It starts by describing basic pre-processing and features used in natural

language processing and then moves on to describing different kinds of machine learning models used in the project.

The method chapter mainly describes the implementation of the models described in the Theory chapter. It also describes how the data sets were processed to make the data compatible with the models as well as which steps were taken to improve the models.

The Results chapter presents the performances that were achieved by the different models on the different data-sets as well as examples of the visualisation.

The Discussion and Conclusion chapters analyses the results and draws conclusions from the work done in the project. They also suggests ideas for further work that can be done on the topic.

Chapter 2
Theory

2.1 Related work

Text classification is a well developed research area within natural language processing. The first mention of a neural network based model with a hierarchical attention structure, described in section 2.9 was made by (Yang et al., 2016). They suggest a two-level attention structure with word attention followed by sentence attention, described in section 2.9.2. The suggested network is a general approach to text classification where the structure of the article is taken into account.

On the topic of fake or biased news classification an approach has been to use LSTMs, described in section 2.6.3 or other methods on the text and headlines using the assumption that the headlines of fake news articles often are over dramatical and less connected to the content of the actual article. By concatenation the text-body with the headline, a LSTM based method using both the text and the headline can classify the news with less connected headlines. This has for example been done by (Chopra et al., 2017).

An article where the previously described methods are somewhat combined was recently published by (Singhania et al., 2017). They suggest using the same hierarchical attention structure as (Yang et al., 2016) but with an extra layer. The extra layer involves a headline-body attention. The idea being that the three level hierarchical attention network (3HAN) can both compare the body and headline as done in (Chopra et al., 2017) and utilise the article structure by the same principle as (Yang et al., 2016). The results look very promising with the authors claiming to have reached a classification accuracy of 96.77%. They have however not published their test-set which raises questions about the models performance.

2.2 Performance measures

The performance measures used are precision, recall and F_1. These are defined using four different concepts, true positive, false positive, true negative and false negative. If we see a biased article as a positive classification a true positive, t_p is when a biased article is defined as biased. A false positive f_p would be when a real article is defined as biased. Similarly, a true negative t_n is when a real article is classified as real and a false negative f_n is when a biased article is classified as real.

Precision is defined as

$$Precision = \frac{t_p}{t_p + f_p} \tag{2.1}$$

and a high precision means that the model has very few false positives for the class. Recall is defined as

$$Recall = \frac{t_p}{t_p + f_n} \tag{2.2}$$

A high recall means that the algorithm miss very few positive values. F_1 is defined as

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.3}$$

The F_1 score is the harmonic mean of precision and recall. A high F_1 means a high score on both the other measures.

2.3 Features

Features are ways of representing the data in a way that machine learning models can interpret. As text can not be directly interpreted mathematically it needs to be transformed. Five different kinds of features were used for the models. Bag-of-words and bigrams for the baseline models, GloVe-embeddings and FastText-embeddings for the English neural network based models and Word2Vec-embeddings for the Swedish model.

2.3.1 Bag-of-words model

A Bag-of-words model is a simple representation for a text. It does not take into consideration the order of words but simply counts the number occurrences of each word. The text is represented by a vector where the dimensions correspond to the amount of unique words in the data-set. Every dimension represents a word and the value represents the number of occurrences in a sentence. For example if we have the two sentences:

I like to watch football

and

I like football but I like tennis as well

we get:

Table 2.1: Bag-of-words vector examples

	I	like	as	watch	to	well	but	football	tennis
Sentence 1	1	1	0	1	1	0	0	1	0
Sentence 2	2	2	1	0	0	1	1	1	1

The Bag-of-words model makes it possible to represent all the texts in the data-set as individual vectors. This gives the possibility to use machine learning algorithms to classify the texts.

2.3.2 Bi-grams

Using bi-grams in the model somewhat take the word order into account. Instead of simply counting the word frequency it counts the frequency of word pairs. For example the sentence:

He is a bad president

becomes:

Table 2.2: The bi-grams for "he is a bad president"

(He, is)	(is, a)	(a, bad)	(bad, president)
1	1	1	1

The advantage is that the order and context of the words are represented. For example "bad president" contains information of what is bad which would not be caught otherwise. The model can also be expanded to a N-gram model where N words are combined in a dimension.

2.3.3 Word embeddings

Word embeddings is a feature where a word is translated into a vector of a given dimension. This vector models the semantics of the word. Something that is not done by the bag-of-words. The process of learning these embeddings can be done in several ways. The vectors are created in such a way as to let the vectors of similar words lay close to each other in the vector space. For example in GloVe embeddings supplied by the authors of the algorithm (Pennington et al., 2014) the word "thesis" has the following closest word vectors with corresponding cosine similarities:

Table 2.3: Closest words to "thesis" and corresponding cosine similarities

dissertation	0.87781
doctoral	0.76769
phd	0.73197
ph.d.	0.69371
doctorate	0.62604

Where the cosine similarity for the two vectors A and B is defined as:

$$\cos(\theta) = \frac{A \cdot B}{||A||||B||} \tag{2.4}$$

Three different algorithms to create embeddings used in this article are described bellow.

Word2Vec embeddings

Word2Vec is an algorithm for obtaining word vectors. It was presented by (Mikolov et al., 2013) and provides vectors similar in performance to GloVe. However the vectors are trained in different ways. Word2vec are trained as a predictive model. Pre-trained vectors for several languages are available around the web.

GloVe embeddings

GloVe - Global Vectors is an unsupervised algorithm for making word vectors. It was presented by (Pennington et al., 2014) and produces vectors for words of a predefined dimension. GloVe differs from Word2vec in that it is a count-based model.

Pre-trained vectors for English are supplied from the projects website. But the algorithm also allows for new vectors to be trained on a data-set giving vectors specially made for the data.

FastText

FastText is an open-source library developed by Facebook AI Research (Bojanowski et al., 2017) for creating word embeddings and classifiers. It differs from GloVe and Word2Vec embeddings since it can not only create embeddings for words in it's vocabulary but also predict an embedding for words not in the vocabulary through sub-word embeddings. This makes it possible for a FastText model to predict a word embedding for any word.

2.4 Baseline models

To be able to make conclusions about the performance of more complex models a baseline needs to be established. A baseline is a simpler model to witch the more advanced model can be compared. If a simpler model performs as well then the more complex model is not a reasonable choice for the problem. Two different models were used to establish a baseline for the classification. A Support Vector Machine and a Logistic Regression based model. These were chosen as they are simple and well examined models that are quick to train.

2.4.1 SVM

Support Vector Machine (SVM) is a classifier based around the idea of finding an optimal boundary between the classes. It selects points from the training data as support vectors

and use them to find a hyperplane that maximizes the boundary. A SVM generally performs well when the amount of features are bigger than the number of training examples. This is often the case for text data. It can classify with based on linear function of the features. It can also be modified to perform a non-linear classification. The hyperplane defining the classification is given by:

$$W^T X + b = 0 \tag{2.5}$$

Where W is the weight vector, b is the bias and X is the input.

2.4.2 Logistic regression

Logistic regression is a linear classifier suggested by (Cox, 1958). It outputs the probability that a data point belongs to a certain class. The classes are usually labelled 0 and 1. But the model can also be expanded to a multinomial logistic regression that can handle more than two classes. The model is fitted to data using one of several possible optimization algorithms. The probability of Y belonging to class 1 is described by:

$$Pr(Y = 1|X) = \frac{1}{1 + e^{-W^T X}} \tag{2.6}$$

Where X is the input and W the weight vector.

2.5 Neural networks

A neural network is basically a network of nodes. The nodes are usually grouped into several layers where one layer known as the input layer receives the data and one layer known as the output layer outputs the result. Usually every node in each layer is connected to every node in the next layer. That means that every node in one layer has every output from the nodes in the previous layer multiplied by a weight for each as input. The node applies a simple function, known as an activation function on the weighted inputs.

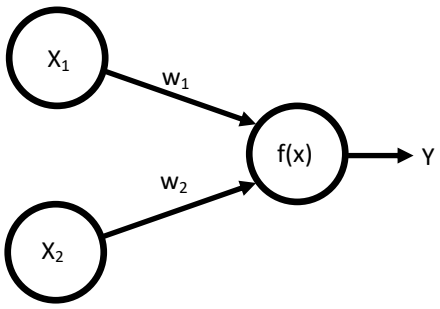

Figure 2.1: A simple neural network with two input nodes and one output node

A simple neural network is shown in Figure 2.1. For this network the output would be:

$$Y = f(x_1 W_1 + x_2 W_2) \tag{2.7}$$

Where the function f is an activation function. For example a sigmoid- or step-function. Which is a simple function with many inputs.

The idea behind the neural network is that it can be "trained" on a specific problem. To fit the network to a problem the weights in the network have to be changed. This is often done by back-propagation. First a forward propagation is made, passing values through the network and producing a result. Then a backward propagation is made. Using the correct output for the given input deltas, a sort of error for each weight is calculated. These are then used to calculate a numerical gradient for each weight. The gradient is then used by an optimizer to make a small change to each weight resulting in a better model.

2.5.1 Training

When training the neural network with the previously described backwards propagation one usually talks about epoch, iteration and batch. An epoch is when an entire training set is passed forwards and backwards through the network once. A whole training set can not be passed through the network at once. So it is divided into a number of batches, that are passed through, one by one. Iterations is the number of batches that are needed for one epoch.

Three other important concepts are training, validation and test data. The training data is the data passed through the network during training. This is the data used for the backwards propagation. The validation data is used to evaluate the training. For example one could test a model on the validation data after one, two and three epochs to see how many epochs gives the best results. The validation data can also be used to test different values of hyper parameters described bellow. This makes the validation data part of the training, which means that the performance of the model on the validation data is not a good measure for how the model will perform on new data. Therefore there i a third dataset, the test data. The test data is not part of the training process. It is only used to evaluate the performance of the final model.

As mentioned earlier an optimizer is used to change the weights of the network using the gradient. These optimizers usually have so called hyper parameters that can be tuned to the specific problem. One example is the learning rate that governs how much the weights are changed. A larger learning rate means a larger change in weights. In this thesis two different optimizers were used, ADAM (Kingma and Ba, 2014) and NADAM (Dozat, 2015).

2.6 Machine learning frameworks

To simplify the implementation of machine learning models, different pre-implemented frameworks are usually used. One of the most common frameworks is Google's TensorFlow. One great advantage with TensorFlow is that it can automatically differentiate the network which means one does not have to calculate the derivatives by hand. To simple the limitation even further other frameworks built on top of TensorFlow can be used. In this thesis Keras was chosen as it greatly simplifies the implementation and allows for

quickly building new models. Keras has a layer-based approach for building networks. For example the previously described GRU is a pre implemented layer in Keras that can be put on top of other layers to create a network. Keras also have several pre implemented optimizers that can be used.

2.6.1 Dropout

Dropout is a regularization technique used in neural networks suggested in (Srivastava et al., 2014). During training several nodes in the network are "dropped" meaning they are removed from the network along with it's connections to other nodes. The nodes that are removed are chosen at random with a given probability, the dropout rate. Dropout often decreases the over-training as it forces the neurons to learn from multiple outputs.

2.6.2 Recurrent neural network

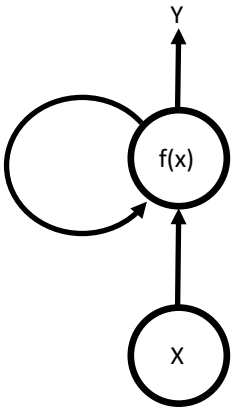

Figure 2.2: A simple recurrent network with one in-node, one out-node and one recurrent loop.

Recurrent neural networks are a subclass of neural networks. They are characterised by the output of the current time-step being dependent on information from previous time-steps. A simple version is shown in Figure 2.2 which has a node that gets input from the previous time-step. This network has a problem. When optimizing the weights the gradient is dependent on the previous time-steps. This can get very computationally complex and can also sometimes results in the vanishing and exploding gradient problems.

2.6.3 Gated Recurrent Unit - GRU

The Gated Recurrent Unit (GRU) was first introduced by (Cho et al., 2014).It is a way of reducing the computational complexity of recurrent networks and avoiding the vanishing and exploding gradient problems. The GRU was presented after the Long Short-term Memory (LSTM) and is a simpler version with only two gates and without an external internal memory. The update gate z and the reset gate r are defined as:

$$z = \sigma(x_t U^z + h_{t-1} W^z + b_z) \tag{2.8}$$

$$r = \sigma(x_t U^r + h_{t-1} W^r + b_r) \tag{2.9}$$

Where U^z, W^z, U^r, W^r are weights, x_t is the input, h_t is the output and σ is the activation function (usually a sigmoid).

The candidate for the output value $\tilde{h}_t$ is determined using the reset gate as:

$$\tilde{h} = \tanh(x_t U^h + (h_{t-1} r) W^h) \tag{2.10}$$

The output value is then determined by filtering with the update gate. Bigger z gives focus on the new value:

$$h_t = (1 - z)\tilde{h} + z h_{t-1} \tag{2.11}$$

2.7 Notations

In the following sections more complex models are described and the notations are therefore described here to make it easier to understand.

Given an article with a headline and a body each word in the body is denoted w_{it}, the t:th word in the i:th sentence and the words embedding is denoted X_{it}. In the same way a GRU going to the right starting from sentence i is denoted $\overrightarrow{h_i}$. A bidirectional GRU starting from sentence i becomes $h_i = \left[\overleftarrow{h_i}, \overrightarrow{h_i}\right]$, where $\overleftarrow{h_i}$ and $\overrightarrow{h_i}$ are concatenated. Finally the i:th word in the headline is denoted y_i and a bidirectional GRU over headline word i is denoted h_i^3. The 3 is to describe the 3 hierarchical level of the network as suggested in (Singhania et al., 2017).

2.8 Attention GRU

Attention is a neural network design that allows the network to focus on a specific feature and then use other features in the context of that feature. The attention GRU presented here is one of the more simple versions. It involves the network focusing the attention on each word and then running a GRU backwards and forwards from that word. This makes it possible for the model to look at each word with the context for the other words. It then allows the model to focus on the important words, giving them a higher weight value. The model is described mathematically below.

2.8.1 Word Encoder

If we in a sentence s_i have T_i words each word can be denoted w_{it}. Using an embedding matrix W_e we can embed the words into vectors:

$$x_{it} = W_e w_{it}, t \in [1, T] \tag{2.12}$$

If we denote a forward GRU, $\overrightarrow{GRU}$ we can write a forward and a backward GRU for each word as:

$$\overrightarrow{h_{it}} = \overrightarrow{GRU}(x_{it}), t \in [1, T] \tag{2.13}$$

$$\overleftarrow{h_{it}} = \overleftarrow{GRU}(x_{it}), t \in [1, T] \tag{2.14}$$

We can then concatenate the two GRU values as:

$$h_{ij} = \left[\overrightarrow{h_{ij}}, \overleftarrow{h_{ij}}\right] \tag{2.15}$$

2.8.2 Word Attention

To get more focus on the relevant words we add the attention mechanism given by the following three equations:

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{2.16}$$

$$\alpha_{it} = \frac{e^{u_{it}^T u_w}}{\sum_t e^{u_{it}^T y_w}} \tag{2.17}$$

$$s_i = \sum_t = \alpha_{it} h_{it} \tag{2.18}$$

The first equation is basically the word annotation h_{ij} being fed through a single layer perceptron. How similar the resulting values are to the word level relevance vector u_w gives the attention weights α_{ij}. We then get the sentence encoding through using the attention weights to sum the word annotations.

2.9 Hierarchical Attention Network

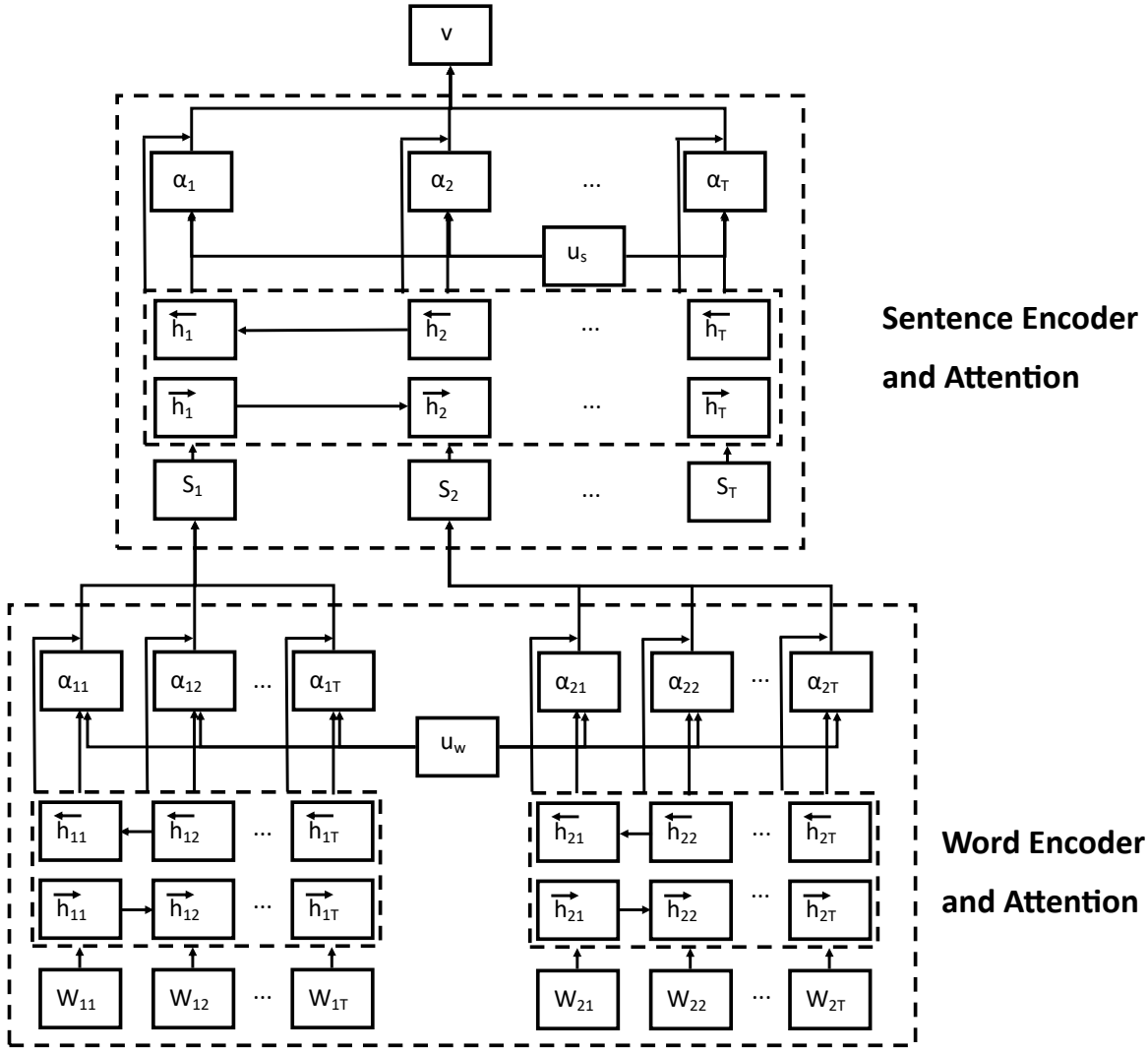

Figure 2.3: The hierarchical attention network where v is the final encoding of the text, the document vector.

The Hierarchical Neural Network (HAN) was presented by (Yang et al., 2016) and builds on the same idea as the Attention GRU but also applies attention to each sentence. So just as every word in each sentence is put in context of the other words and gets a weight value depending on its importance for the classification each sentence encoding is also put into context and gets a weight. This then results in a text encoding.

2.9.1 Sentence Encoder

Applying a GRU to the sentence encoding s_i we get:

$$\overrightarrow{h_i} = \overrightarrow{GRU}(s_i), i \in [1, L] \tag{2.19}$$

$$\overleftarrow{h_i} = \overleftarrow{GRU}(s_i), i \in [1, L] \tag{2.20}$$

Where L is the number of sentences in the text. If we concatenate $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ we get $h_i = \left[\overrightarrow{h_i}, \overleftarrow{h_i}\right]$.

2.9.2 Sentence Attention

Using the same attention mechanism as for words we get:

$$u_i = tanh(W_w h_i + b_w) \tag{2.21}$$

$$\alpha_i = \frac{e^{u_i^T u_w}}{\sum_t e^{u_i^T y_w}} \tag{2.22}$$

$$v = \sum_t = \alpha_i h_i \tag{2.23}$$

Here v is a document vector that contains the information for each sentence. A simple illustration of the hierarchical attention network where the creation of the attention weights is left out is shown in Figure 2.3.

2.10 3HAN

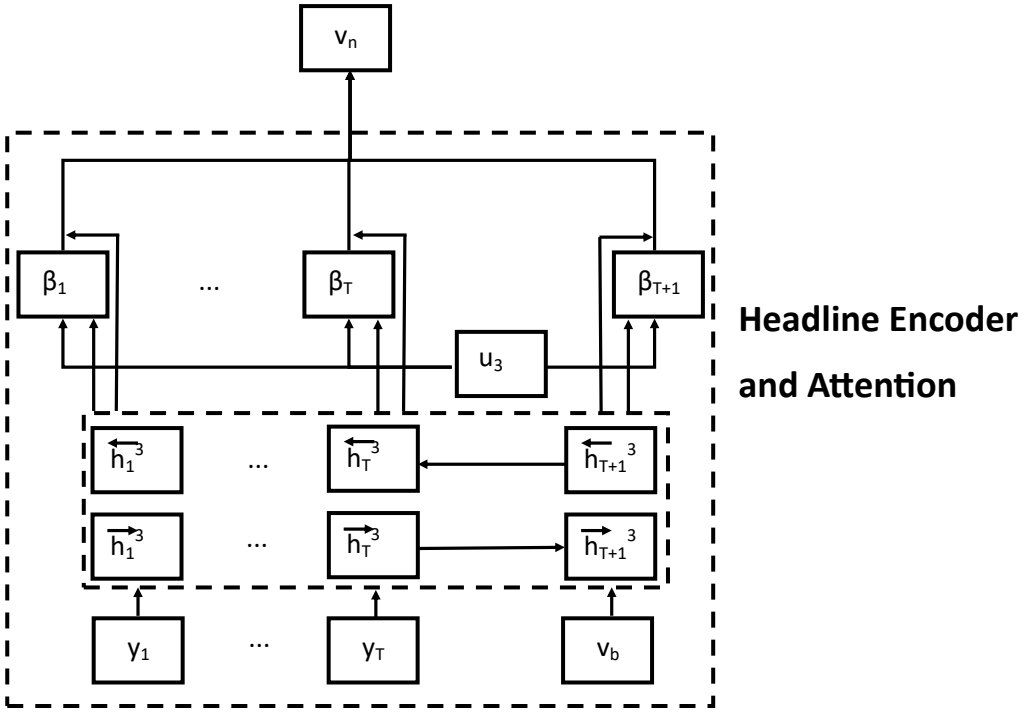

Figure 2.4: The 3HAN network where v_b annotate the body encoding from figure 2.3.

The 3 level Hierarchical Neural Network (3HAN) was presented by (Singhania et al., 2017) and offers a new take on the classical HAN model tailored for detecting fake news. Since one of the main features of a fake news article often is a headline that exaggerates or is not related to the content of the article the authors suggest a separate attention structure for the headline. Each word gets an attention weight and the body encoding is appended to the end of the sentence as an extra word.

2.10.1 Headline Encoder

Denoting the k words in the headline w_{01} through w_{0k} and concatenating the text body encoding v onto the word vector as seen in Figure 2.4 we get the word embedding:

$$y_i = W_e(w_{0i}) \tag{2.24}$$

Running a bidirectional GRU over the words and body encoding gives:

$$\overrightarrow{h_i}^3 = \overrightarrow{GRU}(y_j), j \in [1, i] \tag{2.25}$$

$$\overleftarrow{h_i}^3 = \overleftarrow{GRU}(y_j), j \in [1, i] \tag{2.26}$$

That can be written:

$$h_i^3 = \left[\overrightarrow{h_i^3}, \overleftarrow{h_i^3}\right] \tag{2.27}$$

Where the 3 in the above equations denotes that this is the third layer of the hierarchical network.

2.10.2 Headline Attention

Similar to the previous attention mechanisms the news vector v_n is given by:

$$u_i = \tanh(W_3 h_i^3 + b_3) \tag{2.28}$$

$$\beta_i = \frac{exp(u_i^T u_3)}{\sum_i exp(u_i^T u_3)} \tag{2.29}$$

$$v_n = \sum_i \beta_i h_i^3 \tag{2.30}$$

An illustration of the headline encoder from the body encoding v to the news vector v_n can be seen in Figure 2.4.

Chapter 3
Method

3.1 Methodology

As a first step both the hierarchical attention and the 3HAN model are implemented as TensorFlow backed Keras models in Python. Most of the models are built from basic Keras layers but the attention mechanism is implemented as a separate custom made layer.

The two models are compared with each other as well as several baseline models such as Support vector machine and logistic regression as well as simpler neural network based models on a dataset consisting of news from biased and trusted sources. To compare the performance precision, recall and F1-score are used.

Several different embeddings such as GloVe and FastText with different embedding dimensions as well as different training data are tested for the 3HAN model to see if the performance can be improved compared to the model with 300-dimensional GloVe embeddings proposed in the original article.

The hierarchical attention model is tested on the already well examined 20 Newsgroups dataset (Lang, 2008) to get a result that can be compared with other state of the art models. This dataset is not what the model is intended for but the comparison might still give a hint of the models performance compared to other models.

The hierarchical attention model is also applied to a Swedish dataset consisting of political speeches from the Swedish parliament (The Swedish Parliament, 2010) to examine if its performance can be generalized to other languages and types of classification tasks. For this model several different set-ups are examined to improve the performance, for example different embeddings, embedding dimensions and different amounts of training data.

To properly compare the hierarchical attention model and the 3HAN model a way of visualising the attention is implemented. This allows for comparison of what the models focus on when making a classification, both at a word, sentence and headline level. The implementation is done in Flask (Ronacher et al., 2010) as a web page with the Python model in the background.

3.2 Datasets and pre-processing

3.2.1 Biased and normal news dataset

The data set is put together from two different sources. The (Corney et al., 2016) and the Kaggle (Risdal, 2016) data set. The sources were checked so that the news are only from active websites claiming to publish "news". The "real" news are taken from several news websites. The original dataset contains a million articles and from these I have randomly sampled the same amount of articles as are in the fake news set. The same amount as the number of usable articles from the biased news dataset.

The Fake News dataset from Kaggle originates from the BS detector (Sieradski, 2017) and contains several labels such as fake news, bs and junksci. These were merged into one class labelled "biased". This is of course not a perfect way of creating a data set. Preferably all articles should have been fake news which might make the results more useful. The line between fake news and the other classes in the original dataset is however slim and the results from the combined dataset might still be useful as a comparison.

A first step for the fake news dataset was to go through all sources and check so they actually contained "news". Some of the sources were micro-blogs with short opinions and didn't publish articles. These were removed from the dataset along with sites that were no longer online as the content of these sites were uncertain.

The dataset with real news contained one million articles, way too much data. So an equal amount of data to the fake news was sampled. As there is no guarante that the dataset does not contain fake news only articles from big media houses such as CNN, The Guardian, etc were used.

3.2.2 News classification example

The texts that should be classified in the biased news dataset are news articles or articles claiming to be news articles from several different sources. Here follows parts of two examples with headline and text as they are provided in the dataset:

Headline:
'Report: Eating Raw Weed Prevents Bowel Cancer, Fibromyalgia and Neuro-degenerative Diseases'

Text:
"Cannabis is taken in different forms all over the world, while being primarily smoked, being eaten raw has been proven to provide incredible health benefits, whilst also being non-psychoactive, therefore more appealing to a wider range of people. Marijuana can be described as the new range of superfoods, it contains over 400 chemical compounds containing beneficial vitamins, essential oils, and acids."

Headline: 'XPrize's $20m carbon recycling award aims to cut fossil fuel emissions'

Text: 'The idea of capturing carbon emissions and turning it into something valuable

has long intrigued scientists, businesses, politicians and environmentalists alike. But it's never proven economically viable. Could the XPrize change that? Given the threat of climate change, what should the world do with its reserves of fossil fuels?"

The first article is from the website "humansarefree.com" (Humans are Free, 2016), a pseudo science website publishing health advice. The second is an article from the Guardian "guardian.co.uk" (Gunther, 2015), a renounced media source based in the UK. The task consists of classifying articles like these. The first should be classified as biased while the second should be classified as "normal" news.

3.2.3 Biased and normal news pre-processing

The data was formatted into three columns, headline, text body and class. The classes were labelled 0 and 1. The texts and headlines were then processed with Beautiful Soup (Richardson, 2014) to remove HTML tags. The stopwords were removed and all upper-case letters were replaced with lower-case letters. Finally the texts and headlines were tokenized using a tokenizer with the most common 200000 words trained on the training data. Depending on the model they were either tokenized into a sequence of words or into a sequence of sentences that consisted of sequences of words. Rows that after the text pre-processing were missing either text body or headline were removed. The texts were then split into half resulting in one training and one testing dataset.

Due to the fact that the input data always has to have the same dimensions, a maximum amount of words in a sentence and a maximum amount of sentences in a document have to be chosen. Should a sentence be longer the words will be cut away. Should it be shorter the remaining word-places will be filled with zeros. Most models have been of the dimension 100 sentences and 30 words per sentence. But some bigger and smaller dimensions have also been tested.

One text inputed into the model with 100 sentences and 5 words per sentence will have the dimension (100, 5). One row could for example look like:

$$[1964 \quad 9371 \quad 158936 \quad 567 \quad 3] \tag{3.1}$$

If the sentence only had 4 words.

3.2.4 Swedish dataset and pre-processing

The Swedish data set consists of speeches made in the Swedish parliament. They are downloaded from the Swedish government's website (The Swedish Parliament, 2010). The last ten years of speeches were used. The data was divided into eight different classes, one for each party, V, S, MP, C, L, M, KD, SD. The classification task consists of determining to which party the speaker belongs. These Speeches have no headlines so they can only be used for the models that only uses the text body.

The Swedish data was formatted into two columns where the first contains the text of the speech and the second contains the one hot encoded classes. This means that a speech from V gets a vector that looks like:

$$[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \tag{3.2}$$

and a vector for a speaker from L looks like:

$$[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \tag{3.3}$$

The text data was processed in the same way as the news dataset.

3.2.5 20 Newsgroups dataset and pre-processing

The 20 Newsgroups dataset contains newsgroups posts of 20 different subjects. They were divided into 4 groups, comp, politics, news and religion as done in (Hingmire et al., 2013) to allow for comparison of the results. It was pre-processed in the same way as the previous datasets. The different categories were represented in the same way as the different parties for the Swedish dataset.

3.2.6 Embedding layer

The tokenizer was as mentioned used to find the 200000 most common words in the training data. This was done to resemble a human vocabulary. These words were then used to create a embedding layer for the model together with several different types of embedding models such as GloVe and FastText. The embedding layer maps an index of a certain word to an embedding for that word. Using the layer one can input a sequence of word-indexes into the model and these will then be transformed into word embeddings.

3.3 Output

The target values for the neural network based models are chosen to be two values of the form $[0, 1]$ for a real news article and $[1, 0]$ for a biased news article. Another choice would be to use a single digit with either 1 or 0 for the different classes. Two values were chosen as it sometimes gives better results even if it should theoretically give the same results.

To get an output of the desired form every neural network based model has a dense layer as the last layer. The dense layer uses a single layer perceptron to dense the input down to two values. The dense layer have softmax activation functions.

3.4 Baseline

The first step of the project was to establish a baseline for the biased news classification. Using the sklearn package a Naive Bayes model, a Logistic Regression model and a SVM was implemented in Python. These are quite simple and quick models and the best performing one was used as a baseline for the more complicated neural network based models. A grid search was used to find the best features. Either just words or bi-grams.

3.5 Training method

3.6 Optimization

Most of the neural network models in this section uses ADAM for optimization. The only exception is the Swedish dataset and the 20 Newsgroups dataset where the NADAM optimizer was used in addition to ADAM. The NADAM optimizer was only used with standard hyperparameters but for the ADAM where they were changed as described below.

3.6.1 Hyperparameters

As mentioned in the theory there are several parameters that can be tuned when training a model. Due to the growing parameter space the tuning was limited to a few parameters, learning rate, number of epochs and dropout rate. The other parameters involved were found to give very little effect on the training and were frozen to limit the parameter space.

The tuning was in the beginning not done in a systematic way. This method quickly gave way to a systematic grid search, usually running over night. This was mostly due to the increasing time required per epoch. In the grid-search the learning rate ranged between 1 and 0.0001. The dropout ranged from 0 to 0.9. Usually a broader search was made and then a more specific closer to the best values of the first search. The best learning rate differed a lot between models but the dropout often ranged from 0.15-0.35.

3.6.2 Training, validation and test data

To evaluate what combination of parameters gave the best model the validation data was used. Usually one would train until the validation performance has reached a maxima and starts decreasing. For this dataset however this was not a useful method. The validation performance would usually not reach a maxima but would instead start oscillating around a value. This made it hard to simply look for the best validation performance as this could still give a lot of over-training. Instead the best way was to look at the gradient of validation and training loss. In the beginning the loss would decrease in big steps. Then the gradient became less steep and the loss decreased slower. This was the best point to stop the training as the small adjustments most of the time only caused over-training.

When the models had been trained with the above described methodology they were evaluated on the test dataset. This gives an approximation of how the models perform in general, on new data.

3.6.3 Oversampling

Since the biased news dataset was rather small a problem during training might be lack of data. To solve this problem more real news articles were sampled from the real news dataset. To avoid unbalance the biased news were oversampled to give a better distribution between the classes. The proportions used was four times more real news and as before approximately the same amount of real and biased news.

As the above mentioned approach results in the fake articles being greatly oversampled another approach was also tested to avoid overtraining on the fake data. The extra real news were added as before but the fake news were kept without oversampling resulting in an uneven training set which was used for training.

3.7 Simple neural network based models

3.7.1 GRU

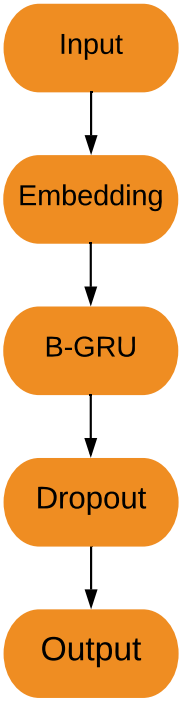

Figure 3.1: The structure of the GRU model, where B-GRU is a bidirectional GRU

The first neural network-based model implemented was a simple GRU. It was implemented in Python using Keras with a Tensor-flow back-end. The structure can be seen in Figure 3.1. The model consisted of an input layer, an embedding layer, a bidirectional GRU without sequential return and finally a dense layer.

The model was primarily trained with GloVe embeddings with 300 dimensions but also smaller dimensions. Several modifications to the model were also tested. For example adding more GRU layers of different width and using dropout layers in between.

This model does not take the sentence structure into account but will instead receive a one dimensional input consisting of all the words. So if the model takes 3000 words it will receive a vector with 3000 values. The embedding layer will then transform it into (3000, 300) if 300 dimensional embeddings are used. Then the bidirectional GRU returns

the depth chosen for the GRU times two as it is bidirectional. Finally the dense layer will return 2 values.

3.7.2 Attention layer

The original 3HAN article (Singhania et al., 2017) used a Theano backed attention layer. To be able to use TensorBoard for analysing the model and because Theano is not being developed anymore, an attention layer was implemented with a TensorFlow backend. Most of the code was inspired by the attention layer developed by (Baziotis, 2017).

The layer takes the bidirectional GRU values and calculates the encoding vector either for a sentence, a text or a headline. It returns the results but also has an option to return the attention weights as well. This is to allow for a deeper analysis of how the models work.

3.7.3 Attention GRU

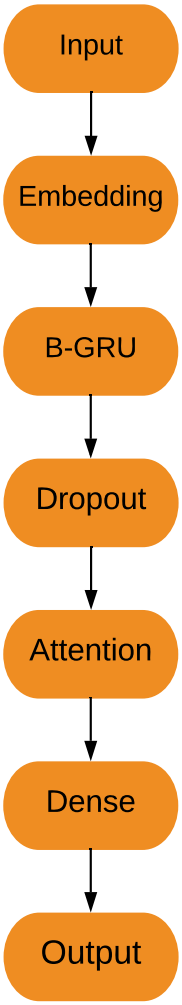

Figure 3.2: The structure of the Attention GRU model, where B-GRU is a bidirectional GRU

The structure of the Attention GRU model can be seen in Figure 3.2 and was built with an input layer, an embedding layer, a bidirectional GRU layer, a dropout layer and then the attention layer on top of the GRU layer. The bidirectional GRU here outputs not only the last value but all values along the way giving the h_{ij}. These values are then densed down to two output values with a dense layer. The two output values represent the two classes "normal" and "biased" where the predicted class has a 1 and the other class has a 0. Several trainings are done for the model all using 300 dimensional GloVe embeddings but with different GRU breadth.

Similar to the GRU model the attention GRU does not take the sentences into account. The difference comes after the bidirectional GRU. The GRU is set to return sequences. This means that the GRU will return (3000, 600) and the attention layer will give a vector with 600 values. This is finally passed through the dense layer to give 2 values.

3.8 Hierarchical attention network

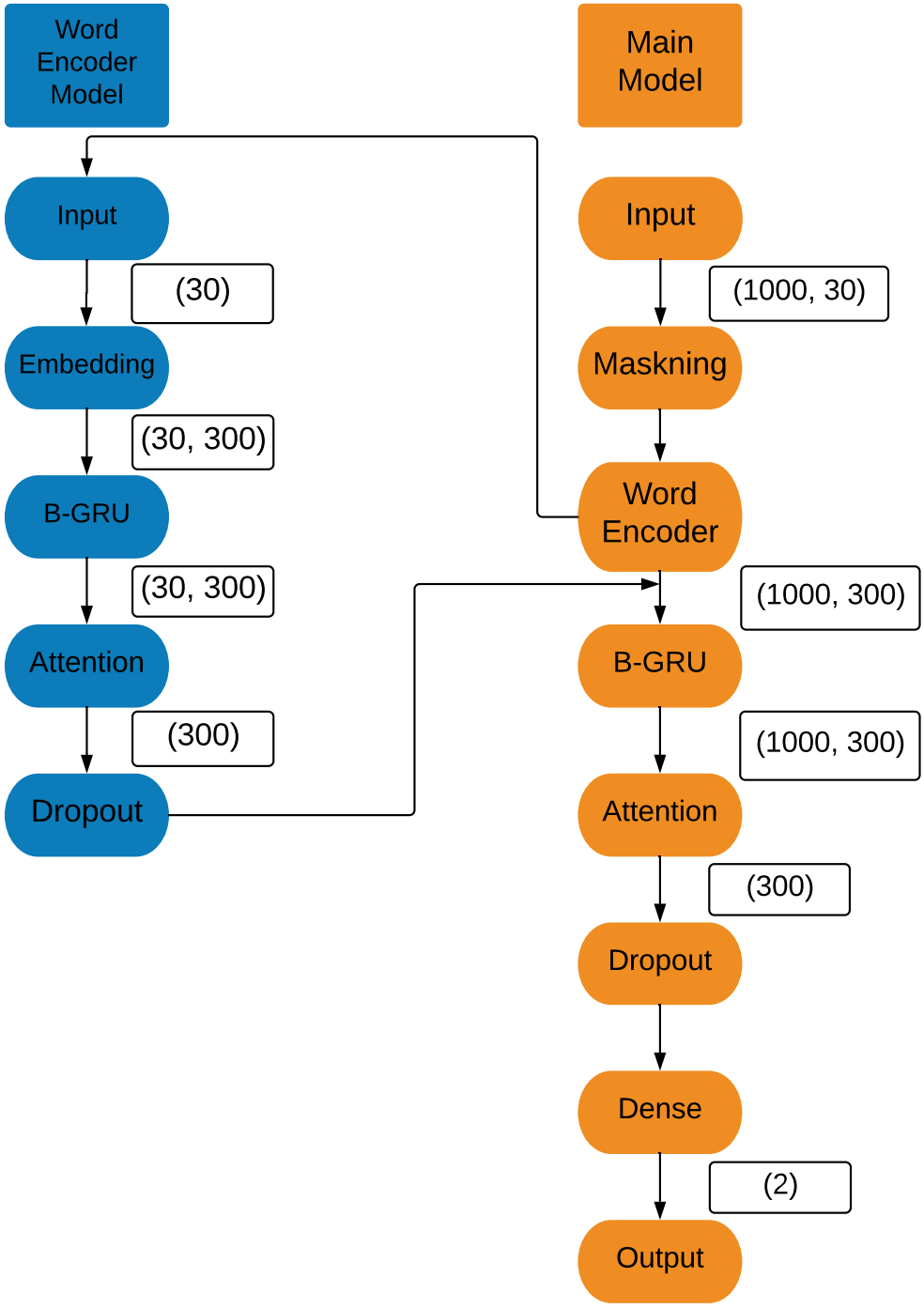

Figure 3.3: The structure of the hierarchical attention network, where B-GRU is a bidirectional GRU

The hierarchical attention model can be seen in Figure 3.3. It is implemented with a nested model structure. This means that one model is used inside the other model. The nested

model is the word encoder. It consists of an input layer, an embedding layer, a bidirectional GRU and an attention layer.

The next model has an input layer where the actual sequential data is inputed. It then has a masking layer to speed up the calculations. Then the nested word encoder is used. It gives the S_i for each sentence. After the nested model a bidirectional GRU and then an attention layer is placed. This finishes of the model giving the text encoding that is fed to the final dense layer.

The model was trained on 300 dimensional GloVe embeddings. The width of the GRUs was not variated as it is locked to half the dimension of the word embedding as described earlier.

The hierarchical attention model receives an input of the dimension (100, 30) for a text of 100 sentences and 30 words per sentence. The dimensions of the following layers output can be seen in Figure 3.3.

3.8.1 Swedish speeches classification using HAN

To test the hierarchical attention model on a Swedish dataset the only modification that has to be done is to the embeddings. As the dataset described in (The Swedish Parliament, 2010) does not contain any headlines but just the content of the speech the classical hierarchical attention network is used instead of the 3HAN model.

The dataset itself is a more difficult classification task than the biased news dataset previously examined. Since it's a 8 class problem the difficulty is higher. Therefore more effort was put into optimizing the model. Two different embeddings were examined, Swevec (Fallgren et al., 2016) and FastTexts Swedish embeddings. As with the previous models ADAM was used for the optimization with different learning rates. In addition the optimiser NADAM was tried. It is similar to ADAM but uses Nesterov momentum.

3.8.2 20 Newsgroups

The 20 Newsgroups is tested to give results that can be compared to other articles. The classification was done with the hierarchical attention network as the 3HAN requires headlines. Both FastText and GloVe embeddings were tried as well as NADAM and ADAM optimizers.

3.9 3HAN

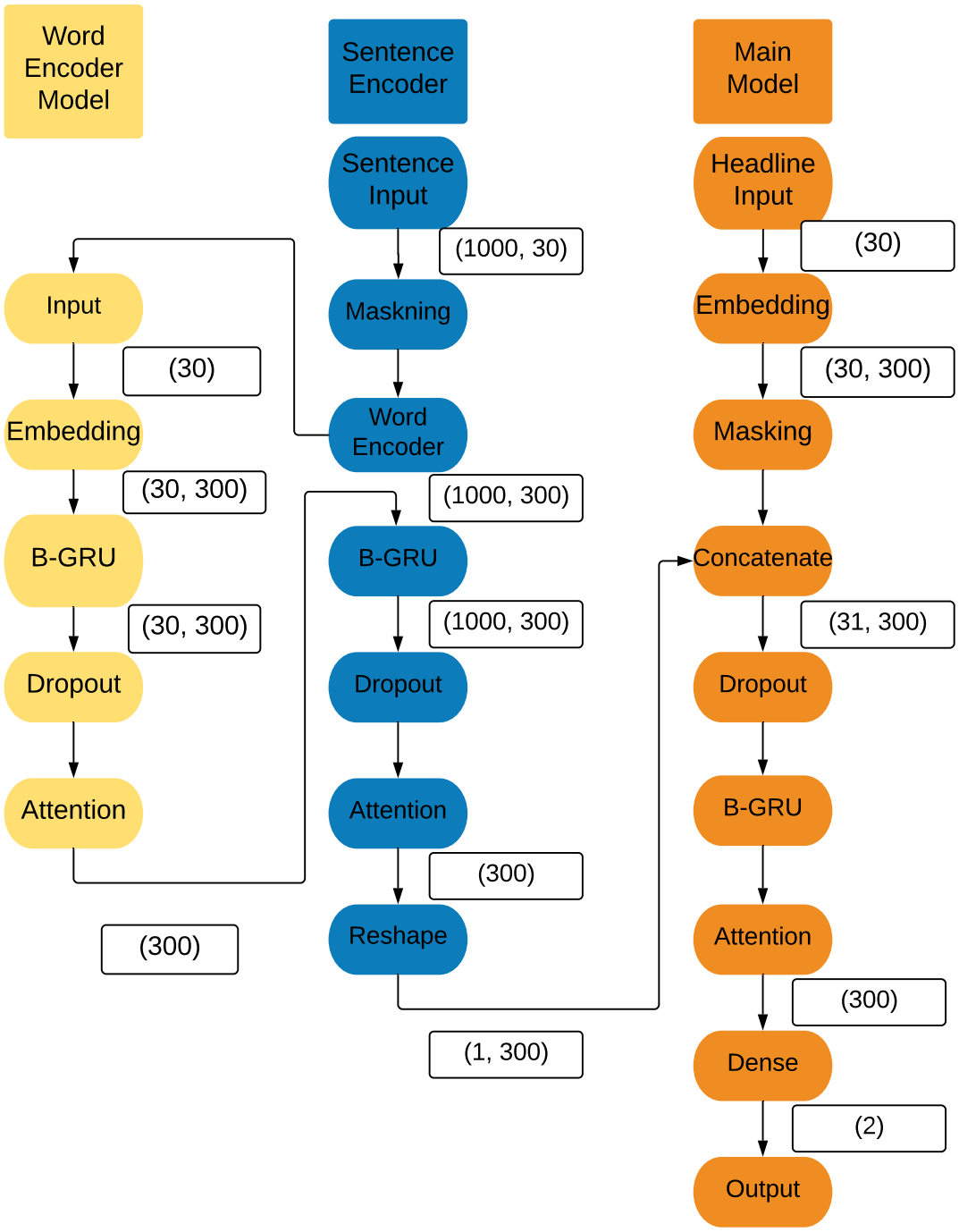

Figure 3.4: The structure of the 3HAN model, where B-GRU is a bidirectional GRU

The 3HAN model is implemented in the same way as the hierarchical attention model with some additions. It's structure is described in Figure 3.4. Instead of the last dense layer there is a reshape layer. This is followed by an input layer for the headline and another embedding

layer. In these layers the value from the reshape is not used, only the new input. The same is true for the next layer, a masking layer. Then the output from the masking layer and the output from the reshape are concatenated together and fed into a bidirectional GRU. This is followed by a dropout layer and the final attention layer. The model is finished of with the standard dense layer. The model was trained with 300 dimensional GloVe embeddings.

The 3HAN model takes an input of the shape ((100,30), 30). This is the same 100 sentences and 30 words per sentence as the hierarchical attention network. In addition it has a 30 dimensional vector containing the headline.

3.9.1 Pre-trained 3HAN

In their paper (Singhania et al., 2017) gets the best performance when first pre-training the word encoder of their 3HAN model on the headlines of the training data. This was done by training the word encoder on the headline data. On top of the word encoder, a dense layer is placed, with the same output as the 3HAN network. This is then trained as the same classification problem but with only the headline as input.

This method was also examined on the biased news dataset. It was simple to implement in Keras as the word encoder already is a nested model. The word encoder was thus trained on the headlines and the weights were then loaded into the whole 3HAN model that was trained as before.

3.9.2 Embedding improvements

To try to improve the performance of the 3HAN model different kind of embeddings were tried with the model to see what gives the best performance. The original article only used 300 dimensional GloVe embeddings. As a greater number of dimensions of the embeddings are not always a guarantee for better results smaller dimensional embeddings were tested, namely 50, 100, 150 and 200 dimensional GloVe embeddings all taken from (Pennington et al., 2014). As noticed before this also means changing the depth of the GRUs from 150 to 25, 50, 75 and 100.

In addition to different dimensions 300 dimensional embeddings with randomly initiated values were tried. This means that the embeddings are trained together with the model on the training data so that all the weights in the embedding layer are also changed during training. Another way of fitting the embeddings to the data is to first train the embeddings to the data and then use them in the model. This was also tested, the embeddings were trained using Gensim as Word2Vec embeddings. These were then used to train the model.

A recently published kind of embeddings are Facebooks FastText embeddings. These have the advantage of being able to predict an embedding for every word, not only the ones in the embeddings vocabulary. Two different approaches were tried when using FastText embeddings. Firstly they were used in the same way as other embeddings, simply to build the embedding layer. Secondly another approach was tried. Theoretically the model can be adapted to input the data already processed with the embeddings. Then no embedding layer is required. This way all the words are replaced with embeddings, not just the most common ones. This approach was tried but sadly the model required too much modification for it to work. Instead a work around was implemented. Using a bigger embedding layer and building it from both the training and testing data basically all words will be

included and it will give approximately the same result as inputing the data already processed with the embeddings. It is important however to point out that the embeddings are locked during the training and therefore are not influenced.

3.10 Visualization

As the attention weights in the hierarchical attention models are tied to specific words and sentences the models allow for a visualization of the underlying "focus" of the network. The visualization then enables further analysis of the models.

To build a visualization model for the hierarchical attention network and the 3HAN the flask (Ronacher et al., 2010) framework was used as it allows for easily incorporating the Python based Keras models. The visualization was essentially designed as a web page with an input box for the text on the hierarchical attention network version and two input boxes for the 3HAN version, one for the headline and one for the text. A simple "Analyse" button then starts the same pre-processing and models as used for the biased news classification described before. To allow for simple visualization reduced models with only 10 sentences and 10 words per sentence were used.

To be able to visualize the attention the attention weights α and β are needed. These can be easily outputted from the Keras model by creating intermediate outputs for the different attention layers. As the α values are basically the importance of each word and sentence for the models decision, visualizing these give a way to see what the model deems important features.

The word attention was visualized as a 10 times 10 matrix. Every cell contains a word and the background colour corresponds to the attention weight for that word. Similarly the sentence attention is visualized with a 10 times 1 matrix to the side of the word attention matrix. Here every cell only contains a colour, also corresponding to the particular sentences attention weight. Finally, for the 3HAN model an attention matrix is also made for the headline. The first 10 cells contain the words of the headline colour-coded with their respective attention weight. The 11th and last cell contains the text body attention and therefore has the text "Body".

To implement the matrices the package SeaBorn's heatmap was used. It allows for colour-coding the matrices and adding texts in the cells as annotations. The figures are then outputed to a new page that also displays the result of the analysis.

Chapter 4
Results

In the following sections the best performances from the different models are presented. The length of the data were 30 words per sentence and 100 sentences for the hierarchical models and 3000 words for the simpler as this was found to be the length giving the best time performance without affecting the results. The metrics used are the metrics described in chapter 2.2. In the last section an example of the visualisation model is presented.

4.1 Real-Biased news classification

The table below shows the results for the real-biased news dataset. Only the best performing models in each category are shown.

Table 4.1: Results for the biased news dataset with 10496 articles support

Model	Precision	Recall	F1-score
LogReg	0.86	0.85	0.85
SVM	0.88	0.87	0.87
GRU 300	0.87	0.86	0.86
Attention 100	0.88	0.88	0.88
Attention 200	0.89	0.89	0.89
Attention 300	0.91	0.91	0.91
Hierarchical 50	0.95	0.95	0.95
Hierarchical 100	0.95	0.94	0.94
3HAN 300	0.94	0.94	0.94
3HAN 50	0.91	0.91	0.91
3HAN 100	0.94	0.94	0.94
3HAN 200	0.92	0.92	0.92
3HAN 300 Trainable	0.91	0.91	0.91
3HAN Random 300	0.84	0.84	0.84
3HAN Word2Vec trained on data	0.93	0.93	0.93
3HAN 300 pre-train	0.95	0.95	0.95
3HAN FastText train	0.96	0.96	0.96
3HAN FastText train and test	0.96	0.96	0.96
3HAN Oversampled 7:7	0.93	0.93	0.93
3HAN 5 times real	0.86	0.83	0.83

The two best performing baseline models are presented in the first section of table 4.1. They are not limited to 3000 words but are trained and tested on the whole articles. Both performed better without bi-grams and those results are the ones presented.

The second section of table 4.1 contains the GRU and attention GRU models. They are trained and tested with articles limited to 3000 words. For the GRU only the best model is presented while several are presented for the attention GRU to show how the depth of the GRU influences the results.

The hierarchical attention network models results are shown in the third section of table 4.1. It was as mentioned trained and tested with 100 sentences and 30 words per sentence. The 300 dimensional GloVe embeddings were used and different depths were tried. However increased depth did not influence the result and therefore only the two lowest depths are presented.

In the fourth section of table 4.1 the 3HAN model, trained with 100 sentences and 10 words per sentence, is presented. 300 dimensional GloVe embeddings are used. In the subsections different modifications are tested to increase the performance of the model. The 300 dimensional model is followed by models with different embedding dimensions. The 3HAN 300 trainable uses the same 300 dimensional models that are tested earlier but this one has used a trainable embedding layer so that the embeddings are changed during training. The final three models in section 4 are the 3HAN with randomly initialized trainable embeddings, the 3HAN with 300 dimensional Word2vec embeddings trained on the data and the 3HAN model with the word encoder pre-trained on the headlines.

The fifth section of table 4.1 contains the 3HAN with 300 dimensional FastTest embeddings. The first one has the tokenizer only trained on the training data while the second one is also trained on the training data to simulate the performance if every word is replaced with an embedding.

The final section is dedicated to the models with extra real news data and oversampled biased news. Several different oversampled models were tried with the extra real news data. The listed two models, one with equally sampled and oversampled data with a 7:7 ratio and one with 5 times more real data but no oversampling. Both are the best performing in their respective categories.

4.2 Swedish Parliament Speeches classification

For the Swedish dataset two kinds of embeddings were tested. The SweVec and the FastTest embeddings. Below follows both kinds trained with NADAM as well as the best performing model trained with ADAM.

Table 4.2: Hierarchical attention network using 300 dimensional SweVec embeddings and trained with NADAM

	Precision	**Recall**	**F1-score**	**Support**
V	0.81	0.40	0.54	1882
S	0.83	0.64	0.72	8857
MP	0.79	0.19	0.30	3053
L	0.81	0.29	0.43	1627
C	0.88	0.21	0.35	1766
M	0.55	0.75	0.64	5134
KD	0.56	0.43	0.49	1421
SD	0.69	0.37	0.48	2052
avg / total	0.74	0.51	0.57	25792

Table 4.3: Hierarchical attention network using 300 dimensional FastText embeddings and trained with NADAM

	Precision	**Recall**	**F1-score**	**Support**
V	0.79	0.56	0.66	1882
S	0.74	0.87	0.79	8857
MP	0.80	0.35	0.49	3053
L	0.90	0.36	0.52	1627
C	0.92	0.31	0.46	1766
M	0.70	0.69	0.70	5134
KD	0.91	0.35	0.50	1421
SD	0.63	0.59	0.61	2052
avg / total	0.76	0.63	0.66	25792

Table 4.4: Hierarchical attention network using 300 dimensional FastText embeddings and trained with ADAM

	Precision	**Recall**	**F1-score**	**Support**
V	0.55	0.66	0.60	1882
S	0.76	0.80	0.78	8857
MP	0.60	0.51	0.55	3053
L	0.73	0.36	0.48	1627
C	0.56	0.51	0.53	1766
M	0.66	0.69	0.67	5134
KD	0.70	0.41	0.52	1421
SD	0.66	0.52	0.58	2052
avg / total	0.68	0.64	0.65	25792

4.3 20 Newsgroups

Table 4.5: Results for the hierarchical attention model trained with 300 dimensional GloVe embeddings and NADAM

	Precision	**Recall**	**F1-score**	**Support**
comp	0.97	0.89	0.93	1955
politics	0.77	0.83	0.80	1050
rec	0.94	0.85	0.89	1590
religion	0.93	0.70	0.80	968
avg / total	0.92	0.83	0.87	5563

For the 20 Newsgroups dataset the best model was found to be the hierarchical attention network trained with NADAM and using the 300 dimensional GloVe embeddings. Its results are shown in table 4.5. This can be compared with the results from (Hingmire et al., 2013) where the best model gets a f1 of 0.93.

4.4 Visualization

The following text is from the BBC News website (News, 2018) and is classified as real by both the hierarchical attention network and the 3HAN.

Headline:
"Brexit: EU leaders warn time is running out for a deal"

Text:
"Time is running out for a Brexit deal, leaders of European countries have warned ahead of more talks in Salzburg.

There is still no agreement on issues including how to avoid new checks on the Northern Irish border.

At the EU summit on Wednesday evening Mrs May stressed her "serious" proposals for future co-operation would ensure a "shared closed relationship".

European Commission chief Jean-Claude Juncker described her 10-minute presentation as "interesting".

"It was polite, it was not aggressive, she was doing her job," he told reporters as he arrived for a second day of talks.

Kuenssberg: The sound of "no, no, no" EU must 'evolve' Irish plans, insists May Sturgeon calls for Brexit to be delayed Brexit: All you need to know Speaking to BBC Radio 4's Today programme, two of the EU leaders said they hoped the UK would hold another referendum on Brexit, in the hope of reversing the 2016 result.

Maltese Prime Minister Joseph Muscat said most of his counterparts would like the "almost impossible" to happen.

Andrej Babis, the Czech Republic's prime minister, added he hoped the British people might change their minds.

But Mrs May said there was no question of the UK seeking to extend its EU membership.

The UK is due to leave the EU on 29 March 2019, and both sides are trying to reach a deal in time, with a crunch summit specially convened in mid-November."

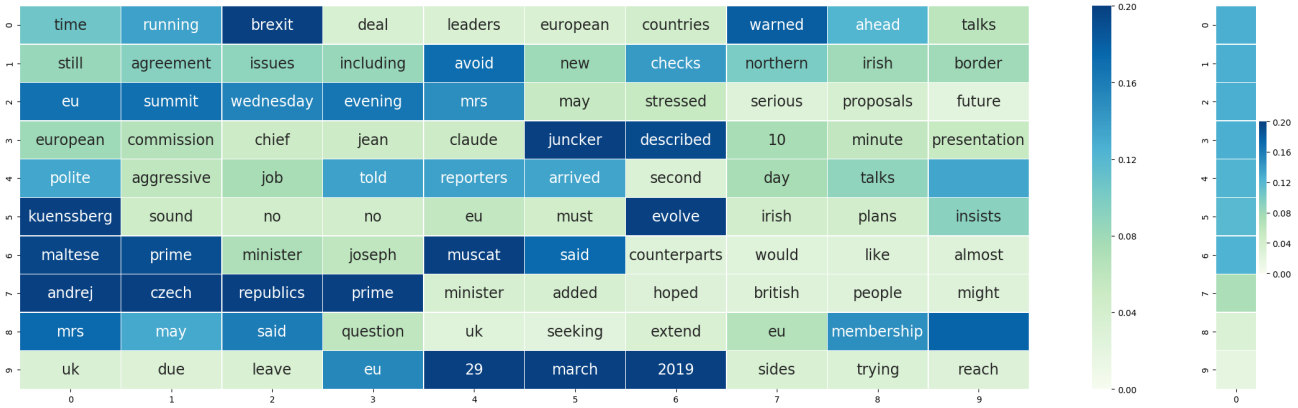

Figure 4.1: The attention weights visualized for the hierarchical attention network. The sentence attention is visualized in the table to the right. Each square representing the sentence to it's left.

Figure 4.2: The attention weights visualized for the 3HAN model. The sentence attention is visualized in the table to the right.

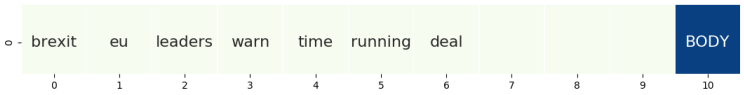

Figure 4.3: The headline attention visualization for the 3HAN model. The scale is the same as for Figure 4.1.

In Figure 4.1 the attention weights of the hierarchical attention network for the text above are visualized. In Figure 4.2 the same is done for the 3HAN model. The headline for the 3HAN model is visualized in Figure 4.3. The headline is not fed in as the first sentence of the hierarchical attention network as to allow for comparison between the weights.

Chapter 5
Discussion

The best performing baseline model was the SVM as can be seen in 4.1. When comparing to other models the hierarchical models perform better than all the baseline models. They also perform better than the basic neural network based models. We can see that the best performing attention GRU is the only non-hierarchical model to outperform the baseline models.

The hierarchical models give a clearly better performance than the baseline and basic models. It is however not a cheap performance increase. The best deep hierarchical models take hours to train, the best deep basic neural network based models take around an hour and the baseline models take five minutes. Depending on the purpose of the model this might be worth if to get the extra performance as it pushes all metrics close to 1.

Interestingly enough the performance of 0.95 of the pre-trained 3HAN with GloVe is not far away from the 0.9677 that (Singhania et al., 2017) reached on their fake news dataset. It is not the same training or test data so they can not be directly compared. However, the model performs well on two totally different datasets which is encouraging.

Another important observation is that the hierarchical attention network outperforms the 3HAN model. An interesting question is if the third layer actually does what it is supposed to do. The idea in the original article (Singhania et al., 2017) was that the third layer would compare the text body to the headline. But I am not sure that is what the model is doing. This will be discussed further in the visualization section.

As the 3HAN model is the less examined I have focused my work on examining and improving it. Mostly this has consisted of trying different embeddings. The only obvious improvement was made by the FastTest embeddings. Interestingly the makeshift solution to having all words embedded with FastText by training the tokenizer on the test data as well did not perform better than just using FastText as embeddings for the 200000 most common words in the training set. The randomly initiated embeddings and the embeddings trained on the training data did not perform well. Not surprisingly as the dataset is quite small and might not be enough to train well performing embeddings.

5.1 Swedish speeches dataset

For the Swedish dataset there are not many comparisons to be made. This is to the best of my knowledge the first time a hierarchical attention model has been applied to Swedish data and the first text classification of the used dataset. We can however see that the FastText embeddings in table 4.3 outperformed the SweVec in table 4.3. Another interesting observation that can be seen from table 4.4 and 4.3, is that the NADAM optimizer outperforms the ADAM even if NADAM only used standard parameters and ADAM was parameter optimized.

5.2 Visualization

There are interesting observations to be made from the visualization examples. Though it is difficult to draw any conclusions from the attention weights of single words we can see that words like brexit and 2019 get a high attention value in both models. Brexit is of course an important word as it is a sensitive political subject. 2019 is harder to analyse. As the dataset is two years old, 2019 would refer to a time three years into the future. This might be an important feature as biased or fake news might be more focused on current affairs.

The absolutely most stand-out fact is that the headline in Figure 4.3 has basically no attention. All the weight has been put on the text body. This is a pattern being repeated in a lot of the test cases. Some have a small attention weight on one word but the body always has most attention. I think this is due to the model not doing what the original authors intended. All the information in the headline is basically lost as the model has learnt to recognise and focus on the body encoding as it contains more information about the article. The idea that the model would be able to compare the headline with the body seems unlikely. An important reason for this is that the model does not know what position is the body encoding but has to learn this. As it does not focus on other words it is more reasonable to assume that it has simply learnt to ignore the headline. This would make it less effective than the hierarchical attention network which is usually fed the headline as a sentence and therefore makes more use of it.

5.3 20 Newsgroups dataset

The best model for the 20 Newsgroups dataset in table 4.5 got a f1 of 0.87. It is not as good as the state of the art performance of (Hingmire et al., 2013) with a f1 of 9.93, but it is comparable to the other models in the paper. From this it is easy to see that the hierarchical attention network has an overall good performance on text classification even if the model probably don't benefit from its structured approach when classifying the posts. It is also interesting to see that the GloVe embeddings outperformed the FastText on this classification task. So what embeddings to chose depend not only on the model and task but also on the data being classified. However NADAM outperforms ADAM on this dataset as well. So for this model it seems to be a good choice.

Chapter 6
Conclusion

6.1 Conclusion

Some important conclusions can be made from the studies of the hierarchical attention network models. The obvious one is that the FastText embeddings performs better than any other tested embeddings on two out of three tasks. Another is that the 3HAN model has performed well on the dataset created and the dataset in its original article. So it seems to be a well working model.

Interestingly enough the 3HAN model does not seem to outperform the hierarchical attention model. In (Singhania et al., 2017) it did outperform the hierarchical attention network but it should be noted that the 3HAN was pre-trained on the headlines which both the original article and this thesis found to improve the model performance. This was not done to the hierarchical attention network in (Singhania et al., 2017). So the comparison made in the original article is not fair.

From the visualization analysis we can see that the model does not compare the headline to the text body. Instead the model has learnt to most of the time ignore the headline and focus on the text. It can however notice an important word in the headline sometimes. On the dataset however, it performs worse than the hierarchical attention network.

Another thing that should be noted is that the NADAM algorithm seem to give better results than the ADAM optimizer even if the NADAM only uses standard parameters and the ADAM is optimized in a grid search.

6.2 Future work

The most obvious opportunity for further examination and something that I would love to see done is having the models tested on a proper real and fake news dataset. A dataset that is public and of high quality with a specific test-set so that the 3HAN and hierarchical

attention network models can be properly compared to other models by other researchers.

Another thing that could be done is comparing the hierarchical attention network and 3HAN on other datasets. Would they perform at an equal level with each other on other datasets as well? And could this mean that the third hierarchical attention level is obsolete?

A thorough analysis of the attention weights could be useful. A study of the weights for more examples might give a more conclusive answer to the question if the headline is usually ignored by the model, or not.

Bibliography

Baziotis, C. (2017). Keras layer that implements an attention mechanism for temporal data. supports masking. follows the work of raffel et al. https://gist.github.com/cbaziotis/6428df359af27d58078ca5ed9792bd6d (accessed May, 2018).

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Cho, K., Merrienboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chopra, S., Jain, S., and Sholar, J. (2017). Towards automatic identification of fake news: Headline-article stance detection with lstm attention models.

Corney, D., Albakour, D., Martinez, M., and Moussa, S. (2016). What do a million news articles look like? In *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016.*, pages 42–47.

Cox, D. (1958). The regression analysis of binary sequences (with discussion). *Journal of the Royal Statistical Society. Series B (Methodological)*, 20:215–242.

Dozat, T. (2015). Incorporating nesterov momentum into.

Fallgren, P., Segerblad, J., and Kuhlmann, M. (2016). Towards a standard dataset of swedish word vectors. *Proceedings of the Sixth Swedish Language Technology Conference*.

Gunther, M. (2015). Xprize's $20m carbon recycling award aims to cut fossil fuel emissions. https://www.theguardian.com/sustainable-business/2015/sep/29/

xprize-carbon-recycling-cleaner-fossil-fuel-coal-gas (accessed May, 2018).

Hingmire, S., Chougule, S., Palshikar, G., and Chakraborti, S. (2013). Document classification by topic labeling.

Humans are Free (2016). Report: Eating raw weed prevents bowel cancer, fibromyalgia and neuro-degenerative diseases. http://humansarefree.com/2016/11/report-eating-raw-weed-prevents-bowel.html (accessed May, 2018).

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Lang, K. (2008). 20 newsgroups. http://qwone.com/~jason/20Newsgroups/ (accessed August, 2018).

Mihailidis, P. and Viotty, S. (2017). Spreadable spectacle in digital culture: Civic expression, fake news, and the role of media literacies in "post-fact" society. *American Behavioral Scientist*, 61(4):441–454.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

News, B. (2018). Brexit: Eu leaders warn time is running out for a deal. https://www.bbc.com/news/uk-politics-45586010 (accessed September, 2018).

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation.

Richardson, L. (2014). Beautiful soup documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (accessed May, 2018).

Risdal, M. (2016). Getting real about fake news | kaggle. https://www.kaggle.com/mrisdal/fake-news (accessed May, 2018).

Ronacher, A., Lord, D., Mönnich, A., and Unterwaditzer, M. (2010). Flask. http://flask.pocoo.org/ (accessed July, 2018).

Sieradski, D. (2017). Bs-detector. https://github.com/bs-detector/bs-detector (accessed August, 2018).

Singhania, S., Fernandez, N., and Rao, S. (2017). 3han: A deep neural network for fake news detection. *Neural Information Processing Lecture Notes in Computer Science*, page 572–581.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

The Swedish Parliament (2010). Anföranden. https://data.riksdagen.se/data/anforanden/ (accessed July, 2018).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Machine learning for distinguishing fake news

In recent years there has been a dramatic increase of fake news in the media. This article takes a machine learning approach, trying to tackle the problem.

A main problem for social media networks trying to deal with fake news is the pure amount of articles being published and shared every day. It is practically impossible for social networks like Facebook or Twitter to manually go through every article and sort out the fake ones. Thankfully recent years advances in artificial intelligence and machine learning makes it possible to automate the process of determining if a news article is trustworthy or not.

The process of automatically classifying a text into two or more classes is called text classification. In this case the classes will be “fake” and “real”. In this study a method based around “hierarchical attention networks” is used. The idea behind a “hierarchical attention network” is that the information in the article is contained not only in the words but in the context. So instead of counting the occurrences of words like simpler methods do, each word is instead put into context of the surrounding words. The same is done for sentences. Each sentence is treated in the context of its surrounding sentences. In this way the network learns what information is important at each level. That means that it can distinguish what words are important for a sentence and what sentences are important for the text.

For example, fake news would usually be about politics and rarely about catastrophes. So the word “earthquake” might be important in determining if the article is fake or not. In the same way a sentence describing the current weather conditions of a city might not be relevant on its own as it could be part of a fake article about how a political candidate uses taxpayer’s money to travel but could just as likely be part of a travel guide.

In addition to applying this method to fake news the study has examined different ways of “training” the model. Training is the process of feeding the model with examples, allowing it to improve how well it performs. The best result that could be reached on the “test set” (a set of data, not part of the training process) was 96% accuracy. A result this good could of course decrease the amount of manual work that would be required for a company to sort out the fake news in its feeds. At the very least it could be used to make a rough filter that could decrease the amount of time required to rout out fake news from our news feed.