Duale Hochschule Baden-Württemberg
Mannheim

Projektarbeit

Neural Classification of Recruitment Advertisements

Course of Study: Business Informatics

Field of Study: Software Engineering

Author:	Fabian Wolf
Matriculation Number:	7345461
Company:	Movilizer GmbH
Department:	Research and Development
Course of Study:	WWI17SEC
Head of Course of Study:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Scientific Supervisor:	M.Sc. Boas Bamberger bamberger@uni-mannheim.de +49 621 181 1562
Business Supervisor:	Oliver Erlenkaemper oliver.erlenkaemper@honeywell.com +49 621 150 207 36
Processing Period:	05 August 2019 – 18 November 2019

Abstract

Title	Neural Classification of Recruitment Advertisements
Author:	Fabian Wolf
Course of Study:	WWI17SEC
Company:	Movilizer GmbH

The career guidance provider Aivy receives recruitment advertisements from several companies, so those of interest for Aivy's customers can be forwarded to them. In order to exclusively forward the recruitment advertisements of interest, they need to be classified first. Manual classification creates expensive and recurrent costs, while the implementation of a neural classification model substituting human labor is a one-time investment. On that account, a neural model holds the potential of significant cost reduction. This paper seeks to propose a neural model for the recruitment advertisement classification task. Therefore, state of the art models for neural text classification are discussed in regard to the requirements of the recruitment advertisement classification task. The state of the art on neural text classification is captured by the literature review conducted by this paper. The requirements are derived from the recruitment advertisement classification task. In general, any state of the art model for text classification satisfies the requirements. Accordingly, no model can be excluded based on the requirements. However, XLNet is the on average best performing model on several text classification leaderboards.[1],[2],[3] Furthermore, XLNet is available under Apache License 2.0. This license allows for commercial use free of charge. For this reason, this paper proposes the pre-trained XLNet fine-tuned on the recruitment advertisement classification task for this task.

[1]https://gluebenchmark.com/leaderboard/
[2]https://nlpprogress.com/english/text_classification.html
[3]https://paperswithcode.com/sota

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
AWS	Amazon Web Service
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MLP	Multi Layer Perzeptron
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network

1 Introduction

1.1 Motivation

Neural text classification is a popular field in Natural Language Processing (NLP) with various real world applications. Within that field a vast amount of classifier models has been introduced.[4] Accordingly, finding and understanding the state of the art models is key in applying them to real world applications. This paper reports the results of a literature review on the state of the art on neural text classification models, finds CNNs, RNNs and Transformers to be the most used models and conveys a deeper, mathematical understanding of these models.

Furthermore, this paper proposes a model for a real world business process. This business process is the task of German recruitment advertisement classification and is described as part of a business scenario by the start-up Aivy.[5] Aivy provides career guidance for their customers. That is why companies send recruitment advertisements to Aivy, so those of interest for Aivy's customers can be forwarded to them. In order to exclusively forward the recruitment advertisements of interest, they need to be classified first. Manual classification creates expensive and recurrent costs, while the implementation of a neural model substituting human labor is a one-time investment.[6] On that account, a neural model holds the potential of significant cost reduction. In the course of the literature review, no paper was found on the classification of German recruitment advertisements. Consequently, this paper proposes a model for a novel classification task.

1.2 Problem Statement

The recruitment advertisement classification task assigns classes to recruitment advertisements. According to Aivy's business scenario, these recruitment advertisements

[4] Zichao Yang et al. 2016; Cho et al. 2014; J. Y. Lee and Dernoncourt 2016; Hochreiter and J. Schmidhuber 1997; Kant et al. 2018; P. Zhou, Shi, et al. 2016; Y. Wang and Tian 2016; Liang, Wu, and C. Huang 2019; Y. Kim 2014; Gao, Ramanathan, and Tourassi 2018; X. Zhang, J. Zhao, and LeCun 2015; Johnson and T. Zhang 2017; Le Hoa, Cerisara, and Alexandre 2017; P. Yang et al. 2018; Rezaeinia, Ghodsi, and Rahmani 2018; W. Zhao et al. 2018; Lai et al. 2015; J. Zheng and L. Zheng 2019; Johnson and T. Zhang 2016; Vaswani et al. 2017; B. Wang 2018; Iyyer et al. 2015.

[5] The original text of the document of Aivy is attached in Section A.1

[6] Infrastructure costs are neglected due to their insignificance compared to the costs of human labor, for example, see `https://aws.amazon.com/de/ec2/pricing/on-demand/`

consist of a job title and a job description. However, according to Aivy's business scenario, annotation and structure of the data cannot be assumed. Furthermore, structured data can be stripped of its annotations by a parser. Thus, a recruitment advertisement is uniformly represented as plain text.

The classes are standardized by the German Federal Employment Agency *Bundesagentur für Arbeit*. A class consists of a four digit decimal code, called *Tätigkeitsschlüssel*, which loosely translates to vocation id. This key is structured as a tree. Meaning, each higher digit aggregates the job categories of the next lower digit into a higher level job class. For example, the vocation id 1219 is the key for manager in the horticulture sector, and the vocation id 121 is the key for the class aggregating jobs of the horticulture sector, like manager in the horticulture sector, tree caregiver, agriculture engineer, gardener, etc..[7]

In order to train the model proposed by this paper, Aivy provides a dataset based on data from the Stepstone Application Programming Interface (API).[8] The dataset is created through Stepstone's search algorithm. For each vocation id the corresponding job title is queried from the Stepstone API. All recruitment advertisements returned by that query are labeled with that job title and its corresponding vocation id. Stepstone's search algorithm probably is not perfect, and human labeling is expensive. That is why the relevance ranking of the search algorithm is returned as a kind of confidence metric as well. The returned data is annotated and structured as XML, an example is shown in Listing A.1. The XML is structured via elements. The important elements are explained in the following.

- $< title >$ element contains the job title
- $< description >$ element contains the job description
- $< standardJobtitle >$ element contains the job title corresponding to the vocation id
- $< vocationId >$ element contains the vocation id
- $< matchRank >$ element contains the relevance ranking

In order to run the model Aivy provides the Amazon Web Service (AWS) infrastructure.

[7]Bundesagentur für Arbeit 2010.

[8]Access to the Stepstone API is part of Stepstone's cooperation partner program, under `https://www.stepstone.de/stellenanzeige-online-aufgeben/`. Accordingly, the Stepstone API is not publicly accessible. Therefore, the URL of the API cannot be provided.

1.3 Scope

Summarizing, the scope of this paper comprises a proposed model for the task of recruitment advertisement classification and the providing of insight on the state of the art on neural text classification models. This paper exclusively focuses on neural models and text classification. Non neural methods are excluded, as neural models are dominating the top of text classification leaderboards.[9,10,11] Other tasks like image classification or language modeling are excluded, since the classification of recruitment advertisements falls into the category of text classification.

This paper focuses on the technical aspects of state of the art models for neural text classification. Therefore, this paper does not discuss infrastructure alternatives, implementation frameworks, surrounding processes and economic efficiency. Surrounding processes refers to all processes around the process of recruitment advertisement classification itself, for example accessing and storing classified recruitment advertisements. In this context, economic efficiency refers to the comparison of the proposed model and alternative classification methods. However, cost advantages of a neural model over another neural model like training time and license costs, are taken into account.

Due to time constraints, experiments to optimize and test the proposed model cannot be run. On that account, the implementation of the proposed model and experiments are excluded from this paper.

1.4 Approach

In order to capture the state of the art on neural text classification models, this paper conducts a literature review. For the proposed model, the state of the art models resulting from the literature review are analyzed in regard to the requirements of the described business process. Especially performance on text categorization tasks and the amount of required training data is taken into account. The requirements are listed in Chapter 4.

[9]`https://gluebenchmark.com/leaderboard/`
[10]`https://nlpprogress.com/english/text_classification.html`
[11]`https://paperswithcode.com/sota`

1.5 Structure

The rest of this paper is structured as follows.

- Chapter 2 defines and illustrates terms required to understand this paper.
- Chapter 3 lists the state of the art on neural text classification models and describes the most used models.
- Chapter 4 lists the requirements of the task described in Section 1.2.
- Chapter 5 discusses the state of the art models in regard to the task described in Section 1.2 and proposes a model for the task.
- Chapter 6 summarizes and evaluates the contributions of this paper. Furthermore, it proposes future work.

2 Fundamentals

Understanding this paper requires knowledge about certain terms. These terms are defined and illustrated in this chapter.

2.1 Text Classification

Text classification is a special topic within the more general field of text mining which tries to extract useful information from unstructured texts.[12] Text classification does this by labeling a text with its corresponding class labels. A text can correspond to only one class (single-label) or more classes (multi-label). Assigning only one label out of several classes is called multi-class classification while assigning several labels is called multi-label classification.

It is common to represent the class distribution of a text with a one-hot array for multi-class classification or a multi-hot array for multi-label classification. Each scalar of the vector represents the probability of a given class. For example, imagine classifying recruitment advertisements for baker, barber and butcher. In this example the class distribution vector has three scalars the first one corresponding to baker, the second one to barber and the third one to butcher. In consequence, a golden vector for a bakers recruitment advertisement looks like (2.1).

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{2.1}$$

For neural multi-class classification this is achieved using the softmax activation function. The softmax function represents a probability distribution over all classes. Therefore, the golden vector has the probability 1 for the recruitment advertisements class and 0 for all other classes. As the predicted vector is most likely not equal to the golden vector, the class with the highest scalar value is taken as the predicted class.[13] For multi-label classification this does not work, as the golden vector can contain several probabilities of 1. Because of this the sigmoid or logistics activation function is used. The sigmoid activation function "squashes" all values into the range $[0; 1]$.[14] Here again, the predicted vector is most likely not equal to the golden vector. That is why a threshold is used to decide whether a classes is taken as predicted or not.

[12] Baharudin et al. 2010.
[13] Krause 2017.
[14] Krause 2017.

All classes with a corresponding scalar value above a certain threshold are taken as predicted.[15]

2.2 Neuron

A neuron is the basic unit of a neural network. A neuron consists of n weighted connections $w_i, i \in [1; n]$, a sum function $\sum$ and an activation function φ,[16] see Figure 2.1. Common activation functions are sigmoid, tanh and Rectified Linear Unit (ReLU). In recent years ReLU is commonly used.[17] Chaining these operations produces the output y of a neuron. Given an input $\vec{x}$ this operator chain results in Equation (2.2).[18]

$$\begin{aligned} y &= \varphi(\textstyle\sum_{i=1}^{n} x_i \cdot w_i) \\ &= \varphi(\vec{w}^{\mathrm{T}} \cdot \vec{x}) | \vec{w}^{\mathrm{T}} = (w_1, w_2 \ldots w_n) \end{aligned} \tag{2.2}$$

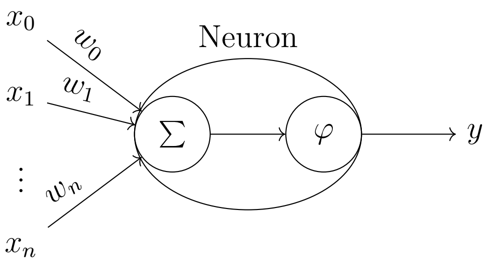


Figure 2.1: Neuron (own figure)

2.3 MLP

A MLP or multilayer feed forward network is a basic neural network structure. A MLP consists of interconnected neurons. The neurons are arranged in m layers $l \in [0, m]$. A layer can contain any number n of neurons. Each neuron of layer l is connected to each neuron in layer $l - 1$, see Figure 2.2.[19] The connections between two layers can be described in a matrix. Given the neurons n_{kl} in row $k, k \in [0, n]$ of layer l, the weight w_{ij} describes the connection between the neuron n_{jl-1} in row j of the previous layer $l - 1$ and the neurons n_{il} in row i of the layer l. In consequence, each

[15] LeCun, Bengio, and G. Hinton 2015.
[16] Jain, Mao, and Mohiuddin 1996.
[17] LeCun, Bengio, and G. Hinton 2015.
[18] Jain, Mao, and Mohiuddin 1996.
[19] Svozil, Kvasnicka, and Pospichal 1997.

layer possesses an own weight matrix W_l. In this matrix the weights describing all incoming connections of a neuron n_{kl} are arranged in row k. This row corresponds to the transposed weight vector of a neuron $\vec{w}^{\mathrm{T}}$, as described in Equation (2.2). Resulting therefrom, applying an element wise activation function to the product of the weight matrix W_l and the input vector $\vec{x_l}$ of a layer results in the output vector of that layer $\vec{y_l}$. This operation chain is described by Equation 2.3. Consequently, this equation can be seen as applying a linear vector space transformation followed by a non linear vector space transformation of the layers input.

$$\vec{y_l} = \varphi(W_l \cdot \vec{x_l}) \tag{2.3}$$

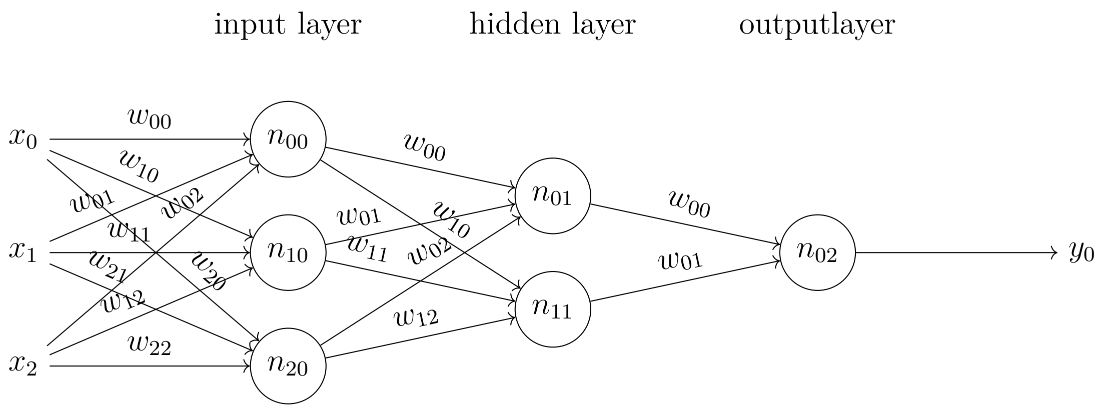
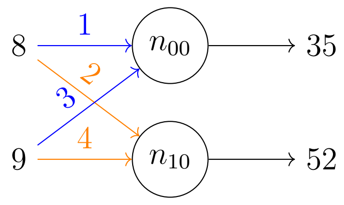


Figure 2.2: MLP Illustration (own figure)

Figure 2.3 illustrate the transfer of the connections between two layers to their matrix representation. In this illustration all incoming connections of a neuron are highlighted in the same color. These connections are described by the transposed weight vector of that neuron. The transposed weight vectors are highlighted in the same color as their corresponding connections. Concatenating these transposed weight vectors results in the weight matrix of that layer. For reasons of simplicity, in this illustration the activation function φ_0 is the identity function.



$$\vec{y_0} = \varphi_0(M_0 \cdot \vec{x_0}) = \varphi_0\left(\begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \cdot \begin{pmatrix} 8 \\ 9 \end{pmatrix}\right) = \varphi_0\left(\begin{pmatrix} 1 \cdot 8 + 3 \cdot 9 \\ 2 \cdot 8 + 4 \cdot 9 \end{pmatrix}\right) = \begin{pmatrix} 35 \\ 52 \end{pmatrix}$$

Figure 2.3: Illustration of the Matrix Representation (own figure)

Additionally to neurons a layer may contain bias units. A bias unit is a neuron without any input and an output of 1. Therefore, a bias just adds its weights to the neurons it is connected to. Consequently the impact of a bias on a layer can be described by a bias weight vector b_l, see Equation (2.4).

$$\vec{y_l} = \varphi(W_l \cdot \vec{x_l} + b_l) \tag{2.4}$$

A MLP consists of several layers. The first layer is called the input layer, the last layer is called the output layer and any other layer is called hidden layer. Each layer uses the output of the previous layer as its own input, except for the input layer, which receives the networks input $\vec{x}$.[20] As a result, the MLP can be described as a chain of layers resulting in Equation (2.5).

$$\vec{y} = \varphi(W_m \cdot \ldots \varphi(W_1 \cdot \varphi(W_0 \cdot \vec{x} + b_0) + b_1) \cdots + b_m) \tag{2.5}$$

2.4 Learning with Neural Networks

Machine learning in general is the art of training an algorithm to generate new information from data. The goal of machine learning is not to model explicitly how to extract this information, but to let the computer itself learn the model.[21]

A neural network is an universal function approximator.[22] Meaning, they can learn any function, for example classifying recruitment advertisements as barber, baker and butcher. How well a neural network approximates a function is measured by an objective function $\mathcal{L}$. The objective function usually represents some kind of error on a given task. Hence, it is the goal to minimize that function. This is done by adjusting

[20]Svozil, Kvasnicka, and Pospichal 1997.
[21]Mohri, Rostamizadeh, and Talwalkar 2012.
[22]Hornik, Stinchcombe, and White 1989.

the neural network's parameters θ. To properly adjust θ, backpropagation[23] is used to calculate a gradient vector $\partial\theta$. This gradient indicates by what amount the error increases or decreases if θ is increased by a tiny amount. The objective function with regard to θ can be seen as high-dimensional hilly landscape, with the negative gradient vector indicating the direction of steepest descent in that landscape. This gradient is used to descend in that landscape to find the minimum. In practice, usually stochastic gradient descent is used to minimize the objective function.[24]

Supervised Learning is learning using an objective function that measures the error (or distance) between the actual output of the neural network and the desired output. Therefore, a large dataset of labeled input data is required. In regard to the given example, this dataset consists of recruitment advertisements labeled by their corresponding class. After training the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine. The generalization ability is the ability to produce sensible outputs on new inputs not seen during training.[25]

Unsupervised Learning is learning without knowing the desired output.[26] In regard to the given example, in the case of unsupervised learning the classes corresponding to the recruitment advertisements are unknown.

2.5 Transfer Learning and Pre-Training

Transfer learning aims to help learning a specific task by learning another task. For example, learning the classification of recruitment advertisements requires some form of learning a language. This applies to the task of language modeling as well. Consequently, training a model for language modeling might help to learn the classification of recruitment advertisements.

Transfer learning is defined as follows: "Given a source domain $\mathcal{D}_\mathcal{S}$ and learning task $\mathcal{T}_\mathcal{S}$, a target domain $\mathcal{D}_\mathcal{T}$ and learning task $\mathcal{T}_\mathcal{T}$, transfer learning aims to help improve the learning of the target predictive function $f_\mathcal{T}(\cdot)$ in $\mathcal{D}_\mathcal{T}$ using the knowledge in $\mathcal{D}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$, or $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$" Pan and Q. Yang 2010. A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is comprised of a feature space $\mathcal{X}$ and a probability distribution $P(X)$, $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$. A feature space is a vector space containing a vector embedding

[23]Rumelhart, G. E. Hinton, and Williams 1986.
[24]LeCun, Bengio, and G. Hinton 2015.
[25]LeCun, Bengio, and G. Hinton 2015.
[26]Ghahramani 2004.

for each input. In regard to the given example, the recruitment advertisement are a corpus of words. These words can be embedded via vectors, which form a sequence of vectors. This sequence of vectors is the input to the classifier. Resulting from that, the vector space in which the input is located is called feature space. A task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ is comprised of a label space $\mathcal{Y}$ and a prediction function $f(\cdot)$. In the given example, recruitment advertisements are labeled by their job titles. These titles are the classes of this classification task. As explained in Section 2.1, for this example, each class distribution can be represented with a one-hot vector. Hence, the vector space containing these one-hot vectors is called label space. For this task, the objective of the prediction function $f(\cdot)$ is to predict the corresponding class $f(x_i) = y_i \in \mathcal{Y}$ from an input $x_i \in \mathcal{X}$. That means to predict the job title corresponding to a recruitment advertisement.[27]

Pre-training and fine-tuning is a form of transfer learning. Pre-training refers to training a model on a source task solely for the purpose of learning the target task faster. For pre-training, the model is trained on a source task with a vast amount of training data available. Usually, an unsupervised task, like a kind of language modeling, is used. Subsequently, the model is fine-tuned. For fine-tuning, the model is initialized with the parameters learned in the pre-training and trained on the target task.[28] This is especially useful for target tasks with few training data.

[27]Pan and Q. Yang 2010.

[28]Joshi et al. 2019; Devlin et al. 2018; Y. Liu et al. 2019; Radford et al. 2018; Zhilin Yang et al. 2019; Howard and Ruder 2018.

3 State of the Art

Over the years the research community has introduced various neural models for text classification.[29] In order to capture the whole variety of neural text classification models leaderboards, journals and academic full text databases have been screened for papers on neural text classification, namely:

- Association for Computational Linguistics [30]
- Institute of Electrical and Electronics Engineers Xplore [31]
- Association for Computing Machinery Digital Library [32]
- arXiv[33]
- Journal of Machine Learning Research[34]
- MIT Press Journals[35]
- Proceedings of Machine Learning Research[36]
- Papers With Code[37]
- GLUE[38]
- NLP-Progress[39]
- Open AI[40]

[29]Zichao Yang et al. 2016; Cho et al. 2014; J. Y. Lee and Dernoncourt 2016; Hochreiter and J. Schmidhuber 1997; Kant et al. 2018; P. Zhou, Shi, et al. 2016; Y. Wang and Tian 2016; Liang, Wu, and C. Huang 2019; Y. Kim 2014; Gao, Ramanathan, and Tourassi 2018; X. Zhang, J. Zhao, and LeCun 2015; Johnson and T. Zhang 2017; Le Hoa, Cerisara, and Alexandre 2017; P. Yang et al. 2018; Rezaeinia, Ghodsi, and Rahmani 2018; W. Zhao et al. 2018; Lai et al. 2015; J. Zheng and L. Zheng 2019; Johnson and T. Zhang 2016; Vaswani et al. 2017; B. Wang 2018; Iyyer et al. 2015.
[30]`https://www.aclweb.org/portal/`
[31]`https://ieeexplore.ieee.org/Xplore/home.jsp`
[32]`https://dl.acm.org/`
[33]`https://arxiv.org/`
[34]`http://www.jmlr.org/`
[35]`http://proceedings.mlr.press/`
[36]`http://proceedings.mlr.press/`
[37]`https://paperswithcode.com/`
[38]`https://gluebenchmark.com/leaderboard/`
[39]`https://nlpprogress.com/`
[40]`https://openai.com/`

- Google Scholar [41]

The papers have been queried from these databases based on search queries created from Term Table A.1 and selected based on the selection criteria listed in Section A.4. Due to time constraints, this paper leaves out some databases, like Open Review.[42] Nevertheless, this paper has analyzed over 100 papers on neural text classification. On that account, it is very likely all frequently used models are captured. The most used models are listed in Concept Matrix A.2 and described in this chapter. The concept matrix maps literature to the models they support. The remaining models are listed in Section 3.4.

3.1 CNN

A highly regarded CNN is the LeNet5 by Yann LeCun. LeNet5 was originally used for image classification. A grayscale image can be represented as matrix with each element corresponding to the grayscale of a pixel. Colored images can be represented in the same way by three matrices one for the red, green and blue scale value of a pixel. These matrices are called channels. A CNN is structured like a MLP, which neurons only connect partially to the input, see Figure 3.1.[43]

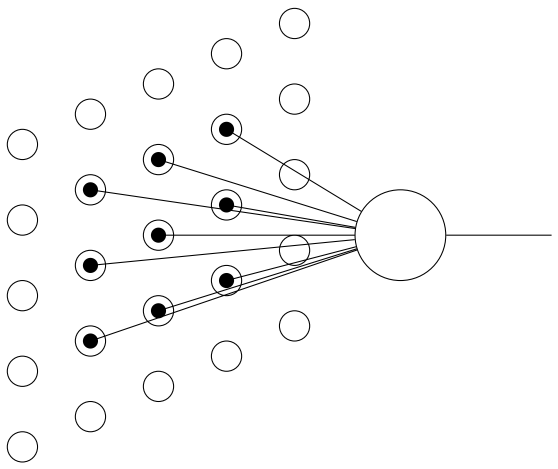


Figure 3.1: Neuron of a CNN (own figure)

In order to understand a CNN three main operations need to be explained:

- Convolution

[41]`https://scholar.google.de/`
[42]`https://openreview.net/`
[43]LeCun, Léon Bottou, et al. 1998.

- Non Linearity
- Sub-Sampling

Convolution in regard to CNNs loosely refers to discrete convolution. Let the input matrix be X. Further, let the weight matrix of the CNN's neuron be w with k rows and l columns called filter or kernel. Further, let the sub matrix resulting from the area the neuron is connected to be $X[i : i + l - 1, j : j + k - 1]$. Then the convolution is defined as (3.1).

$$X * w := w \cdot X[i : i + l - 1, j : j + k - 1] = O \tag{3.1}$$

Meaning that each element of the resulting output matrix O is the matrix product between the sub matrix of the area the neuron is connected to and the weight matrix of the neuron. The factor by which i and j are incremented determines the area the neuron is connected to. This factor is called *stride*. In consequence, an output matrix is produced for each kernel. The resulting output matrices are called activation maps or feature maps.[44]

In order to illustrate this abstract definition, consider the case of a grayscale image represented as a matrix X, as shown in Figure 3.2. Furthermore, consider the kernel w, as shown in Figure 3.2. Then, the convolution of the image and the kernel can be computed by laying the kernel on top of the image. The element of the activation map is computed by multiplying and summing the overlapping elements. To iterate through the elements the kernel is "slid" over each row and each column. Resulting in the output O, as shown in Figure 3.2.

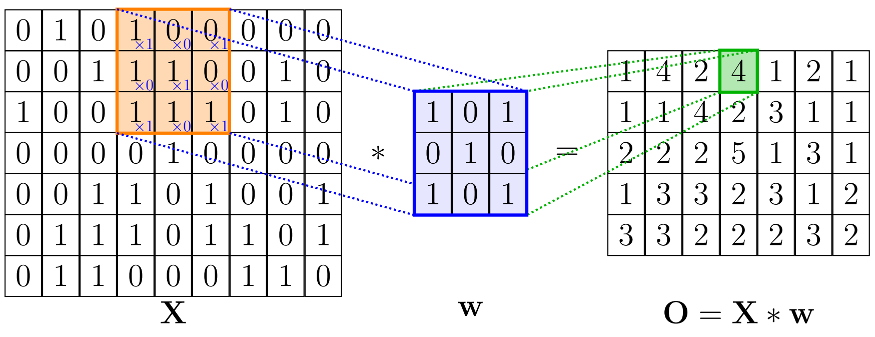


Figure 3.2: Convolution Illustration (own figure)

As shown in Figure 3.2, the resulting output is smaller than the original input. To preserve the original size, *padding* may be used. *Padding* can be described as adding

[44] Ye Zhang and Wallace 2015.

zeros at the edges of the input. Therefore, the output size results in Equation (3.2).[45]

$$outputsize = \frac{inputsize - kernelsize + 2 \cdot padding}{stride} + 1 \tag{3.2}$$

Non linearity needs to be applied to the convolution in order to model non linearly correlated data, because the convolution as defined in (3.1) is completely linear. Applying a non linear activation function φ and a bias b results in Equation (3.3).[46]

$$y = \varphi(X * w + b) = \varphi(w \cdot X[i : i + l - 1, j : j + k - 1] + b) \tag{3.3}$$

Thus, Equation (3.3) is rather similar to Equation (2.2) which is used in Section 2.2 to describe a neuron. The only difference is the shape of the input the neuron is connected to. Deducing from this, neurons of a MLP and a CNN do not differ in their structure.

Sub-sampling is used after convolution and non linearity are applied.[47] Sub-sampling means reducing the size of the feature maps. The computational complexity of a CNN increases over depth in regard to the number and size of the feature maps.[48] Due to that, sub-sampling reduces the computational complexity in multi layer CNNs. Sub-sampling can be achieved by different strategies. However, Ye Zhang and Wallace 2015 found that max pooling outperforms other sub-sampling strategies. Max pooling is defined as (3.4).[49]

$$MaxPooling(X) := max(X[i : i + l - 1, j : j + k - 1]) \tag{3.4}$$

Accordingly, max pooling works similar to convolution. The pooling kernel is “slid” over the input matrix, but instead of convolving the sub matrix and the filter, the maximum of the sub matrix is taken, see Figure 3.3.

[45]Conneau et al. 2016.
[46]LeCun, Léon Bottou, et al. 1998.
[47]LeCun, Léon Bottou, et al. 1998.
[48]Conneau et al. 2016; Johnson and T. Zhang 2017.
[49]Boureau et al. 2010.

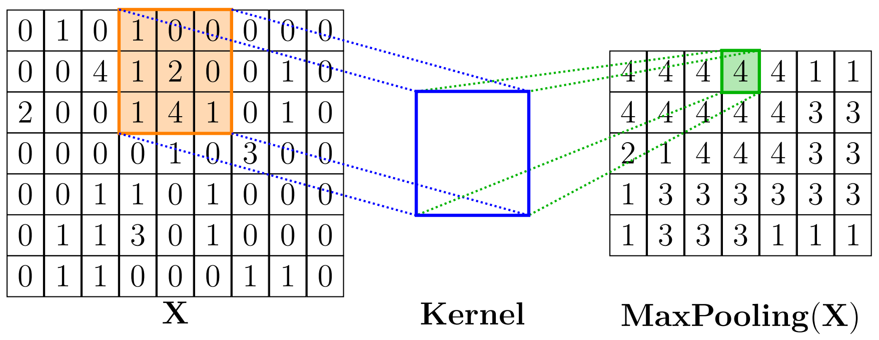


Figure 3.3: Max Pooling Illustration (own figure)

These three operations form the basic building block of a CNN. This building block is comprised of a convolutional layer followed by a non linearity. The pooling layer is applied after the non linearity and is optional. This building block can be repeated any number of times.[50] This is illustrated in Figure 3.4.

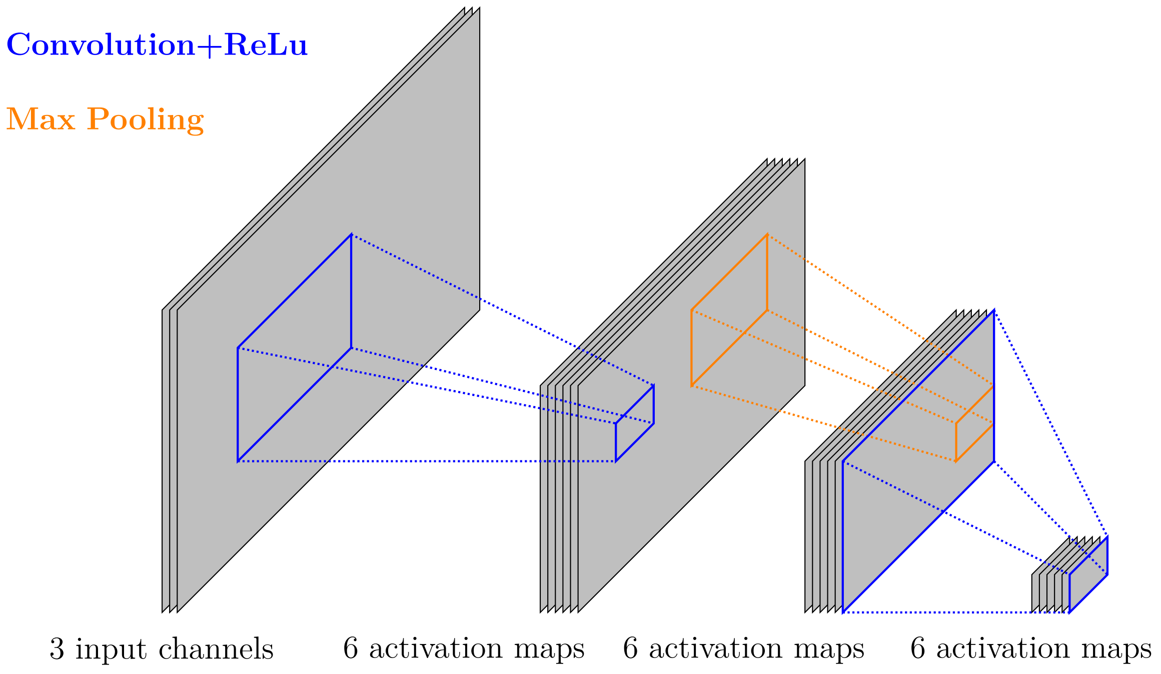


Figure 3.4: CNN (own figure)

For text classification the CNN works in the same way. The input for text classification can be described as a sequence of words. These words can be embedded as

[50]LeCun, Léon Bottou, et al. 1998.

vectors which are concatenated into a sequence matrix. This sequence matrix is the input channel to the CNN.[51]

3.2 RNN

A RNN is structured like a MLP with neurons connect to themselves, see Figure 3.5.[52] Due to that, the output of a RNN, in contrast to a MLP, is dependent on previous computations. On this account, RNNs are able to process sequential information like text. A text can be seen as a sequence of words. These words can be encoded as vectors.[53] This results in a sequence of vectors. Each element of that sequence is processed in the same way. That is why RNNs are called recurrent.

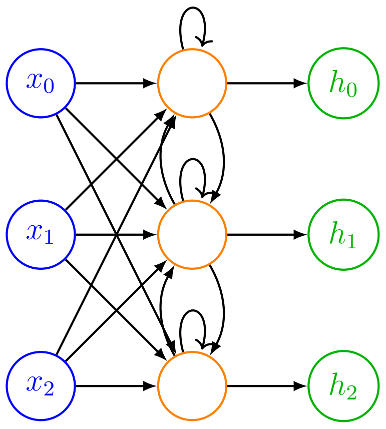


Figure 3.5: Recurrent Neural Network (own figure)

As shown in Section 2.3, connections can be described using a weight matrix. Therefore, let the recurrent connections be described by the weight matrix $W^{(hh)}$ and the connections to the input by the weight matrix $W^{(hx)}$. Further let the input vector x_t be element of the sequence $(x_1, x_2 \dots x_n)$. Then the information, carried between each time step t of the sequence, called hidden state h_t, can be described as (3.5).

$$h_t = \varphi(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \tag{3.5}$$

Consequently, in theory RNNs are able to propagate information through arbitrarily long sequences. However, in practice they are limited to only a few time steps. This

[51]Y. Kim 2014.
[52]Elman 1990.
[53]S. Wang and Manning 2012; Turian, Ratinov, and Bengio 2010; Mikolov, Sutskever, et al. 2013; Mikolov, K. Chen, et al. n.d.; Joulin, Grave, Bojanowski, and Mikolov 2016; Smith 2019; Pennington, Socher, and Manning 2014; McCann et al. 2017.

is because of the vanishing gradient problem.[54] In order to overcome the vanishing gradient problem Long Short-Term Memory (LSTM)[55] and Gated Recurrent Unit (GRU)[56] were designed. Both variations use gated units and a memory vector to directly propagate information between time steps, thus preventing the information from vanishing.[57] According to the results of the papers reviewed for the Concept Matrix A.2, both variations perform much better than vanilla RNNs. However, neither GRUs, nor LSTMs seem to have a significant performance advantage over the other.[58] As more of the reviewed papers used LSTMs, this paper only introduces LSTMs. A LSTM consists of one LSTM cell per time step in a sequence. A LSTM cell at time step t consists of an input gate i_t, an output gate o_t, a forget gate f_t and a cell state c_t, see Figure 3.6. The cell state runs along the entire sequence only influenced by information filtered through the input and forget gate. Therefore, information can be preserved over a long sequence. Gates use the sigmoid function σ and element wise multiplication $\otimes$ to block or convey information. The sigmoid function “squashes” all values in the range of $[0; 1]$, thus creating a vector which scalars are in the range of $[0; 1]$. Any scalar multiplied by 0 is 0. Due to that, any information conveyed by that scalar is blocked. Accordingly, any multiplier above 0 conveys a portion of the information. In consequence, the element wise multiplication multiplies all scalars with a scalar in range of $[0; 1]$. The forget gate applies this operations to the cell state. The scalars of the cell state multiplied by 0 are “forgotten”. The input gate applies a non linearity to the input information, and uses the gating mechanism to decide what part of that information is added to the cell state. The resulting information is called new cell content $\tilde{c_t}$. The output gate applies a non linearity to the cell state, and uses the information conveyed by the input gate for its own gating mechanism to decide what information of the cell state is returned as output. The output is called hidden state h_t.

[54]Hochreiter, Bengio, et al. 2001.
[55]Hochreiter and J. Schmidhuber 1997.
[56]Cho et al. 2014.
[57]Hochreiter, Bengio, et al. 2001.
[58]Greff, R. K. Srivastava, and J. K. Schmidhuber 2015; Shah, Knowles, and Ghahramani 2015.

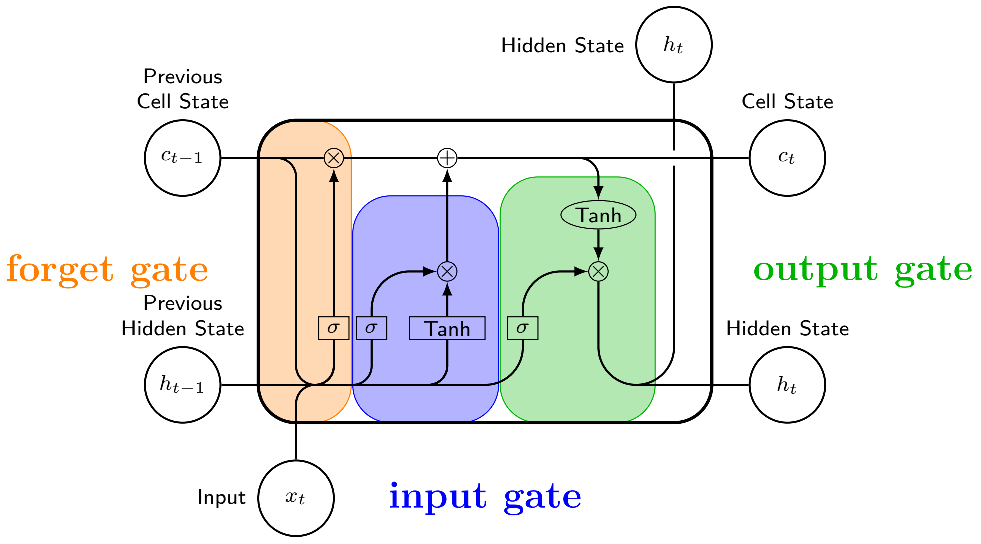


Figure 3.6: LSTM Cell (based on Graves 2013)

As described so far, the gates do not know what information to block or convey. That is why MLPs are used inside the gates, having either $Tanh$ or σ as their activation function. These MLPs learn to operate the gates. They consist of an input layer, an output layer and one hidden layer. As shown in Section 2.3, these layers can be described by matrices, denoted W. In consequence, a LSTM can be described as in Equation (3.6).[59] [60]

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)\\
\tilde{c}_t &= Tanh(W_{xc}x_t + W_{hc}h_{t-1} + W_{cc}c_{t-1} + b_c)\\
c_t &= i_t \otimes \tilde{c}_t + f_t \otimes c_{t-1}\\
h_t &= o_t \otimes Tanh(c_t)
\end{aligned} \tag{3.6}$$

[59] Graves 2013.

[60] As Figure 3.6 shows a concatenation while Equation (3.6) uses a sum operation, it is reasonable to point out that these are equal. $[x_t; h_{t-1}; c_{t-1}][W_x; W_h; W_c] = x_tW_h + h_{t-1}W_h + c_{t-1}tW_c$

3.3 Transformer

Architectures based on the Transformer namely BERT ,[61] ALBERT[62] and XLNet[63] dominate the top of the GLUE leaderboard.[64] [65] The Transformer, originally introduced by Vaswani et al. 2017, mainly utilizes a mechanism called attention. The Transformer is comprised of an encoder and a decoder. However, only the encoder or a modified version of the encoder is used by BERT, ALBERT and XLNet. For this reason, this section focuses on the Transformer encoder.

The encoder is composed of a stack of $N+1$ Blocks. The first block is comprised of an input embedding and a positional encoder. The remaining N blocks are Transformer blocks. The input to the Transformer encoder is a sequence of words of length T. The output of each block is a sequence of hidden states $h_t^{(n)}$ where t is the position in the sequence and n is the block. $n = 0$ denotes the first block. As a result, the input of each block is the sequence of previous hidden states. Each block is composed of two layers. The first is a multi-head self-attention mechanism, and the second is a MLP. A residual connection,[66] followed by layer normalization[67] is employed around each of the two layers. Therefore, the output of each layer can be described as in Equation (3.7).

$$LayerNorm(x + Layer(x)) \tag{3.7}$$

With $Layer(x)$ being the function of the layer itself and $LayerNorm(x)$ being the function described by He et al. 2015, the residual connections bypass the layers and allow the input to flow past them unchanged. On that account, a residual connection can be described as summing the input vectors to the output vectors. Hence, the inputs and outputs of all layers in the model need to be of the same dimensionality. The Transformer encoder is shown in Figure 3.7.

[61]Devlin et al. 2018.
[62]Lan et al. 2019.
[63]Zhilin Yang et al. 2019.
[64]A. Wang et al. 2019.
[65]`https://gluebenchmark.com/leaderboard/`
[66]He et al. 2015.
[67]Ba, Kiros, and G. E. Hinton 2016.

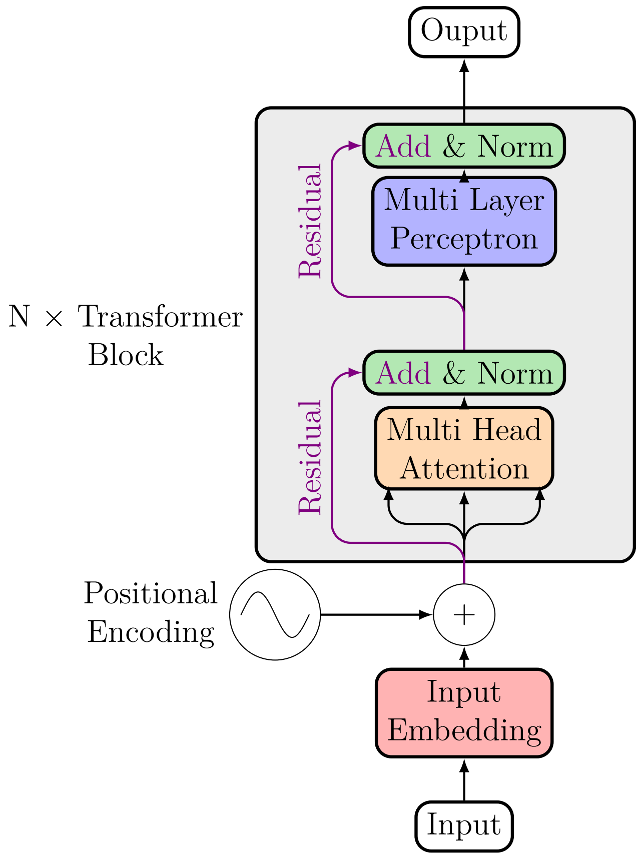


Figure 3.7: Transformer Encoder (based on Vaswani et al. 2017)

Multi-Head Attention is described in the following paragraph.

> An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Vaswani et al. 2017

Vaswani et al. 2017 use dot-product attention[68] scaled by the dimension d_k of the keys. To obtain the scaled dot-product attention the dot-product of the queries with all keys, each divided by $\sqrt{d_k}$ is computed. Subsequently, a softmax function is applied to obtain the weights on the values. At last the values are weighted. The input to a Transformer Block is the sequence of previous hidden states. These hidden states form the queries, the keys and the values. Resulting from this, queries, keys and values are the same, so the attention of a query is to the query itself. For this reason, this kind of attention is called self-attention. In practice, the attention function can be computed

[68]Bahdanau, Cho, and Bengio 2014.

on all queries simultaneously, by concatenating the queries into a matrix Q, the keys into a matrix K and the values into a matrix V, see Equation (3.8).[69]

$$Attention(Q, K, V) = softmax(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}})V \tag{3.8}$$

“Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.” Vaswani et al. 2017 Multi-head attention linearly projects the queries, keys and values h times to d_k dimensions. The projection matrices W_i^Q, W_i^K and $W_i^V, i \in [1; h]$ are individually learned and thus differ from each other. The attention function is applied to each of these projections. The resulting matrices are concatenated and again projected. Resulting in a matrix, being the multi-head attentions output. This is shown in Equation (3.9).[70]

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, head_2, \ldots head_h)W^O \\ \text{where } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3.9}$$

MLPs cannot be applied to matrices. This is a problem, because the multi-head attention, as described in Equation (3.9) returns a matrix. However, this matrix is comprised of attention vectors. On this account one MLP can be applied per attention vector.[71]

Input embeddings are used to convert tokens to vectors. The input embedding is a learned embedding. This is similar to other embedding approaches, like S. Wang and Manning 2012; Turian, Ratinov, and Bengio 2010; Mikolov, Sutskever, et al. 2013; Mikolov, K. Chen, et al. n.d.; Joulin, Grave, Bojanowski, and Mikolov 2016; Smith 2019; Pennington, Socher, and Manning 2014; McCann et al. 2017.[72]

Positional Encoding is used to inject information about a token’s relative or absolute position in the input sequence. This is necessary since the attention mechanism, unlike recurrence or convolution, does not preserve spatial information. In order to be summed, positional encoding and input embedding must have the same dimensionality d. Each embedded token is a vector of size d. Consequently the positional encoding must return a scalar for each dimension $i \in [1, d]$.[73] Vaswani et al. 2017 uses the sine

[69]Vaswani et al. 2017.
[70]Vaswani et al. 2017.
[71]Vaswani et al. 2017.
[72]Vaswani et al. 2017.
[73]Vaswani et al. 2017.

and cosine functions of different frequencies to encode a token's position *pos* in the input sequence , as shown in Equation (3.10).

$$\begin{aligned} PositionEmbedding(pos, 2i) &= sin(\tfrac{pos}{100000^{2/d}}) \\ PositionEmbedding(pos, 2i+1) &= cos(\tfrac{pos}{100000^{2/d}}) \end{aligned} \qquad (3.10)$$

3.4 Miscellaneous

The remaining models found in the course of the literature review are listed below.

- Deep Averaging Network
- Hierarchical Attention Network
- Capsule Network
- Hierarchical Ensemble
- CNN-RNN Ensemble
- RNN-CNN Ensemble

The Deep Averaging Network uses less training time per epoch, but this comes at the cost of weaker performance,[74] the Hierarchical Attention Network is outperformed by a LSTM[75] and the Capsule Network while shown to outperforming simple LSTMs and CNNs,[76] does not surpass some designed LSTMs and CNNs on top of the GLUE,[77] NLP Progress[78] and Papers With Code[79] leaderboards. For these reasons Deep Averaging Networks, Hierarchical Attention Networks and Capsule Networks are not further investigated in this paper.

Ensembles in general are not single neural network architectures but an ensemble comprised of models. The Hierarchical Ensemble uses one model, for example a LSTM, to encode word vectors into sentence vectors and the same or another model to encode the sentence vectors into a document vector. Subsequently a softmax classifier is applied to the document vector. The remaining ensembles use similar strategies. A RNN-CNN Ensemble encodes a sequence of vectors into a matrix using an LSTM, then a CNN is applied to that matrix. A CNN-RNN Ensemble encodes a matrix into a sequence of vectors using a CNN. Subsequently this sequence is used as input to

[74]Iyyer et al. 2015; Cer et al. 2018.
[75]Adhikari et al. 2019.
[76]L. Xiao et al. 2018; J. Kim et al. 2018; S. Srivastava, Khurana, and Tewari 2018; Ren and Lu 2018; W. Zhao et al. 2018.
[77]`https://gluebenchmark.com/leaderboard/`
[78]`https://nlpprogress.com/english/text_classification.html`
[79]`https://paperswithcode.com/task/text-classification`

a LSTM. These models are not explained separately as they simply consist of CNNs and RNNs. However, ensembles are considered for the proposed model.

4 Requirements

As described in Section 1.2, this paper proposes a model for the recruitment advertisement classification task. From this task requirements are derived. These requirements must be met by the proposed model. This chapter provides a tabular overview of these requirements, see Table 4.1. This table divides the requirements in key-value pairs. Thus, requirements can be referenced in the discussion in Section 5.1 using the keys.

Table 4.1: Requirements

Key	Value
Task	Neural text classification
Input	Recruitment advertisement comprised of job title and job description represented as plain text
Output	Vocation id and standardized job title as defined by *Bundesagentur für Arbeit*[80]
Training data	Recruitment advertisements as provided by Aivy, see Section 1.2, comprised of job title, job description, vocation id, standardized job title and match rank
Dataset size	Uncertain
Infrastructure	AWS
Cost	A model with similar performance but cost advantages over another models is preferred

Dataset size, **Infrastructure** and **Cost** are derived from the task but are imprecise requirements. On that account, if possible, this paper specifies these requirements further or explains why the requirements cannot be specified further. Regarding **Dataset size**, the dataset is created through Stepstone's search algorithm, and human labeling is expensive. For this reason, the amount of usable training data is uncertain. Therefore, the amount of required training data should be as small as possible. Regarding **Infrastructure**, Aivy does not give concrete specifications which instance of AWS they use. The smallest G4 instance and simultaneously the cheapest AWS GPU

[80]Bundesagentur für Arbeit 2010.

instance in Frankfurt is the g4dn.xlarge.[81,82,83] The g4dn.xlarge has 16 GB GPU memory.[84] Consequently, 16 GB GPU memory can be assumed as infrastructural constraint. Regarding **Cost**, discussing the economic efficiency of the proposed model in comparison to alternative classification methods exceeds the scope of this paper. However, cost advantages of a neural model over another neural model are taken into account. That is why a model with similar performance but cost advantages over another models is preferred.

[81] `https://aws.amazon.com/ec2/instance-types/?nc1=h_ls`

[82] `https://aws.amazon.com/ec2/instance-types/?nc1=h_ls`

[83] The g4dn.xlarge instance is $0.658 per hour. Since real time classification is not required, the model can be run for a few hours each month to classify gathered recruitment advertisements. These costs are negligible.

[84] `https://aws.amazon.com/ec2/instance-types/?nc1=h_ls`

5 Proposed Model

This chapter discusses which model satisfies the requirements listed in Chapter 4 the most. The requirements are discussed individually. In order to keep it clear which requirement is discussed the requirement is referenced using its key specified in Chapter 4 followed by a short summary of the requirement. Based on this discussion a model for the recruitment advertisement classification task is proposed. Subsequently, the proposed model is described in detail.

5.1 Discussion

Task requires a neural model for text classification. The state of the art on these models is analyzed by the conducted literature review. The literature review finds CNNs, RNNs, Ensembles and Transformers to be considered for the proposed model. All models found are listed in Chapter 3. Among these models, Transformers achieve the highest average performance on the GLUE,[85] NLP Progress[86] and Papers With Code[87] leaderboards. According to these leaderboards, XLNet[88] is the on average best performing Transformer, followed by BERT.[89] The current rank one of the GLUE[90] leaderboard is BERT's successor ALBERT.[91] However, the paper on ALBERT was just released September 2019 and is on conference submission to the ICLR 2020. Hence, as off now there is not much data available on ALBERT's performance for classification tasks. As a result, XLNet seems to be the model of choice if it satisfies the other requirements.

Input requires the model's input to be structured as plain text. The input of XLNet is structured as a sequence of tokens.[92] A plain text is converted to a sequence of tokens by a tokenizer. Consequently, XLNet complies with the restrictions imposed by **Input**. The same applies to BERT and ALBERT. However, XLNet has an edge over BERT and ALBERT. The sequence length of BERT and ALBERT is limited.[93] On that account, recruitment advertisements longer than the specified sequence length

[85]`https://gluebenchmark.com/leaderboard/`
[86]`https://nlpprogress.com/english/text_classification.html`
[87]`https://paperswithcode.com/task/text-classification`
[88]Zhilin Yang et al. 2019.
[89]Devlin et al. 2018.
[90]`https://gluebenchmark.com/leaderboard/`
[91]Lan et al. 2019.
[92]Zhilin Yang et al. 2019.
[93]Devlin et al. 2018; Lan et al. 2019.

are truncated. This is not the case for XLNet.[94] In general, plain text can be converted to comply the input restrictions of any state of the art model.

Output requires the model to return the vocation id and the standardized job title corresponding to the input. Recruitment advertisement classification is a multi-class classification. Accordingly, the output of the model is a vector of dimensionality K equal to the number of classes. Each dimension corresponds to a class.[95] For further explanation, see Section 2.1. Due to that, an index mapping can be used between this vector and arrays containing the vocation id and standardized job title. Hence, any state of the art model can be used to satisfy the **Output** requirement.

Training Data describes the available training data. The training data is not structured as required by **Input**. Because of this, the training data needs to be converted. This is done by a parser. Since the training data is labeled using the Stepstone API's search algorithm, the data is most likely to be noisy. For this reason, the use of a match rank threshold should be examined. The quality of the training data can be further improved by human labeling, but the costs increase with the size of the dataset. However, for a fine-tuning approach a small dataset is sufficient. Derived from this, any state of the art model can be used to satisfy the **Training Data** requirement.

Dataset size leads to the use of pre-training and fine-tuning, as specified in Section 2.5, since the amount of usable training data is uncertain and the creation of high quality training data through human labeling is expensive. A pre-training and fine-tuning method is defined for XLNet, BERT and ALBERT. All three models use some kind of language model pre-training. In addition, BERT and ALBERT use a kind of sentence order objective.[96] Following, all three models can be used to satisfy the **Dataset size** requirement. In general, pre-training and fine-tuning can be used for any state of the art model.

Infrastructure requires the use of AWS. The AWS instance g4dn.xlarge has 16 GB GPU memory.[97] This is sufficient for any state of the art model, even for the lager models XLNet, BERT and ALBERT.[98,99,100]

Costs requires a models with similar performance but cost advantages over another model to be preferred. XLNet and BERT not only outperform other models, they are also available already implemented and pre-trained under Apache License 2.0.[101,102]

[94] Zhilin Yang et al. 2019.
[95] Devlin et al. 2018; Zhilin Yang et al. 2019; Lan et al. 2019.
[96] Devlin et al. 2018; Zhilin Yang et al. 2019; Lan et al. 2019.
[97] `https://aws.amazon.com/ec2/instance-types/?nc1=h_ls`
[98] `https://github.com/zihangdai/xlnet/blob/master/README.md`
[99] `https://github.com/google-research/bert`
[100] Lan et al. 2019.
[101] `https://github.com/zihangdai/xlnet/blob/master/LICENSE`
[102] `https://github.com/google-research/bert/blob/master/LICENSE`

Implementation and pre-training is expensive.[103] Apache License 2.0 allows for commercial use free of charge. This results in performance per cost advantages for XLNet and BERT.

In general, any state of the art model for text classification satisfies the requirements. However, mainly due to performance reasons and performance per cost advantages XLNet seems to be the best choice. Therefore, this paper proposes the pre-trained XLNet fine-tuned on the recruitment advertisement classification task as optimal model for this task.

5.2 XLNet

XLNet is an adapted Transformer model. The adaption is necessary, because of XLNet's novel pre-training objective. On that account, the pre-training objective is explained in order to properly understand XLNet's architecture. According to Zhilin Yang et al. 2019, the most successful pre-training objectives are autoregressive language modeling and autoencoding. Both methods have an advantage complementary to the disadvantage of the other. Autoregressive language modeling predicts the next word in a sequence. Thus, only words at one end of the sequence can be predicted. Accordingly, autoregressive language modeling is only depended on left or right context depending on the direction of prediction. For example, given the sequence "New York is a city", a left to right autoregressive language model could predict "city" from "New York is a", but not "is" from "New York, a city". This method could only predict "is" from "New York". Consequently, the context on the right is lost.[104]

Autoencoding predicts masked words in a sequence. Hence, words anywhere in the sequence can be predicted, but if several words are masked the context of these words is lost. For example, given the sequence "New York is a city" with "is" and "a" masked, both masked words are predicted from "New York, city". Even though due to the sequence of words "a" should be predicted from "New York is, city". As a result, sequence context is lost.[105]

Summarizing, autoregressive language modeling preserves sequence context while losing left or right context, and autoencoding preserves left and right context while losing sequence context. To overcome the disadvantages while preserving the advantages, XLNet's objective is to predict masked words anywhere in the sentence from the whole sequence context including all masked words previous to the predicted word.

[103]Devlin et al. 2018; Zhilin Yang et al. 2019; Lan et al. 2019.
[104]Zhilin Yang et al. 2019.
[105]Zhilin Yang et al. 2019.

For example, given the sequence “New York is a city” with “New” and “York” masked, “New” is predicted from “is a city” and “York” is predicted from “New, is a city”.[106]

XLNet achieves this by providing the complete input sequence. However, predicting a masked word given the whole sequence including the masked word is trivial. For the prediction, the hidden state used to predict the masked word should only be aware of the masked words context and not the masked word itself.[107] For this reason, XLNet uses two hidden states, as described in Equation (5.1).[108]

- The content representation h_{z_t} encoding the context of the input and the input x_{z_t} itself. This representation serves a similar role to the hidden states in a vanilla Transformer.
- The query representation g_{z_t} only encoding the context $x_{z<t}$ and the position z_t, but not the input x_{z_t} itself as discussed above.

$$\begin{aligned} g_{z_t}^{(m)} &\leftarrow \mathit{Attention}(Q = g_{z_t}^{(m-1)}, KV = h_{z<t}^{(m-1)}; \theta) \\ h_{z_t}^{(m)} &\leftarrow \mathit{Attention}(Q = h_{z_t}^{(m-1)}, KV = h_{z\leq t}^{(m-1)}; \theta) \end{aligned} \tag{5.1}$$

The query representation is aware of the complete context, but not the input itself. Due to that, it satisfies the requirements of the pre-training objective. The content representation is calculated the same way as in a vanilla Transformer. Therefore, during fine-tuning, the query stream is dropped and the content stream is used as in a normal Transformer.[109]

In addition to the query stream XLNet has two more adaptions compared to a vanilla Transformer a relative positional encoding scheme and a segment recurrence mechanism. The relative positional encoding scheme as defined by Z. Dai et al. 2019 is used to replace the vanilla Transformer’s position encoding. The segment recurrence mechanism serves as memory to enable an unrestricted sequence input. The input sequence of a the vanilla Transformer is restricted to length T. The segment recurrence mechanism is defined as follows. “Without loss of generality, suppose we have two segments taken from a long sequence s; i.e., $\tilde{x} = s_{1:T}$ and $x = s_{T+1:2T}$. [...] We process the first segment, and then cache the obtained content representations $\tilde{h}^{(m)}$ for each layer m.” Zhilin Yang et al. 2019 Then for the next segment x, the attention update with memory can be written as in Equation (5.2).[110]

$$h_{z_t}^{(m)} \leftarrow \mathit{Attention}(Q = h_{z_t}^{(m-1)}, KV = [\tilde{h}^{(m-1)}, h_{z\leq t}^{(m-1)}]; \theta) \tag{5.2}$$

[106]Zhilin Yang et al. 2019.

[107]Words in the context of XLNet are actually byte pair encoded word pieces with positional information embedded in a vector space.

[108]Zhilin Yang et al. 2019.

[109]Zhilin Yang et al. 2019.

[110]Zhilin Yang et al. 2019.

For fine-tuning, XLNet is first initialized with the pre-trained parameters. These parameters are fine-tuned by using the labeled data of the fine-tuning task. Recalling, the task of the proposed model is to classify recruitment advertisements. XLNet is fine-tuned for a classification task by applying a softmax classifier to the last hidden state in the output sequence of the last Transformer Block. The softmax classifier is a MLP with one hidden layer and as many outputs as the number of classes. Hence, its output is a class distribution vector with each scalar representing the probability of the corresponding class. For further explanation, see Section 2.1. Given the last hidden state $h_T^{(N)}$ and the weight matrix of the softmax classifier $W_{K \times H}$, where N is the number of Transformer Blocks, T is the length of the sequence, K is the number of classes and H is the number of the hidden vector's dimensions, the softmax classifier is described in Equation (5.3).[111]

$$softmax(W_{K \times H} h_T^{(N)}) \tag{5.3}$$

In regard to the recruitment advertisement classification task, the labeled data of the fine-tuning task refers to the tokenized job title followed by the job description. Since the training data as provided by Aivy is structured in XML, it needs to be fitted to the input structure of XLNet. This is done by a parser and a tokenizer. The class distribution vector is derived from the vocation ids, with one dimension corresponding to one vocation id.

After fine-tuning, the parameters of the model are frozen. To be classified, a recruitment advertisement needs to comply to XLNet's input structure. On that account, its job title and job description are stripped of their structure and tokenized. Subsequently, this input is fed into the fine-tuned XLNet. This results in the class distribution vector of that input. Since each dimension of the class distribution vector corresponds to a vocation id, the argmax of the vector can be mapped directly to a vocation id and a standardized job title. These are the model's prediction for that recruitment advertisement.

[111] Zhilin Yang et al. 2019; Devlin et al. 2018.

6 Conclusion

6.1 Summary

This paper provides insight on the state of the art on neural text classification models and proposes a model for the task of recruitment advertisement classification. The state of the art on neural text classification models is captured by the literature review conducted by this paper, requirements are derived from the task of recruitment advertisement classification, and the state of the art models are discussed in regard to the derived requirements. Based on that discussion, this paper proposes the pre-trained XLNet fine-tuned on the recruitment advertisement classification task for this task.

In this paragraph the deficiencies of this paper are reflected. Due to time constraints, not all papers on neural text classification have been analyzed. Nevertheless, this paper has analyzed over 100 papers on text classification. On that account, it is very likely all frequently used models are captured. A huge amount of models results from the literature review. Therefore, in order to keep the overview, this paper focuses on the most used and best performing models. As discussed in Chapter 4, some of the derived requirements are imprecise. On that account, if possible, this paper specifies these requirements further or explains why the requirements cannot be specified further. In general, any state of the art model for text classification satisfies the requirements. That is why no model can be excluded based on the requirements. However, mainly due to performance reasons and performance per cost advantages XLNet seems to be the best choice.

6.2 Future Work

Future work can be conducted in regard to the model itself and in regard to the whole business scenario described by Aivy.

Regarding the model, due to time constraints, this paper does not implement and test the proposed model. This leads to future work and experiments aiming at testing and improving model performance. These can be grouped into two categories hyperparameter optimization and preprocessing.

This paper proposes future work and experiments on the following hyperparameters.

- As discussed in Chapter 5, pre-training is too expensive. For this reason, the hyperparameters for pre-training are determined by the already pre-trained models.[112] Zhilin Yang et al. 2019 published two pre-trained models with different hyperparameter configurations XLNet Base and XLNet Large. XLNet Large performs better on the leaderboards, but is computationally more expensive. Hence, this paper proposes to investigate whether the increase in performance justifies the increase in computational costs.
- For fine-tuning Zhilin Yang et al. 2019 conducted several hyperparameter space searches. The resulting hyperparameters are listed in Table A.3. Because of this, this paper proposes to test the hyperparameters proposed by Zhilin Yang et al. 2019 and only conduct further hyperparameter optimization if the results suggest so. For example, to test an increased learning rate decay or a decreased initial learning rate in the case of catastrophic forgetting.[113]
- Sun et al. 2019 shows that further in-task or in-domain pre-training can increase performance. Therefore, this paper proposes to investigate the application of in-task and in-domain pre-training.

This paper proposes future work and experiments for the following preprocessing steps.

- As discussed in Section 1.2, the provided dataset is not perfectly labeled. That is why this paper proposes to investigate the use of a match rank threshold. Furthermore, if the dataset remains too noisy even after applying a match rank threshold, this paper proposes to consider the use of human labeled data.
- Recruitment advertisements are comprised of a job title and a job description. This forms the input of the classification task. The output of the classification task is a class which corresponds to a standardized job title. On that account, it seems reasonable that the job title is sufficient for the classification. This reduces the required sequence length significantly and might reduce the cost of human labeled data, since labeling a title seems much less work than labeling title and description. In consequence, this paper proposes to investigate the use of the job title without the job description as input.
- In the course of the literature review some papers were found that show adversarial training and feature engineering to improve performance.[114] In consequence, this paper proposes to investigate these techniques for recruitment advertisement classification.

[112]For further details see Zhilin Yang et al. 2019.

[113]Sun et al. 2019.

[114]Camacho-Collados and Pilehvar 2017; Lenc and Král 2017; Amin and Nadeem 2018; Asim et al. 2019; Lyubinets, Boiko, and Nicholas 2018; X. Zhang, J. Zhao, and LeCun 2015; Miyato, A. M. Dai, and Goodfellow 2017; Goodfellow, Shlens Jonathan, and Szegedy Christian 2015.

Regarding the business scenario, this paper focuses on the process of recruitment advertisement classification. Thus, this paper proposes future work on the surrounding processes, the integration of the whole business scenario and the assessment of its economic efficiency. Regarding the economic efficiency the use of alternative infrastructure can be considered as well.

References

Adhikari, Ashutosh et al. (2019). “Rethinking Complex Neural Network Architectures for Document Classification”. In: pp. 4046–4051. DOI: 10.18653/v1/N19-1408. URL: https://www.aclweb.org/anthology/N19-1408.

Amin, Muhammad Zain and Noman Nadeem (2018). “Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System”. In: *CoRR* abs/1809.02479. URL: http://arxiv.org/abs/1809.02479.

Arnold, Sebastian et al. (2019). “SECTOR: A Neural Model for Coherent Topic Segmentation and Classification”. In: *CoRR* abs/1902.04793. URL: http://arxiv.org/abs/1902.04793.

Asim, Muhammad Nabeel et al. (2019). “A Robust Hybrid Approach for Textual Document Classification”. In: URL: https://arxiv.org/pdf/1909.05478.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). “Layer Normalization”. In: URL: https://arxiv.org/abs/1607.06450.

Baharudin, Baharum et al. (2010). “A Review of Machine Learning Algorithms for Text-Documents Classification”. In: *Journal of Advances in Information Technology* 1. DOI: 10.4304/jait.1.1.4-20. URL: http://www.jait.us/index.php?m=content&c=index&a=show&catid=160&id=859.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473. URL: https://arxiv.org/abs/1409.0473.

Baker, Simon, Anna Korhonen, and Sampo Pyysalo (2016). “Cancer Hallmark Text Classification Using Convolutional Neural Networks”. In: *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining*, pp. 1–9. URL: https://www.aclweb.org/anthology/W16-5101.

Baumel, Tal et al. (2017). “Multi-Label Classification of Patient Notes a Case Study on ICD Code Assignment”. In: *CoRR* abs/1709.09587. URL: http://arxiv.org/abs/1709.09587.

Baziotis, Christos, Nikos Pelekis, and Christos Doulkeridis (n.d.). “DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis”. In: (), pp. 747–754. DOI: 10.18653/v1/S17-2126. URL: https://www.aclweb.org/anthology/S17-2126.

Boureau, Y-Lan et al. (2010). “Learning Mid-Level Features For Recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference*, pp. 2559–2566. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.205.8511&rep=rep1&type=pdf.

Bundesagentur für Arbeit, ed. (2010). *Schlüsselverzeichnis für die Angaben zur Tätigkeit.* Regensburger StraSSe 10490478 Nürnberg. URL: https://www.arbeitsagentur.de/datei/dok_ba015567.pdf.

Camacho-Collados, José and Mohammad Taher Pilehvar (2017). “On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis”. In: *CoRR* abs/1707.01780. URL: http://arxiv.org/abs/1707.01780.

Cer, Daniel et al. (2018). “Universal Sentence Encoder”. In: *CoRR* abs/1803.11175. URL: http://arxiv.org/abs/1803.11175.

Chalkidis, Ilias, Ion Androutsopoulos, and Nikolaos Aletras (2019). “Neural Legal Judgment Prediction in English”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4317–4323. DOI: 10.18653/v1/P19-1424. URL: https://www.aclweb.org/anthology/P19-1424.

Chalkidis, Ilias, Emmanouil Fergadiotis, et al. (2019). “Extreme Multi-Label Legal Text Classification: A Case Study in EU Legislation”. In: *Proceedings of the Natural Legal Language Processing Workshop 2019*, pp. 78–87. DOI: 10.18653/v1/W19-2209. URL: https://www.aclweb.org/anthology/W19-2209.

Chalkidis, Ilias, Manos Fergadiotis, et al. (2019). “Large-Scale Multi-Label Text Classification on EU Legislation”. In: *CoRR* abs/1906.02192. URL: http://arxiv.org/abs/1906.02192.

Chang, Wei-Cheng et al. (2019). “X-BERT: eXtreme Multi-label Text Classification with BERT”. In: *CoRR* abs/1905.02331. URL: http://arxiv.org/abs/1905.02331.

Chen, Guibin et al. (2017). “Ensemble application of convolutional and recurrent neural networks for multi-label text categorization”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2377–2383. DOI: 10.1109/IJCNN.2017.7966144. URL: http://ieeexplore.ieee.org/document/7966144/.

Cho, Kyunghyun et al. (2014). “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches”. In: *CoRR* abs/1409.1259.

Clark, Kevin et al. (2019). “BAM! Born-Again Multi-Task Networks for Natural Language Understanding”. In: *CoRR* abs/1907.04829. URL: http://arxiv.org/abs/1907.04829.

Cliche, Mathieu (2017). "BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs". In: *CoRR* abs/1704.06125. URL: `http://arxiv.org/abs/1704.06125`.

Conneau, Alexis et al. (2016). "Very Deep Convolutional Networks for Text Classification". In: *CoRR* abs/1606.01781. URL: `http://arxiv.org/abs/1606.01781`.

Dai, Andrew M. and Quoc Le V (2015). "Semi-supervised Sequence Learning". In: *CoRR* abs/1511.01432. URL: `http://arxiv.org/abs/1511.01432`.

Dai, Zihang et al. (2019). "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". In: *CoRR* abs/1901.02860. URL: `http://arxiv.org/abs/1901.02860`.

Dernoncourt, Franck, Ji Young Lee, and Peter Szolovits (2017). "Neural Networks for Joint Sentence Classification in Medical Paper Abstracts". In: *Proceedings of the 15th Conference of the European Chapter of the*, pp. 694–700. DOI: `10.18653/v1/E17-2110`. URL: `http://aclweb.org/anthology/E17-2`.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. URL: `http://arxiv.org/abs/1810.04805`.

Elman, Jefrey L. (1990). "Finding Structure in Time". In: *Cognitive Science 14*, pp. 179–211. URL: `https://crl.ucsd.edu/~elman/Papers/fsit.pdf`.

Gao, Shang, Arvind Ramanathan, and Georgia Tourassi (2018). "Hierarchical Convolutional Attention Networks for Text Classification". In: *Proceedings of The Third Workshop on Representation Learning for NLP*, pp. 11–23. DOI: `10.18653/v1/W18-3002`. URL: `https://aclweb.org/anthology/W18-3002/`.

Ghahramani, Zoubin (2004). "Unsupervised Learning". In: *Advanced Lectures on Machine Learning*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Vol. 3176. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 72–112. ISBN: 978-3-540-23122-6. DOI: `10.1007/978-3-540-28650-9{\textunderscore}5`. URL: `https://link.springer.com/chapter/10.1007%2F978-3-540-28650-9_5`.

Giannakopoulos, Athanasios et al. (2019). "Resilient Combination of Complementary CNN and RNN Features for Text Classification through Attention and Ensembling". In: *CoRR* abs/1903.12157. URL: `http://arxiv.org/abs/1903.12157`.

Goodfellow, Ian, Shlens Jonathan, and Szegedy Christian (2015). "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations 2015.*

Graves, Alex (2013). “Generating Sequences With Recurrent Neural Networks”. In: *CoRR* abs/1308.0850. URL: `http://arxiv.org/abs/1308.0850`.

Greff, Klaus, Rupesh Kumar Srivastava, and Jan Koutnürgen Schmidhuber (2015). “LSTM: A Search Space Odyssey”. In: *CoRR* abs/1503.04069. URL: `http://arxiv.org/abs/1503.04069`.

Hassan, Abdalraouf and Ausif Mahmood (207). “Deep learning for sentence classification”. In: *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–5. DOI: `10.1109/LISAT.2017.8001979`. URL: `http://ieeexplore.ieee.org/document/8001979/`.

Hassan, Abdalraouf and Ausif Mahmood (2018). “Convolutional Recurrent Deep Learning Model for Sentence Classification”. In: *IEEE Access* 6, pp. 13949–13957. DOI: `10.1109/ACCESS.2018.2814818`. URL: `https://ieeexplore.ieee.org/document/8314136`.

Hassan, Abdalraouf and Ausif Mahmood (n.d.). “Efficient Deep Learning Model for Text Classification Based on Recurrent and Convolutional Layers”. In: *16th IEEE International Conference on Machine Learning and Applications* 2017 (), pp. 1108–1113. DOI: `10.1109/ICMLA.2017.00009`. URL: `http://ieeexplore.ieee.org/document/8260793/`.

He, Kaiming et al. (2015). “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385. URL: `http://arxiv.org/abs/1512.03385`.

Hochreiter, Sepp, Yoshua Bengio, et al. (2001). “Gradient Flow in Recurrent Nets: theDifficulty of Learning Long-TermDependencies”. In: *S. C. Kremer and J. F. Kolen, A Field Guideto Dynamical Recurrent Neural Networks.*

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5, pp. 359–366. ISSN: 08936080. DOI: `10.1016/0893-6080(89)90020-8`. URL: `https://www.sciencedirect.com/science/article/abs/pii/0893608089900208?via%3Dihub`.

Howard, Jeremy and Sebastian Ruder (2018). “Universal Language Model Fine-tuning for Text Classification”. In: *CoRR* abs/1801.06146. URL: `http://arxiv.org/abs/1801.06146`.

Howe, Jerrold Soh Tsin, Lim How Khang, and Ian Ernst Chai (2019). “Legal Area Classification: A Comparative Study of Text Classifiers on Singapore Supreme Court Judgments”. In: *CoRR* abs/1904.06470. URL: `http://arxiv.org/abs/1904.06470`.

Hsu, Tzuhan and Yaoqin Zhang (2018). “Petroleum Engineering Data Text Classification Using Convolutional Neural Network Based Classifier”. In: *2018 International Conference on Machine Learning Technologies*, pp. 63–68. DOI: 10.1145/3231884.3231898. URL: http://dl.acm.org/citation.cfm?doid=3231884.

Hu, Yibo et al. (2018). “Short Text Classification with A Convolutional Neural Networks Based Method”. In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1432–1435. DOI: 10.1109/ICARCV.2018.8581332. URL: https://ieeexplore.ieee.org/document/8581332/.

Huang, Ting, Gehui Shen, and Zhi-Hong Deng (2019). “Leap-LSTM: Enhancing Long Short-Term Memory for Text Categorization”. In: *CoRR* abs/1905.11558. URL: http://arxiv.org/abs/1905.11558.

Huang, Zhengjie et al. (2017). “Length Adaptive Recurrent Model for Text Classification”. In: *Proceedings of the 2017 ACM Conference on Information and Knowledge Management : November 6-10, 2017, Singapore*, pp. 1019–1027. DOI: 10.1145/3132847.3132947. URL: http://dl.acm.org/citation.cfm?doid=3132847.

Iyyer, Mohit et al. (2015). “Deep Unordered Composition Rivals Syntactic Methods for Text Classification”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1681–1691. DOI: 10.3115/v1/P15-1162. URL: https://aclweb.org/anthology/P15-1162/.

Jain, A. K., Jianchang Mao, and K. M. Mohiuddin (1996). “Artificial neural networks: a tutorial”. In: *Computer* 29.3, pp. 31–44. ISSN: 00189162. DOI: 10.1109/2.485891.

Jiang, Jun et al. (2019). “Cross-lingual Data Transformation and Combination for Text Classification”. In: *CoRR* abs/1906.09543. URL: http://arxiv.org/abs/1906.09543.

Johnson, Rie and Tong Zhang (2014). “Effective Use of Word Order for Text Categorization with Convolutional Neural Networks”. In: *CoRR* abs/1412.1058. URL: http://arxiv.org/abs/1412.1058.

Johnson, Rie and Tong Zhang (2015). “Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding”. In: *Advances in Neural Information Processing Systems 28*, pp. 919–927. URL: http://papers.nips.cc/paper/5849-semi-supervised-convolutional-neural-networks-for-text-categorization-via-region-embedding.pdf.

Johnson, Rie and Tong Zhang (2016). “Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings”. In: URL: http://arxiv.org/pdf/1602.02373v2.

Johnson, Rie and Tong Zhang (2017). “Deep Pyramid Convolutional Neural Networks for Text Categorization”. In: pp. 562–570. DOI: 10.18653/v1/P17-1052. URL: https://www.aclweb.org/anthology/P17-1052.

Joshi, Mandar et al. (2019). “SpanBERT: Improving Pre-training by Representing and Predicting Spans”. In: *CoRR* abs/1907.10529. URL: http://arxiv.org/abs/1907.10529.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, Matthijs Douze, et al. (2016). “FastText.zip: Compressing text classification models”. In: *CoRR* abs/1612.03651. URL: http://arxiv.org/abs/1612.03651.

Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov (2016). “Bag of Tricks for Efficient Text Classification”. In: *CoRR* abs/1607.01759. URL: http://arxiv.org/abs/1607.01759.

Kabir, Md Yasin and Sanjay Madria (2019). “A Deep Learning Approach for Tweet Classification and Rescue Scheduling for Effective Disaster Management”. In: *ArXiv* abs/1908.01456. URL: https://arxiv.org/abs/1908.01456.

Kamath, Cannannore Nidhi, Syed Saqib Bukhari, and Andreas Dengel (2018). “Comparative Study between Traditional Machine Learning and Deep Learning Approaches for Text Classification”. In: *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1–11. DOI: 10.1145/3209280.3209526. URL: https://dl.acm.org/citation.cfm?id=3209526.

Kant, Neel et al. (2018). “Practical Text Classification With Large Pre-Trained Language Models”. In: *ArXiv* abs/1812.01207. URL: https://arxiv.org/abs/1812.01207.

Kim, Jaeyoung et al. (2018). “Text Classification using Capsules”. In: *CoRR* abs/1808.03976. URL: http://arxiv.org/abs/1808.03976.

Kim, Kang-Min et al. (2019). “From Small-scale to Large-scale Text Classification”. In: *The World Wide Web Conference on - WWW ’19*, pp. 853–862. DOI: 10.1145/3308558.3313563. URL: http://dl.acm.org/citation.cfm?doid=3308558.

Kim, Yoon (2014). “Convolutional Neural Networks for Sentence Classification”. In: *CoRR* abs/1408.5882. URL: http://arxiv.org/abs/1408.5882.

Kowsari, Kamran et al. (2018). “RMDL: Random Multimodel Deep Learning for Classification”. In: *CoRR* abs/1805.01890. URL: http://arxiv.org/abs/1805.01890.

Krause, Thomas (2017). “Towards the Improvement of Automated Scientific Document Categorization by Deep Learning”. In: *CoRR* abs/1706.05719. URL: http://arxiv.org/abs/1706.05719.

Lai, Siwei et al. (2015). "Recurrent Convolutional Neural Networks for Text Classification". In: URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745.

Lan, Zhenzhong et al. (2019). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: URL: https://arxiv.org/abs/1909.11942.

Le Hoa, T., Christophe Cerisara, and Denis Alexandre (2017). "Do Convolutional Networks need to be Deep for Text Classification ?" In: *AAAI Workshops*. URL: https://www.aaai.org/ocs/index.php/WS/AAAIW18/paper/viewPaper/16578.

LeClair, Alexander, Zachary Eberhart, and Collin McMillan (2018). "Adapting Neural Text Classification for Improved Software Categorization". In: *CoRR* abs/1806.01742. URL: http://arxiv.org/abs/1806.01742.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://www.nature.com/articles/nature14539.

LeCun, Yann, Léon Bottou, et al. (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE, November 1998*. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf.

Lee, Ji Young and Franck Dernoncourt (2016). "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 515–520. DOI: 10.18653/v1/N16-1062. URL: https://www.aclweb.org/anthology/N16-1062.

Lee, Younghun, Seunghyun Yoon, and Kyomin Jung (2018). "Comparative Studies of Detecting Abusive Language on Twitter". In: *CoRR* abs/1808.10245. URL: http://arxiv.org/abs/1808.10245.

Lenc, Ladislav and Pavel Král (2017). "Deep Neural Networks for Czech Multi-label Document Classification". In: *CoRR* abs/1701.03849. URL: http://arxiv.org/abs/1701.03849.

Li, Christy Y. et al. (2017). "Convolutional Neural Networks for Medical Diagnosis from Admission Notes". In: *CoRR* abs/1712.02768. URL: http://arxiv.org/abs/1712.02768.

Liang, Qiancheng, Ping Wu, and Chaoyi Huang (2019). "An Efficient Method for Text Classification Task". In: *Proceedings of the 2019 International Conference on Big Data Engineering (BDE 2019) - BDE 2019*, pp. 92–97. DOI: 10.1145/3341620.3341631. URL: http://dl.acm.org/citation.cfm?doid=3341620.

Lin, Junyang et al. (2018). “Semantic-Unit-Based Dilated Convolution for Multi-Label Text Classification”. In: *CoRR* abs/1808.08561. URL: `http://arxiv.org/abs/1808.08561`.

Lin, Zhouhan et al. (2017). “A Structured Self-attentive Sentence Embedding”. In: *CoRR* abs/1703.03130. URL: `http://arxiv.org/abs/1703.03130`.

Liu, Jingshu, Zachariah Zhang, and Narges Razavian (2018). “Deep EHR: Chronic Disease Prediction Using Medical Notes”. In: *Proceedings of the 3rd Machine Learning for Healthcare Conference* 85, pp. 440–464. URL: `http://proceedings.mlr.press/v85/liu18b.html`.

Liu, Jingzhou et al. (2017). “Deep Learning for Extreme Multi-label Text Classification”. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval : August 7-11, 2017, Shinjuku, Tokyo, Japan*, pp. 115–124. DOI: `10.1145/3077136.3080834`. URL: `http://dl.acm.org/citation.cfm?doid=3077136`.

Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang (2016). “Recurrent Neural Network for Text Classification with Multi-Task Learning”. In: *CoRR* abs/1605.05101. URL: `http://arxiv.org/abs/1605.05101`.

Liu, Yinhan et al. (2019). “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692. URL: `http://arxiv.org/abs/1907.11692`.

Lyubinets, Volodymyr, Taras Boiko, and Deon Nicholas (2018). “Automated labeling of bugs and tickets using attention-based mechanisms in recurrent neural networks”. In: *CoRR* abs/1807.02892. URL: `http://arxiv.org/abs/1807.02892`.

Madasu, Avinash and Vijjini Anvesh Rao (2019). “Sequential Learning of Convolutional Features for Effective Text Classification”. In: *ArXiv* abs/1909.00080. URL: `https://arxiv.org/abs/1909.00080`.

Mani, Senthil, Anush Sankaran, and Rahul Aralikatte (2018). “DeepTriage: Exploring the Effectiveness of Deep Learning for Bug Triaging”. In: *CoRR* abs/1801.01275. URL: `http://arxiv.org/abs/1801.01275`.

McCann, Bryan et al. (2017). “Learned in Translation: Contextualized Word Vectors”. In: *CoRR* abs/1708.00107. URL: `http://arxiv.org/abs/1708.00107`.

Meng, Yu et al. (2018). “Weakly-Supervised Neural Text Classification”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management : October 22-26, 2018, Torino, Italy*, pp. 983–992. DOI: `10.1145/3269206.3271737`. URL: `http://dl.acm.org/citation.cfm?doid=3269206`.

Mikolov, Tomas, Kai Chen, et al. (n.d.). “Efficient Estimation of Word Representations in Vector Space”. In: 2013 (). URL: `https://arxiv.org/abs/1301.3781`.

Mikolov, Tomas, Ilya Sutskever, et al. (2013). “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. URL: `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

Miyato, Takeru, Andrew M. Dai, and Ian Goodfellow (2017). “Adversarial Training Methods for Semi-Supervised Text Classification”. In: *International Conference on Learning Representations 2017*.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2012). *Foundations of machine learning*. Adaptive computation and machine learning. Cambridge, Mass. and London: The MIT Press. ISBN: 9780262018258.

Mullenbach, James et al. (2018). “Explainable Prediction of Medical Codes from Clinical Text the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)”. In: *Proceedings of the 2018 Conference of the North American Chapter of*, pp. 1101–1111. URL: `https://www.aclweb.org/anthology/N18-1100/`.

Paisios, Andreas et al. (2019). “A Deep Neural Network Conformal Predictor for Multi-label Text Classification”. In: *Proceedings of the Eighth Symposium on Conformal and Probabilistic Prediction and Applications*, pp. 228–245. URL: `http://proceedings.mlr.press/v105/paisios19a.html`.

Pan, Sinno Jialin and Qiang Yang (2010). “A Survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. ISSN: 1041-4347. DOI: `10.1109/TKDE.2009.191`. URL: `https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf`.

Park, Ji Ho and Pascale Fung (2017). “One-step and Two-step Classification for Abusive Language Detection on Twitter”. In: *CoRR* abs/1706.01206. URL: `http://arxiv.org/abs/1706.01206`.

Parwez, Md. Aslam, Muhammad Abulaish, and Jahiruddin (2019). “Multi-Label Classification of Microblogging Texts Using Convolution Neural Network”. In: *IEEE Access* 7, pp. 68678–68691. DOI: `10.1109/ACCESS.2019.2919494`. URL: `https://ieeexplore.ieee.org/document/8723320`.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. DOI: `10.3115/v1/D14-1162`. URL: `https://www.aclweb.org/anthology/D14-1162`.

Peters, Matthew E. et al. (2018). “Deep contextualized word representations”. In: *CoRR* abs/1802.05365. URL: `http://arxiv.org/abs/1802.05365`.

Radford, Alec et al. (2018). “Improving language understanding by generative pre-training”. In: URL: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

Ren, Hao and Hong Lu (2018). “Compositional coding capsule network with k-means routing for text classification”. In: *CoRR* abs/1810.09177. URL: http://arxiv.org/abs/1810.09177.

Rezaeinia, Seyed Mahdi, Ali Ghodsi, and Rouhollah Rahmani (2018). “Text Classification based on Multiple Block Convolutional Highways”. In: *CoRR* abs/1807.09602. URL: http://arxiv.org/abs/1807.09602.

Rios, Anthony and Ramakanth Kavuluru (2015). “Convolutional Neural Networks for Biomedical Text Classification: Application in Indexing Biomedical Articles”. In: *ACM-BCB ... ... : the ... ACM Conference on Bioinformatics, Computational Biology and Biomedicine. ACM Conference on Bioinformatics, Computational Biology and Biomedicine* 2015, pp. 258–267. DOI: 10.1145/2808719.2808746. URL: https://dl.acm.org/citation.cfm?doid=2808719.2808746.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). “Learning representations by back-propagating errors”. In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0.

Sachan, Devendra Singh, Manzil Zaheer, and Ruslan Salakhutdinov (2019). “Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function”. In: *AAAI*. URL: https://wvvw.aaai.org/ojs/index.php/AAAI/article/view/4672.

Shah, Amar, David Knowles, and Zoubin Ghahramani (2015). “An Empirical Study of Stochastic Variational Inference Algorithms for the Beta Bernoulli Process”. In: *PMLR* 37, pp. 1594–1603. URL: http://proceedings.mlr.press/v37/jozefowicz15.pdf.

Shih, Chin-Hong et al. (2017). “Investigating Siamese LSTM networks for text categorization”. In: *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 641–646. DOI: 10.1109/APSIPA.2017.8282104. URL: http://ieeexplore.ieee.org/document/8282104/.

Smith, Noah A. (2019). “Contextual Word Representations: A Contextual Introduction”. In: *CoRR* abs/1902.06006. URL: https://arxiv.org/abs/1902.06006.

Song, Xingyi, Johann Petrak, and Angus Roberts (2018). “A Deep Neural Network Sentence Level Classification Method with Context Information”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 900–904. DOI: 10.18653/v1/D18-1107. URL: https://www.aclweb.org/anthology/D18-1107.

Srivastava, Saurabh, Prerna Khurana, and Vartika Tewari (2018). “Identifying Aggression and Toxicity in Comments using Capsule Network”. In: *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pp. 98–105. URL: https://www.aclweb.org/anthology/W18-4412.

Sun, Chi et al. (2019). “How to Fine-Tune BERT for Text Classification?” In: *CoRR* abs/1905.05583. URL: http://arxiv.org/abs/1905.05583.

Svozil, Daniel, Vladimír Kvasnicka, and Jirí Pospichal (1997). “Introduction to multi-layer feed-forward neural networks”. In: *Chemometrics and Intelligent Laboratory Systems* 39.1, pp. 43–62. ISSN: 0169-7439. DOI: 10.1016/S0169-7439(97)00061-0. URL: https://www.sciencedirect.com/science/article/abs/pii/S0169743997000610?via%3Dihub.

Turian, Joseph, Lev-Arie Ratinov, and Yoshua Bengio (2010). “Word Representations: A Simple and General Method for Semi-Supervised Learning”. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 384–394. URL: https://www.aclweb.org/anthology/P10-1040.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vinayakumar, R. et al. (2017). “Deep Health Care Text Classification”. In: *CoRR* abs/1710.08396. URL: http://arxiv.org/abs/1710.08396.

Wang, Alex et al. (2019). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding”. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rJ4km2R5t7.

Wang, Baoxin (2018). “Disconnected Recurrent Neural Networks for Text Categorization”. In: pp. 2311–2320. DOI: 10.18653/v1/P18-1215. URL: https://www.aclweb.org/anthology/P18-1215.

Wang, Sida and Christopher Manning (2012). “Baselines and Bigrams: Simple, Good Sentiment and Topic Classification”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 90–94. URL: https://www.aclweb.org/anthology/P12-2018.

Wang, Yiren and Fei Tian (2016). “Recurrent Residual Learning for Sequence Classification”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 938–943. DOI: 10.18653/v1/D16-1093. URL: https://aclweb.org/anthology/D16-1093/.

Wei, Fusheng et al. (2019). “Empirical Study of Deep Learning for Text Classification in Legal Document Review”. In: *CoRR* abs/1904.01723. URL: http://arxiv.org/abs/1904.01723.

Xiao, Liqiang et al. (2018). “MCapsNet: Capsule Network for Text with Multi-Task Learning”. In: *Proceedings of the 2018 Conference on Empiri-cal Methods in Natural Language Processing*. URL: https://pdfs.semanticscholar.org/3757/dc14b4c901bccb94cc59f0b3940ef4ee984f.pdf.

Xiao, Yijun and Kyunghyun Cho (2016). “Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers”. In: *CoRR* abs/1602.00367. URL: http://arxiv.org/abs/1602.00367.

Xu, Binbin et al. (2019). “Neural Language Model for Automated Classification of Electronic Medical Records at the Emergency Room. The Significant Benefit of Unsupervised Generative Pre-training”. In: *ArXiv* abs/1909.01136. URL: https://arxiv.org/abs/1909.01136.

Xu, Haotian et al. (2016). “Text Classification with Topic-based Word Embedding and Convolutional Neural Networks”. In: *ACM-BCB 2016*, pp. 88–97. DOI: 10.1145/2975167.2975176. URL: http://dl.acm.org/citation.cfm?doid=2975167.

Yang, Pengcheng et al. (2018). “SGM: Sequence Generation Model for Multi-label Classification”. In: *CoRR* abs/1806.04822. URL: http://arxiv.org/abs/1806.04822.

Yang, Zhilin et al. (2019). “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. In: *CoRR* abs/1906.08237. URL: http://arxiv.org/abs/1906.08237.

Yang, Zichao et al. (2016). “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*, pp. 1480–1489. DOI: 10.18653/v1/N16-1174. URL: https://aclweb.org/anthology/N16-1174/.

Yin, Wenpeng et al. (2017). “Comparative Study of CNN and RNN for Natural Language Processing”. In: *CoRR* abs/1702.01923. URL: http://arxiv.org/abs/1702.01923.

Yogatama, Dani et al. (2017). “Generative and discriminative text classification with recurrent neural networks”. In: *CoRR*. URL: https://arxiv.org/abs/1703.01898.

Zhang, Xiang and Yann LeCun (2015). “Text Understanding from Scratch”. In: *CoRR* abs/1502.01710. URL: http://arxiv.org/abs/1502.01710.

Zhang, Xiang, Junbo Zhao, and Yann LeCun (2015). “Character-level Convolutional Networks for Text Classification”. In: pp. 649–657. URL: http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf.

Zhang, Ye and Byron C. Wallace (2015). “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification”. In: *CoRR* abs/1510.03820. URL: `http://arxiv.org/abs/1510.03820`.

Zhang, Yinyuan et al. (2018). “Multi-Label Learning from Medical Plain Text with Convolutional Residual Models”. In: *Proceedings of the 3rd Machine Learning for Healthcare Conference*, pp. 280–294. URL: `http://proceedings.mlr.press/v85/zhang18a.html`.

Zhang, Zhuosheng et al. (2019). “Semantics-aware BERT for Language Understanding”. In: URL: `https://arxiv.org/abs/1909.02209`.

Zhao, Wei et al. (2018). “Investigating Capsule Networks with Dynamic Routing for Text Classification”. In: *CoRR* abs/1804.00538. URL: `http://arxiv.org/abs/1804.00538`.

Zhao, Xiaoli, Shaofu Lin, and Zhisheng Huang (2018). “Text Classification of Microblog’s "Tree Hole" Based on Convolutional Neural Network”. In: *ACAI 2018 conference proceeding*, pp. 1–5. DOI: `10.1145/3302425.3302501`. URL: `http://dl.acm.org/citation.cfm?doid=3302425`.

Zheng, Jin and Limin Zheng (2019). “A Hybrid Bidirectional Recurrent Convolutional Neural Network Attention-Based Model for Text Classification”. In: *IEEE Access* 7, pp. 106673–106685. DOI: `10.1109/ACCESS.2019.2932619`. URL: `https://ieeexplore.ieee.org/document/8784247`.

Zhou, Chunting et al. (2015). “A C-LSTM Neural Network for Text Classification”. In: *CoRR* abs/1511.08630. URL: `http://arxiv.org/abs/1511.08630`.

Zhou, Peng, Zhenyu Qi, et al. (2016). “Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling”. In: *CoRR* abs/1611.06639. URL: `http://arxiv.org/abs/1611.06639`.

Zhou, Peng, Wei Shi, et al. (2016). “Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 207–212. DOI: `10.18653/v1/P16-2034`. URL: `https://aclweb.org/anthology/P16-2034/`.

A Appendix

A.1 Aivy

Unsere Challenge:

- Wir bekommen von verschiedenen Firmen Stellenangebote mit JOBTITLE und JOBDESCRIPTION (und weiteren Variablen). Dabei sind beide Variablen in der Regel unstrukturiert/unstandardisiert. Beispiele ist z.B. Aldi Nord: `https://www.aldi-nord.de/karriere/job-search.html?filters=&location=`
- Wir müssen diese JOBTITLES nun standardisieren bzw. kategorisieren: "Köche (m/w)", "Koch (w/m/divers)", "Saisonkoch in Gastro", "Eventkoch", "Diätkoch / Haushälter" oder "Küchenchef" müssen alle als "Koch/Köchin" kategorisiert werden. Das war ein einfaches Beispiel: Schau dir gerne mal "Vertriebler/in" auf Stepstone an, als weiteres Beispiel: `https://www.stepstone.de/5/ergebnisliste.html?stf=freeText&ns=1&qs=%5B%7B%22id%22%3A%22225456%22%2C%22description%22%3A%22Vertriebler%2Fin%22%2C%22type%22%3A%22jd%22%7D%5D&companyID=0&cityID=0&sourceOfTheSearchField=resultlistpage%3Ageneral&searchOrigin=Resultlist_top-search&ke=Vertriebler%2Fin&ws=&ra=30`

Aufgabenstellung:

1. Konzept für einen KI-Prozess / Data-Pipelines: Vergleiche Prozess von lengoo.de, die arbeiten da sehr sauber.
2. Auswahl geeigneter Machine Learning Methoden für das Pre-Processing von Stellenangeboten (Natural Language Processing) und die anschlieSSende Kategorisierung (supervised/unsupervised learning).
3. Diskussion verschiedener Frameworks und IT-Infrastrukturen für die Implementierung des KI-Prozess (wir arbeiten z.B. generell mit AWS, was viele Funktionen auch im ML mitbringt, aber auch recht teuer ist).
4. Umsetzung und Deployment des KI-Prozess und Training der KI (Dafür können wir Zugriff auf die Stepstone API geben. Das Training/Testing kann z.B. mit Stellenangeboten von Aldi Nord und weiteren
5. Unternehmen erfolgen. Die menschliche Korrektur können wir über Amazon MTurk o.ä. Services abbilden).
6. Ausblick auf potenzielle Verbesserungen des KI-Prozesses sowie der eingesetzten ML-Methoden geben.

A.2 Training Data XML

```
<job id="6098301">
        <standardJobtitle>Elektrofachkraft</standardJobtitle>
        <vocationId>2630</vocationId>
        <matchRank>13</matchRank>
        <title><![CDATA[Elektrofachkraft (m/w/d) im
           Gebäudemanagement]]></title>
        <totalClick><![CDATA[0]]></totalClick>
        <description><![CDATA[<p>Als eines der grö ten
           Verkehrsunternehmen in Südhessen wollen wir den
           Menschen in Darmstadt und im Landkreis Darmstadt-
           Dieburg ihre Mobilität so einfach wie möglich machen
           . Daher engagieren wir uns für einen attraktiven und
            leistungsfähigen Nahverkehr in der Region. Zur
           Verstärkung unseres Teams im Gebäudemanagement
           suchen wir eine Elektrofachkraft*.</p>  <p><strong>
           Wir suchen eine</strong></p>  <p><strong>
           Elektrofachkraft (m/w/d) im Gebäudemanagement</
           strong></p><br /><ul>   <li>Beseitigung von
           elektronischen Störungen in der Gebäudetechnik</li>
             <li>Installations-, Wartungs- und
           Instandhaltungsarbeiten an Elektroanlagen</li>   <li
           >Erstprüfungen nach DGUV-V3 von Elektrogeräten</li>
             <li>Unterstützung im Bereich des
           Gebäudemanagements bei allen anfallenden Arbeiten an
            Betriebsgebäuden und Au enstellen</li>  </ul><br
           /><ul>   <li>Abgeschlossene technische
           Berufsausbildung als Elektroniker* (bevorzugt
           Elektrofachkraft für Gebäudetechnik)</li>   <li>
           Mehrjährige Berufserfahrung im elektrotechnischen
           Bereich</li>   <li>Weiterbildung zur befähigten
           Person zur Prüfung von Elektrogeräten</li>   <li>Pkw
           -Führerschein</li>  </ul><br /><ul>   <li>
           Kollegiales Arbeitsumfeld mit kurzen
           Entscheidungswegen</li>   <li>Festanstellung |
           Gleitzeit | Individuelle Weiterbildungsmöglichkeiten
           </li>   <li>60 % Zuschuss zum Jobticket |
           Fahrradleasing | Fitnessstudiorabatte</li>   <li>
           Betriebliche Altersversorgung | Zuschüsse für
           Zahnersatz und Brillen</li>   <li>Kostengünstiges
           Betriebsrestaurant</li>  </ul>]]></description>
        <url><![CDATA[https://www.stepstone.de/stellenangebote
           ----20191016095639160---6098301-inline.html?cid=
            test_test_test]]></url>
         <Category id="10002000" type="FUNCTION"><![CDATA[
            Elektrotechnik, Elektronik]]></Category>
         <date><![CDATA[16.10.2019]]></date>
         <Category id="222" type="STATUTE"><![CDATA[Feste
            Anstellung]]></Category>
         <Category id="90002" type="EXPERIENCE"><![CDATA[Mit
            Berufserfahrung]]></Category>
         <Category id="80001" type="WORKTYPE"><![CDATA[Vollzeit
            ]]></Category>
         <location><![CDATA[Darmstadt]]></location>
         <postalcode><![CDATA[64285]]></postalcode>
         <jobAddress1><![CDATA[Klappacher Stra e 172]]></
            jobAddress1>
         <jobAddress2><![CDATA[]]></jobAddress2>
         <jobAddress3><![CDATA[]]></jobAddress3>
         <jobCity><![CDATA[Darmstadt]]></jobCity>
         <companyName><![CDATA[HEAG mobilo GmbH]]></companyName>
         <geokoordinaten>
         <latitude><![CDATA[49.85222625732422]]></latitude>
         <longitude><![CDATA[8.668535232543945]]></longitude>
         </geokoordinaten>
         <company_logo><![CDATA[http://www.stepstone.de/
            upload_DE/logo/H/logoHEAG_mobilo_GmbH_68344DE.gif]]>
            </company_logo>
         <region><![CDATA[D-PLZ 64]]></region>
         <sector id="17000"><![CDATA[Transport & Logistik]]></
            sector>
</job>
```

Listing A.1: Training Data XML

A.3 Term Table

Table A.1: Term Table

Term	General Term	Subsumable Term	Synonym	Related Term
Classification		Multi-Class Classification: Single-Label Classification: Multi-Label Text Classification: Large Scale Multi-Label Text Classification: Extreme Multi-Label Text Classification	Categorization	Prediction
Text		Sentence: Paragraph: Document: Corpus: Report		
Natural Language Processing	Computer Science: Machine Learning: Data Science: Machine Perception	Sentiment Analysis: Machine Translation: Natural Language Generation: Question Answering: Named Entity Recognition: Dependency Parsing		Text Mining: Computational Linguistics

Table A.1: Term Table

Term	General Term	Subsumable Term	Synonym	Related Term
Neural Network	Computer Science: Data Science: Artificial Intelligence	MLP: FNN: RNN: CNN: SNN: PCNN: LSTM: GRU: GAN; Tree RNN: Attention: Transformer: Deep Learning: HighWayNets: ResNets: CRNN: RCNN: mLSTM: mRNN: Attention: DenseNets: CapsNets: Inception: DNN: ANN	Deep Neural Networks	Deep Learning: Neural

A.4 Selection Criteria

- Selected:
  - Classification of a corpus according to its content
  - realized with any neural model
- Excluded
  - ensembles of neural and non neural models
  - non neural models like Bayesian models, SVM, TF-IDF, etc.
  - papers not publicly available

A.5 Concept Matrix

Table A.2: Concept Matrix

Author	**CNN**	**RNN**	**Transformer**	**Misc.**
C. Zhou et al. 2015	x	x		x
Kabir and Madria 2019				x
Paisios et al. 2019	x			
Song, Petrak, and Roberts 2018				x
J. Zheng and L. Zheng 2019	x	x		x
Asim et al. 2019	x			
Ye Zhang and Wallace 2015	x			
Z. Lin et al. 2017	x	x		
LeClair, Eberhart, and McMillan 2018				x
Lan et al. 2019			x	
Liang, Wu, and C. Huang 2019	x	x		
Vaswani et al. 2017			x	
P. Zhou, Shi, et al. 2016	x	x		
Lyubinets, Boiko, and Nicholas 2018				x
Joulin, Grave, Bojanowski, and Mikolov 2016				
Clark et al. 2019			x	
Cliche 2017	x	x		
Devlin et al. 2018			x	
Baker, Korhonen, and Pyysalo 2016	x			
X. Zhang, J. Zhao, and LeCun 2015	x			
Y. Lee, Yoon, and Jung 2018	x	x		
Kamath, Bukhari, and Dengel 2018	x			
Yin et al. 2017	x	x		
Ren and Lu 2018				x
Amin and Nadeem 2018	x			

Table A.2: Concept Matrix

Author	CNN	RNN	Transformer	Misc.
Rios and Kavuluru 2015	x			
Li et al. 2017	x			
Y. Kim 2014	x			
Hassan and Mahmood 2018				x
Jiang et al. 2019	x	x		
Baziotis, Pelekis, and Doulkeridis n.d.		x		
Peters et al. 2018		x		
Jingshu Liu, Zachariah Zhang, and Razavian 2018	x	x		
Vinayakumar et al. 2017		x		
Jingzhou Liu et al. 2017	x			
Hassan and Mahmood 0207		x		
Lenc and Král 2017	x			
Johnson and T. Zhang 2017	x			
Iyyer et al. 2015	x			x
Mani, Sankaran, and Aralikatte 2018		x		
B. Wang 2018	x	x		x
Le Hoa, Cerisara, and Alexandre 2017	x	x		
Johnson and T. Zhang 2014	x			
Y. Xiao and Cho 2016	x			x
Hassan and Mahmood n.d.	x	x		x
Wei et al. 2019	x			
G. Chen et al. 2017				x
Mullenbach et al. 2018	x			
Chalkidis, E. Fergadiotis, et al. 2019	x	x		
Joulin, Grave, Bojanowski, Douze, et al. 2016				

Table A.2: Concept Matrix

Author	CNN	RNN	Transformer	Misc.
K.-M. Kim et al. 2019	x	x		
Yogatama et al. 2017	x			x
Zichao Yang et al. 2016				x
Gao, Ramanathan, and Tourassi 2018	x			x
Sun et al. 2019			x	
Radford et al. 2018			x	
W. Zhao et al. 2018				x
Shih et al. 2017	x	x		
Chalkidis, M. Fergadiotis, et al. 2019	x	x	x	
T. Huang, Shen, and Deng 2019		x		
McCann et al. 2017		x		
Howe, Khang, and Chai 2019	x		x	
Zhengjie Huang et al. 2017		x		
Parwez, Abulaish, and Jahiruddin 2019	x			
Baumel et al. 2017	x			x
Yinyuan Zhang et al. 2018	x			
B. Xu et al. 2019			x	
Chalkidis, Androutsopoulos, and Aletras 2019		x	x	
Dernoncourt, J. Y. Lee, and Szolovits 2017		x		
Camacho-Collados and Pilehvar 2017	x			x
Park and Fung 2017	x			
Hsu and Yaoqin Zhang 2018	x	x		
Kant et al. 2018		x	x	
Lai et al. 2015	x			
P. Liu, Qiu, and X. Huang 2016		x		
Y. Wang and Tian 2016		x		

Table A.2: Concept Matrix

Author	CNN	RNN	Transformer	Misc.
Giannakopoulos et al. 2019				x
Adhikari et al. 2019	x	x		x
Sachan, Zaheer, and Salakhutdinov 2019		x		
Kowsari et al. 2018	x	x		
Y. Liu et al. 2019			x	
Arnold et al. 2019		x		
Zhuosheng Zhang et al. 2019			x	
J. Lin et al. 2018				x
Johnson and T. Zhang 2015	x			
A. M. Dai and Le V 2015				
Madasu and Rao 2019				x
J. Y. Lee and Dernoncourt 2016	x	x		
P. Yang et al. 2018	x	x		x
Hu et al. 2018	x			
Joshi et al. 2019			x	
Johnson and T. Zhang 2016				x
Rezaeinia, Ghodsi, and Rahmani 2018	x			
P. Zhou, Qi, et al. 2016				x
X. Zhao, S. Lin, and Zhisheng Huang 2018	x			
H. Xu et al. 2016	x			
X. Zhang and LeCun 2015	x			
Krause 2017	x			
Y. Liu et al. 2019			x	
Howard and Ruder 2018			x	
Cer et al. 2018			x	x
Conneau et al. 2016	x			

Table A.2: Concept Matrix

Author	CNN	RNN	Transformer	Misc.
Meng et al. 2018	x			
Chang et al. 2019			x	
Zhilin Yang et al. 2019			x	

A.6 Hyperparameters for Fine-Tuning

Table A.3 lists the hyperparameters used to fine-tune XLNet on the RACE, SQUAD, MNLI and Yelp-5 task. Layer-wise decay refers to exponentially decaying the learning rates for each layer from highest to lowest. Given an initial learning rate l, a layer-wise decay rate α and M layers, the learning rate of the layer m is $l\alpha M - m$.[115]

Table A.3: Hyperparameters for Fine-Tuning

Hyperparameter	RACE	SQUAD	MNLI	Yelp-5
Dropout	0.1	0.1	0.1	0.1
Attention dropout	0.1	0.1	0.1	0.1
Max sequence length	512	512	128	512
Batch size	32	48	128	128
Initial learning rate	2×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Number of steps	$12K$	$8K$	$10K$	$10K$
Learning rate decay	linear	linear	linear	linear
Weight decay	0	0	0	0
Adam epsilon	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Layer-wise decay rate	1.0	0.75	1.0	1.0

[115]Zhilin Yang et al. 2019.

Declaration of Authorship

I hereby declare that the paper submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. Furthermore, I assure that the electronic version submitted corresponds to the printed version.

Bad Dürkheim. 26 11 2019

[illegible]

Place / Date

Fabian Wolf