Transfer Learning for Textual Topic Classification

Pavel Janata

Supervisor: Ing. Jiří Čermák, Ph.D.
Field of study: Open Informatics
Subfield: Computer and Informatic Science
May 2019
BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: Janata Pavel
Personal ID number: 465810
Faculty / Institute: Faculty of Electrical Engineering
Department / Institute: Department of Cybernetics
Study program: Open Informatics
Branch of study: Computer and Information Science

II. Bachelor's thesis details

Bachelor's thesis title in English:
Transfer Learning for Textual Topic Classification

Bachelor's thesis title in Czech:
Transfer learning pro klasifikaci textu

Guidelines:
Recently a significant result was achieved by using transfer learning in natural language processing (NLP) [1]. The main breakthrough was the use of a model pre-trained on Wikipedia corpus to obtain state-of-the-art performance on a classification of textual data in a different dataset.
The student will verify that the performance of this approach is consistent on a different dataset containing textual data along with their classes. The goal of the thesis will be accomplished by successfully performing the following steps:
• Study the state-of-the-art approaches to transfer learning in the field of NLP.
• Analyze existing datasets containing textual data and their corresponding class labels (e.g., Routers Dataset) and choose the one most suitable for validating the approach.
• Use the existing pretrained model "Wikitext 103" provided by FastAI. Fine-tune the model on the dataset chosen in the previous task.
• Evaluate the performance of the model on this dataset and compare it to the results published in [1].

Bibliography / sources:

Name and workplace of bachelor’s thesis supervisor:
Ing. Jiří Čermák, Ph.D., Blindspot Solutions, Prague

Name and workplace of second bachelor’s thesis supervisor or consultant:
doc. Ing. Jiří Vokřínek, Ph.D., Artificial Intelligence Center, FEE

Date of bachelor’s thesis assignment: 14.01.2019
Deadline for bachelor thesis submission: 24.05.2019
Assignment valid until: 30.09.2020

Ing. Jiří Čermák, Ph.D.
Supervisor's signature
doc. Ing. Tomáš Svoboda, Ph.D.
Head of department's signature
prof. Ing. Pavel Ripka, CSc.
Dean's signature
The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt
Student’s signature
Acknowledgements

Most of all, I would like to thank my supervisor, Jiří Čermák, who guided me throughout writing this thesis as well as to Štěpán Kopřiva and all the great people at Blindspot.ai.

I am grateful to all my friends and family: my parents and grandparents, my sister, and my wonderful girlfriend for always supporting me.

Finally, I want to thank the following coffee roasters, which provided me with the much-needed fuel: Pražírna, doubleshot, KK, Dos Mundos, and Coffee Source.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of the university theses.

..............................
Pavel Janata
Prague, 24th May 2019
Abstract

The recent developments of Language Modeling led to advances in transfer learning methods in Natural Language Processing. Language Models pretrained on large general datasets achieved state-of-the-art results in a wide range of tasks. The Universal Language Model Fine-tuning represents an effective transfer learning method for text classification. The goal of this thesis is to further test the robustness of this method in scenarios, commonly found in real-world applications.

Keywords: natural language processing, text classification, artificial neural network, machine learning, transfer learning

Supervisor: Ing. Jiří Čermák, Ph.D. Blindspot Solutions, Praha

Abstrakt


Klíčová slova: zpracování přirozeného jazyka, klasifikace textu, umělá neuronová síť, strojové učení

Překlad názvu: Transfer learning pro klasifikaci textu
## Contents

1 Introduction .......................... 1
   1.1 Motivation ..................... 1
   1.2 Goals of the thesis .......... 2
   1.3 Thesis outline ............... 2

2 Technical Background ............. 3
   2.1 Categorization of learning task 
      based on the training dataset ... 3
   2.2 Artificial Neural Networks .... 4
      2.2.1 Feedforward Neural Networks. 4
      2.2.2 Recurrent Neural Networks . 5
      2.2.3 Long Short Term Memory .... 5
      2.2.4 Training of Artificial Neural 
          Networks .................... 8
   2.3 Transfer learning .......... 8

3 Universal Language Model 
   Fine-tuning .......................... 11
   3.1 Structure of the model ...... 11
   3.2 Training of the model ..... 11
      3.2.1 Tokenization ............. 12
      3.2.2 Pretraining the Language 
          Model ........................ 13
      3.2.3 Fine-tuning ............... 13
      3.2.4 Classifier Training .... 14

4 Experimental setup ............... 15
   4.1 Classification dataset ...... 15
      4.1.1 Dataset Analysis ........ 15
      4.1.2 Baseline results ......... 16
   4.2 Hyperparameters ............ 16

5 Experiments ......................... 19
   5.1 Impact of the size of the training 
      dataset on the classification 
      accuracy ....................... 19
   5.2 Impact of the size of the training 
      dataset on the classification accuracy, 
      with the language model fine-tuned 
      on the whole dataset .......... 20
   5.3 Performance of the ULMFiT model 
      on an imbalanced dataset – with ten 
      classes unbalanced .............. 21
   5.4 Performance of the ULMFiT model 
      on an imbalanced dataset – with two 
      classes unbalanced .............. 22

6 Conclusion .......................... 27
   6.1 Future Work ................. 27

A Detailed results of the experiments 29
B Contents of the CD 33
C Bibliography 35
Chapter 1
Introduction

1.1 Motivation

Natural Language Processing has been a subject of research for many decades, with aims to improve human-computer interaction. As we generate more and more data in all forms, apprehending it is too big of a task for any human. Thus systems capable of comprehending texts in the natural language are necessary in a wide array of real-life applications.

Text classification is one of the fundamental tasks in Natural Language Processing. Its difficulty lies in recognizing the intricate semantic structures, as well as the necessity of understanding the background information and other complexities inherent in the natural languages. Deep learning proved to be an effective approach in other NLP fields such as Machine Translation [27]. However, one of the main obstacles in advancing the classification systems appears to be the lack of large labeled datasets.

As insufficient training data is a common problem in machine learning, various techniques were proposed to solve this issue — one of which being transfer learning, which has seen successful adoption in computer vision. This method attempts to capture a knowledge gained by learning problem, to be then reused on other, somewhat related problem. This allows for deployment on much smaller datasets, as the model already poses some general knowledge, which it gained on the previous task.

In recent years, there has been a lot of development in transfer learning methods for Natural Language Processing. While in the past, it had been used mainly for simple tasks as word embedding [10], the recent developments in language modeling, enabled much more advanced applications of transfer learning. Generally, these new models are trained on extensive unlabelled datasets to be then applicable to a wide range of tasks.

Namely, the language model developed by OpenAI [15], which was trained on text data scraped from the internet, has been able to achieve state-of-the-art results on many language modeling datasets. They have also demonstrated its ability to generate realistic and coherent texts, when prompt with an input text. Such a tool could be useful, for instance, when writing a bachelor’s thesis. However, due to concerns of the possible malicious uses, at the time of publishing their paper, OpenAI has decided not to release the full trained...
models and make available to the public only smaller versions of it. In this work, I have decided to use the Universal Language Model Fine-tuning (ULMFiT), designed by Jeremy Howard and Sebastian Ruder from fast.ai [8], as it is the best available model for this task. Their general Language Model trained on Wikitext-103 dataset (28,595 articles from English Wikipedia), can be fine-tuned for specific classification tasks. It achieves the state-of-the-art results on multiple classification datasets and with its use of unsupervised Language Model fine-tuning, it promises good performance even on semi-supervised datasets, with just a few labeled examples.

1.2 Goals of the thesis

In this work, I will test the robustness and dexterity of the Universal Language Model Fine-tuning by evaluating it in several scenarios. For this, I have selected the 20 Newsgroup dataset to serve as a base for my experiments. I will measure the impact of the size of the training dataset on the model’s performance. Since the model can be trained in semi-supervised mode, I will measure the effectiveness of this method and compare its results to the model trained in fully supervised mode.

Furthermore, as unbalanced datasets are very prevalent in real-world applications, and often lead to undesirable behavior of Machine Learning models, I will test the ULMFiT’s ability to cope with those scenarios, by introducing a severe imbalance into the training dataset and analyzing its results on it.

1.3 Thesis outline

In the next chapter, I present a review of the methods and techniques underlying the studied model. The Universal Language Model Fine-tuning, which serves as the bases of this thesis, is described in Chapter 3. Chapter 4 describes the used dataset and setup of the experiments described in Chapter 5. Finally, Chapter 6 contains the evaluation of the results of the experiments along with propositions of possible future works.
In this chapter, I will characterize some of the methods and concepts used in this work. I present the basic terminology used further in this thesis, as well as a brief description of artificial neural networks, focusing mainly on the architectures related to this work.

### 2.1 Categorization of learning task based on the training dataset

Learning tasks are often categorised based on the used training datasets [18][2]. Supervised learning is a collection of tasks, where the data is labeled, i.e., each entry is provided with some desired output.

When the dataset lacks labels, we call it unsupervised learning. In such tasks, new inputs can be categorised only based on the patterns inferred from the training dataset. Typical instances of unsupervised learning are clustering, anomaly detection, or language modeling, which is a task of assigning probabilities to tokens in a sequence of words.

Finally, semi-supervised learning is a combination of above-mentioned approaches. In the real world applications, obtaining a large labeled dataset is very difficult, thus being able to use data where a small amount of labeled entries is complement with larger quantities of unlabeled ones can be very convenient.

In this work, I focus on the semi-supervised classification.

### 2.2 Artificial Neural Networks

Artificial Neural Network (ANN) is a machine learning system, first conceived to mimic the function of biological neural networks. Today, we use many different types of neural nets in a wide array of tasks such as classification, regression, anomaly detection and many more.

In this section, I briefly cover the basic structure of neural networks and outline some of their architectures. I describe Feedforward Neural Networks (Section 2.2.1), one of the most basic types of ANNs then in Section 2.2.2 I present Recurrent Neural Networks, which allow processing of sequential...
2. Technical Background

Figure 2.1: Example of neuron used in feedforward neural networks with the input vector \( x \), vector of weights \( w_{ij} \), output value \( o_j \) and activation function \( \varphi \) (for examples of commonly used activation function see Figure 2.2). [25]

Data. In Section 2.2.3, I characterize the Long Short-Term Memory, a widely used version of Recurrent Neural Network, which is used in models achieving a state of the art results in speech recognition, natural language processing or time series prediction.

2.2.1 Feedforward Neural Networks

Feedforward Neural Networks (FNN) [20] are composed of individual units called neurons (shown in Figure 2.1). Each neuron can be viewed as functions producing an output value \( o_j \) given by the equation

\[
o_j = \varphi \left( \sum_{i=0}^{n} x_i w_{ij} + b_j \right)
\]  

(2.1)

where \( x_i \) are input values, \( w_{ij} \) are the weights of the neuron, \( b_j \) is its bias term, and \( \varphi \) is some non-linear activation function. It is the non-linearity of the activation function, which allows the neural networks to model complex decision boundaries. Examples of some commonly used activation functions can be found in Figure 2.2.

The neurons are structured into layers. Neural networks usually consist of input and output layers with possibly multiple hidden layers in-between (see Figure 2.3). In Feedforward Neural Networks, information travels only in the direction from the input layer to the output layer, meaning that the output of neurons connects to the inputs of neurons in the subsequent layer.

When an output of each neuron in a layer feeds into every neuron in the next layer, we call those layers fully connected.
2.2. Recurrent Neural Networks

Recurrent Neural Networks (RNN) differ from regular Feedforward Neural Networks in that the neurons do not always connect to the next layer, but instead, they may be connected to one of the previous layers. These connections can form a type of memory allowing the network to retain knowledge over multiple inputs. This makes it well suited for processing of sequential data like text, speech or time series.

However, it was shown [3] that because of the manner in which they are trained, the general Recurrent Neural Networks struggle to retain knowledge over longer sequences of inputs. This led to the development of several RNN architectures trying to fix this issue.

2.2.3 Long Short Term Memory

The Long Short Term Memory (LSTM), introduced by Hochreiter and Schmidhuber [7], tries to solve the mentioned problems of the general RNNs. They have structured the network (see Figure 2.4) into units, responsible for processing the sequential data and keeping the inner state.

At each time step $t$, the LSTM unit receives an input vector $\mathbf{x}_t$ as well as its output from the previous time step, hidden state $\mathbf{h}_{t-1}$. To store its internal state, it uses the cell state $\mathbf{c}$. Inside, the LSTM module consists of a memory cell, forgot gate, input gate, and an output gate. The forget and input gates modify the cell state based on the module’s input. As the names
suggest, the forget gate controls the conservation of the information in the memory cell, and the input gate allows incorporation of new information. Augmenting the previous cell state \( c_{t-1} \) by those gates produces new state \( c_t \). The normalised copy of the cell state can then be filtered by the output gate, which creates the output hidden state \( h_t \) of the module.

The following notation is used in deriving the activation and output vectors. Vectors and matrices are represented by bold letters, with matrices being in upper case. The subscripts indicate the corresponding gate or time step.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>time step</td>
</tr>
<tr>
<td>( c )</td>
<td>cell state vector</td>
</tr>
<tr>
<td>( x )</td>
<td>input vector of the LSTM module</td>
</tr>
<tr>
<td>( h )</td>
<td>hidden state vector</td>
</tr>
<tr>
<td>( i )</td>
<td>activation vector of the input gate</td>
</tr>
<tr>
<td>( f )</td>
<td>activation vector of the forget gate</td>
</tr>
<tr>
<td>( o )</td>
<td>activation vector of the output gate</td>
</tr>
<tr>
<td>( W )</td>
<td>weight matrices</td>
</tr>
<tr>
<td>( b )</td>
<td>bias terms</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>sigmoid function</td>
</tr>
<tr>
<td>( \text{tanh} )</td>
<td>hyperbolic tangent function</td>
</tr>
<tr>
<td>( \circ )</td>
<td>element-wise multiplication</td>
</tr>
</tbody>
</table>
2.2. Artificial Neural Networks

Figure 2.4: LSTM module with the gates highlighted. The yellow markers denote element-wise operations, and the orange blocks are layers with the corresponding activation function – \( \text{tanh} \) denoting hyperbolic tangent and \( \sigma \) marking sigmoid function (see Figure 2.2).

These equations describe the activation functions of the individual gates. Note that as the input gate is composed of two separate layers, two sets of weights and bias terms are used (denoted \( W_f, b_f, W_i', b_i' \)).

\[
f_t = \sigma \left( W_f \cdot [x_t, h_{t-1}] + b_f \right)
\]
(2.2)

\[
o_t = \sigma \left( W_o \cdot [x_t, h_{t-1}] + b_o \right)
\]
(2.3)

\[
i_t = \sigma \left( W_i \cdot [x_t, h_{t-1}] + b_i \right) \circ \text{tanh} \left( W_i' \cdot [x_t, h_{t-1}] + b_i' \right)
\]
(2.4)

Using these the cell state \( c_t \) of the LSTM module can be derived as

\[
c_t = (c_{t-1} \circ f_t) + i_t
\]
(2.5)

and finally the output hidden state \( h_t \) as

\[
h_t = \text{tanh}(c_t) \circ o_t
\]
(2.6)

Stacking multiple LSTM units can lead to each unit operating at different time scale and therefore, capturing different levels of abstraction of the input \([6]\). For example in text processing, one module can learn to recognise relations between individual words, while other operates on the level of sentences. This approach was adopted in the design of the ULMFiT model which is used in this work.
2. Technical Background

2.2.4 Training of Artificial Neural Networks

To train the network for a specific task, a set of observations is used, for which an expected output is known. This data is then fed into the model, and its output for each data point is compared to the ground truth. The expected and actual outputs serve as input to the loss function, which measures the dissimilarity of its inputs. In this work, the cross-entropy loss function $L_C$ is used, which for a set of classes $X$ is defined as

$$L_C = -\sum_{x \in X} y_x \ln(p_x)$$

(2.7)

where $y$ are the expected outputs and $p$ the predicted outputs.

The learning is achieved by adjusting the weights of the model. This is done by the gradient descent method, which models the task of training the network as an optimisation problem of finding a minimum of the loss function. Each iteration $t$, the weights $w_j$ are updated using the following equation:

$$w_{ij}(t + 1) = w_{ij}(t) - \eta \frac{\partial L}{\partial w_{ij}}$$

(2.8)

where $\eta$ is the learning rate and $L$ the loss function. The backpropagation algorithm [17] is used, to obtain the gradient of the loss function. In practice, the gradient descent is not performed on the whole dataset at once, but instead, it is split into batches of arbitrary size. This allows for much faster iterations, and the algorithm can be more memory efficient.

Because of the recurrent connections in RNN, the standard backpropagation algorithm cannot be used in this class of neural networks. Instead, its generalised version called backpropagation through time is employed to calculate the gradients of weights. The network is first unfolded through time, and then the error is backpropagated through the standard feedforward connections as well as through the recurrent connections. In this process, updated weights are calculated for each time step. However, as only one set of weight can be used in for the gradient descent, the values have to be aggregated. This process often leads to vanishing gradient problem [3] [13], as some of the weights can be very far from the optimal solution, but the gradient being vanishingly small prevents them from changing their value. Special architectures were proposed, to combat this problem, for example, the LSTM discussed in Section 2.2.3.

2.3 Transfer learning

Transfer learning [12] refers to an effort of applying knowledge gained from a problem to boost learning on a related problem. In neural networks (described in Section 2.2), this is usually achieved by removing the last few layers, which are generally most tied to the specific task, and reusing the parts responsible for feature extraction and embedding.
2.3. Transfer learning

The most significant advantage of this approach is that training neural networks from scratch can be very computationally expensive, thus being able to train a model on a large dataset and then fine-tuned for a wide array of different tasks is hugely beneficial. However, one of the obstacles which, in some areas (e.g., Natural Language Processing [11]), proved to be difficult to overcome, is the ability to retain the previously obtain knowledge, as too aggressive training can destroy it.
Chapter 3

Universal Language Model Fine-tuning

Universal Language Model Fine-tuning (ULMFiT) is a state-of-the-art for text classification. The model was introduced in a paper by Howard and Ruder [8] from fast.ai.

In their work, they have managed to develop an effective transfer learning design, allowing use of Language Models trained on large general datasets, to then be fine-tuned for classification tasks on different, much smaller ones.

In this chapter, I first present the basic structure of the ULMFiT model (in Section 3.1), and then I describe the process of creating the Language Model (LM) and training the classifier in Section 3.2.

3.1 Structure of the model

The ULMFiT model used for classification (as shown in Figure 3.1c) consists of two main units – the encoder and the linear classifier.

The encoder contains an embedding layer and three stacked LSTM modules. The embedding layer receives an input word in one-hot encoding and passes it as a 400-dimensional vector to the first LSTMs. The following modules take the hidden state of the previous modules as their input. The hidden state of the final LSTM is then fed into the linear classifier which consists of a ReLU layer with 50 activations and the final softmax layer, whose size depends on the number of classes in the classification problem.

The LSTM architecture used in this model is an implementation of AWD-LSTM (which stands for ASGD Weight-Dropped LSTM), introduced by Merity Keskar and Socher [9], which thanks to its many regularization and optimization techniques, achieves state-of-the-art results in Language Modeling tasks.

3.2 Training of the model

In this section, I will walk through the necessary steps, to train the language model from scratch to the final classifier.

As shown in Figure 3.2, the training is split into three main steps. First, the Language Model is trained on a large general dataset (see Section 3.2.2
3. Universal Language Model Fine-tuning

(a) : Pre-training on the general-domain dataset - The output of the last layer corresponds to the probability of the next word in the sequence.

(b) : Fine-tuning on the target-domain dataset - The input and output layer is adjusted to fit the size of dataset’s dictionary.

(c) : Before the final training step, the last block is replaced by a linear classifier.

**Figure 3.1:** A high-level overview of the structure of the model during the training steps. The numbers in the brackets correspond to the size input/output size of the layers. [5]

**Figure 3.2:** Overview of the training steps. [5]

for more details). The pretrained LM is then fine-tuned (Section 3.2.3) on task’s target data, and finally, the classifier is trained (Section 3.2.4) on a labeled dataset.

As the model is trained to perform different task in each of the training steps, the structure also changes. The high-level overview of the model throughout the learning steps can be seen in Figure 3.1.

### 3.2.1 Tokenization

When using the model, the training and validation datasets are first pre-processed. The `fastai.text` library [1] provides all the necessary tools. It introduces special words (tokens) into the dataset marking the beginnings of the texts, capital letters, and words not present in the dictionary.

Then dictionary of used words is created, where each token is assigned a unique id. Only the most common words are included in the dictionary, the size of which can be configured. For my dataset, I have used a dictionary
size of 30,000 words which was recommended in the documentation, and it proved to be sufficient. Words not present in the dictionary are replaced by the *unknown* token. Finally, all the words in the texts are mapped to their ids and saved as arrays of integers.

### 3.2.2 Pretraining the Language Model

Howard and Ruder [8] published their model pre-trained on the huge *Wikitext-103* dataset which allows for its reuse in different tasks. However, the model can be pre-trained on any other large enough datasets.

The idea of pre-training is to gain knowledge of the domain and capture it in the universal Language Model, which could then be used on specific tasks. While the Language Model is trained to predict the next word in a sequence, it should also learn the general properties of the language.

The structure of the model during this step can be seen in Figure 3.1a. First, all words, represented by their ids, are passed to the embedding layer, using the one-hot encoding, outputting a 400-dimensional vector which is fed into 3 LSTMs and then finally, to the decoder, which should predict the next word, again, using the one-hot encoding. The *cross-entropy* loss function is used for this training.

### 3.2.3 Fine-tuning

Howard and Ruder [8] present fine-tuning of the Language Model as a crucial stage for a successful transfer. As the target dataset most certainly differs from the general dataset which was used for creating the LM. The structure of the model remains mostly unchanged (see Figure 3.1b), except the encoding and decoding layers are altered to fit the used dictionary. The Language Model is then trained to predict the next word in a sequence on the target task dataset, in the same manner as described in the previous section.

Although fine-tuning already was a popular approach in different fields of machine learning (e.g., computer vision), there was no such effective method in NLP [11]. Most of the previous attempts suffered from overfitting or catastrophic forgetting. Howard and Ruder [8] introduced several improvements, which make this method viable.

One of them is *discriminative fine-tuning*, which uses different learning rates for different layers of the model, compared to the traditional approach, where a single value is used for all layers. This method is based on the observation, that different layers gain a different kind of knowledge. They have achieved the best results by setting the LR of the last layer to some specific value (0.01 in their case) and then setting the learning rate of each preceding layer to \( \frac{1}{\sqrt{6}} \) times the LR of the subsequent layer.

Another improvement they have introduced are *slanted triangular learning rates*, shown in Figure 3.3, which are modified version of triangular learning rates introduced by Leslie N. Smith [22]. Instead of using a constant value for learning rates, in this technique, they are first linearly increased for faster
convergence into the desired region of the parameter space, and then gradually decreased to gain the necessary precision.

Because, during the fine-tuning the Language Model is again trained only to predict next word in a sequence, it does not require a labeled dataset. This allows for a semi-supervised learning use case.

### 3.2.4 Classifier Training

Finally, the model is trained for the target classification task. For this, two additional linear blocks are added (Figure 3.1c), each with batch normalization and dropouts. ReLU activations are used in the intermediate layer, and softmax activations provide probability distribution over target classes. The dimension of the last layer then corresponds to the number of classes in the target task.

To avoid catastrophic forgetting and losing knowledge the model has gained during previous stages, Howard and Ruder used a technique called gradual unfreezing, where at the beginning only the last layer of the model is trained, and weights of the other layers are frozen. Each epoch another layer is unfrozen until the whole model is being trained.
Chapter 4

Experimental setup

In this chapter I present the setup of the experiments described in Chapter 5 as well as the used 20 Newsgroups dataset.

4.1 Classification dataset

Howard and Ruder [8] conducted experiments on several datasets, including some intended for topic classification (AG news and DBpedia [28]). In my experiments, I mainly wanted to test the robustness of the model. For this, I have chosen the 20 Newsgroups dataset.

Specifically, I have used corpus created by Ana Cardoso Cachopo and published as a part of her Ph.D [4] thesis. She did some preprocessing on the data, like removing special characters, replacing multiple white spaces with a single space, transforming it to lower case, and prepending title to each text. The reason I chose her version of the datasets is that she also provides results of some of the common classification methods on this dataset, which I use as a baseline (see Section 4.1.2) in my experiments.

4.1.1 Dataset Analysis

There are almost 19,000 texts, each belonging into one of the 20 categories. The classes are quite well balanced as shown in Table 4.1. The corpus uses an unusual split, where about 40% of the data is used for validation. However, because this split has been used in other works before, I have decided not to change it.

Scikit-learn [14], which also provides tools for working with this dataset [21], recommend stripping the texts of their metadata, such as headers, footers, and quotes. However, after doing this, I have concluded, that especially deleting the quotations, removes too much of the information, to the point, when a considerable amount of texts become empty. Example of a text with quotations striped is shown in Table 4.2.
4. Experimental setup

<table>
<thead>
<tr>
<th>Class</th>
<th>Number of training texts</th>
<th>Number of validation texts</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>480</td>
<td>319</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>584</td>
<td>389</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>572</td>
<td>394</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>590</td>
<td>392</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>578</td>
<td>385</td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>593</td>
<td>392</td>
</tr>
<tr>
<td>misc.forsale</td>
<td>585</td>
<td>390</td>
</tr>
<tr>
<td>rec.autos</td>
<td>594</td>
<td>395</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>598</td>
<td>398</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>597</td>
<td>397</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>600</td>
<td>399</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>595</td>
<td>396</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>591</td>
<td>393</td>
</tr>
<tr>
<td>sci.med</td>
<td>594</td>
<td>396</td>
</tr>
<tr>
<td>sci.space</td>
<td>593</td>
<td>394</td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>598</td>
<td>398</td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>545</td>
<td>364</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>564</td>
<td>376</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>465</td>
<td>310</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>377</td>
<td>251</td>
</tr>
</tbody>
</table>

Table 4.1: Amount of training and validations texts per class in the 20 Newsgroups dataset

### 4.1.2 Baseline results

Along with the dataset Ana Cardoso Cachopo [4] published results of some classification methods on it. The measured accuracies are shown in Table 4.3, together with the classification result of the ULMFiT model trained on the whole unaltered dataset. The acquired value is an average accuracy on the validation set achieved over ten trainings. Example of the classification result of the ULMFiT model is also shown in the confusion matrix in Figure 4.1.

### 4.2 Hyperparameters

To ensure comparable results, I have done no special tuning of the ULMFiT model for this dataset and preserved mostly the same values of hyperparameters as Howard and Ruder [8] used. I have kept the same batch size of 64, and same base learning rates (0.004 for the language model fine-tuning and 0.01 for the classifier training). The dictionary size for the tokenization (see Section 3.2.1) was set to 30,000. I have fine-tuned the model and trained the classifier for 35 epochs. The reported accuracies are the best classification accuracies achieved during the training. Each experiment was repeated multiple times to ensure statistical significance.
This appeared today in the
> Japan Economic Journal reported
> GM plans to build a Toyota-badged
> car in the US for sale in Japan.
> Bruce MacDonald, VP of GM Corporate
> Communications, yesterday confirmed
> that GM President and CEO Jack Smith
> had a meeting recently with Tatsuro
> Toyota, President of Toyota. this
> meeting the two discussed business
> opportunities to increase GM exports
> to Japan, including further
> component sales as well as completed
> vehicle sales, parts sales, the two
> presidents agreed conceptually to
> pursue an arrangement whereby GM
> would build a Toyota-badged, right-
> hand drive vehicle in the US for
> sale by Toyota in Japan. A working
> group has been formed to finalize
> model specifications, exact timing
> and other details.

(a) Text with
quotations

(b) Text with
quotations striped

Table 4.2: Example of one of the texts before (a) and after (b) removing the quotations. Due to the evident loss of information, I have decided to preserve the quotations in the dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Method</td>
<td>72.40 %</td>
</tr>
<tr>
<td>kNN (k = 10)</td>
<td>75.93 %</td>
</tr>
<tr>
<td>Centroid (Normalized Sum)</td>
<td>78.85 %</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>81.03 %</td>
</tr>
<tr>
<td>SVM (Linear Kernel)</td>
<td>82.84 %</td>
</tr>
<tr>
<td>ULMFiT</td>
<td>87.92 %</td>
</tr>
</tbody>
</table>

Table 4.3: Results of other classification methods on 20 Newsgroups dataset [4], compared to ULMFiT.
### 4. Experimental setup

![Confusion matrix of the classification result produced by the model on the 20 Newsgroups dataset.](image)

The misclassification rate is in this case 11.8%, however, as the newsgroups are arranged into hierarchies, some of them have significant thematical overlap like the `comp.*` hierarchy or the `talk.politics.misc` newsgroup which is consistently misclassified as `talk.politics.guns`.

**Figure 4.1:** Confusion matrix of the classification result produced by the model on the 20 Newsgroups dataset. The misclassification rate is in this case 11.8%, however, as the newsgroups are arranged into hierarchies, some of them have significant thematical overlap like the `comp.*` hierarchy or the `talk.politics.misc` newsgroup which is consistently misclassified as `talk.politics.guns`. 
Chapter 5

Experiments

In my experiments, I focused on the flexibility and resilience of the ULMFiT model. Howard and Ruder [8] in their work used relatively large corpora usually with just a few classes. My goal was to test if the model can achieve as good results on harder datasets. For this, I designed the following four experiments. In the first two experiments, I have tested the impact of the size of the training dataset, on the performance of the model, by training it on gradually smaller subsets of the used dataset (described in Section 4.1). In the second experiment presented in Section 5.2 I trained it in semi-supervised mode, by fine-tuning (see Section 3.2.3) it on the full dataset, and only training the classifier on its subsets.

Finally, in the last two experiments (Sections 5.3 to 5.4), I have introduced an artificial imbalance to the dataset, by removing training data from a selected subset of the classes.

More detailed results of all experiments, can also be found in Appendix A.

5.1 Impact of the size of the training dataset on the classification accuracy

This experiment was conducted to test the robustness of the transfer learning in NLP. I have trained the pre-trained model on gradually smaller datasets and measured its accuracy. The smaller training data were obtained by randomly selecting subsets of fixed sizes, from each class. I have tested the model on $\frac{3}{4}$, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ of the original class sizes.

I have repeated this experiment ten times, each time new random subset of the original dataset was generated. The results reported in Table 5.1 are averages over the ten runs; the nonaggregated results can be found in Table A.1

The results presented in Figure 5.1 show that even with only half of the training data, the model, achieves an average accuracy of 86.26 %, and performs consistently better than the baseline SVN method on full dataset (accuracy of 82.84 %). Moreover even after removing $\frac{7}{8}$ of the training data, the model still achieves accuracies above 70 %, most of the times.
5. Experiments

Figure 5.1: Classification accuracies for supervised and semi-supervised ULMFiT on different relative amounts of training data

<table>
<thead>
<tr>
<th>Relative size</th>
<th>Number of texts</th>
<th>Average accuracy on the validation data</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>11293</td>
<td>87.92 %</td>
<td>0.302 %</td>
</tr>
<tr>
<td>0.75</td>
<td>8471</td>
<td>87.67 %</td>
<td>0.542 %</td>
</tr>
<tr>
<td>0.5</td>
<td>5644</td>
<td>86.26 %</td>
<td>1.143 %</td>
</tr>
<tr>
<td>0.25</td>
<td>2822</td>
<td>80.85 %</td>
<td>1.159 %</td>
</tr>
<tr>
<td>0.125</td>
<td>1411</td>
<td>69.93 %</td>
<td>2.266 %</td>
</tr>
</tbody>
</table>

Table 5.1: Average classification accuracies of the model trained on datasets with different numbers of training examples.

5.2 Impact of the size of the training dataset on the classification accuracy, with the language model fine-tuned on the whole dataset

Obtaining labeled datasets, usually means humans annotating it, which can be very costly (unless it is done by students). Hence, it would be useful, to also gain some knowledge from unlabeled data, which are usually much easier to acquire.

ULMFiT enables semi-supervised training by fine-tuning (see Section 3.2.3) it on a partially annotated corpus and using only its labeled subset to train the final classifier. I have conducted an experiment to determine, how effective the model is at gaining knowledge using this method.

The setup was identical to the previous experiment, except that for the fine-tuning step, the whole dataset, with its labels removed was used. As before, I have repeated this experiment 10 times. The Table 5.2 shows the
5.3 Performance of the ULMFiT model on an imbalanced dataset – with ten classes unbalanced

<table>
<thead>
<tr>
<th>Relative size</th>
<th>Number of texts</th>
<th>Average accuracy on the validation data</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>11293</td>
<td>87.92 %</td>
<td>0.302 %</td>
</tr>
<tr>
<td>0.75</td>
<td>8471</td>
<td>87.45 %</td>
<td>0.351 %</td>
</tr>
<tr>
<td>0.5</td>
<td>5644</td>
<td>86.91 %</td>
<td>0.517 %</td>
</tr>
<tr>
<td>0.25</td>
<td>2822</td>
<td>84.74 %</td>
<td>0.841 %</td>
</tr>
<tr>
<td>0.125</td>
<td>1411</td>
<td>81.52 %</td>
<td>0.819 %</td>
</tr>
</tbody>
</table>

Table 5.2: Average classification rates of the model fine-tuned on the full dataset, but with the classifier trained on its subset.

average achieved accuracies. The Figure 5.1 which compares results of the first two experiments, shows that the model performs noticeably better, when trained in this manner.

The results on the larger data sizes do not differ significantly from the previous experiment with an average accuracy of 86.91 % when trained on half of the dataset compared to 86.26 % achieved in fully supervised setup on the same data size. However, whereas before there was a sharp drop off after 75 % of the training data was removed, when the semi-supervised training was used, the classification rate decreases very gradually. Even on the smallest dataset, with only about 70 training examples per class, the accuracy was consistently above 80 %.

5.3 Performance of the ULMFiT model on an imbalanced dataset – with ten classes unbalanced

Class imbalance is often the main obstacle in utilizing machine learning models in real-life application. In extreme cases, the underrepresentation of some classes in the training data can lead to the model learning to overlook them, as ignoring them, may not result in a significant enough increase in the error rate.

I have introduced an artificial class imbalance, by selecting ten classes, from which I gradually removed training data, while leaving the other ten classes unaltered. In a similar manner to the previous experiments, I have trained the model on the dataset with 3/4, 1/2, 1/4, and 1/8 of the texts from selected classes left. I have run the experiment five times, on each size of the dataset.

The Figure 5.3 shows that although the classification rate on the unbalanced classes has lowered (by up to 25 % in some cases), the model does never learn to ignore them. The confusion matrix in Figure 5.2 further shows that most of the error on the unbalanced classes comes from the model misclassifying them into one of the related classes. As the 20 Newsgroups dataset, has a lot of thematically similar categories (e.g., alt.atheism, soc.religon.christian and talk.religion.misc), when some of them are unbalanced, the model does not have enough data to learn to recognise the subtle differences between them.
5. Experiments

Figure 5.2: Confusion matrix, obtained after training the model on a dataset where artificial imbalance have been introduced. From the classes highlighted in red, only $\frac{1}{8}$ of the original training data has been left. On those classes the misclassification rate is notably higher compared to the results of the model trained on full the dataset, shown in Figure 4.1. However, most of the error happens on thematically overlapping classes.

5.4 Performance of the ULMFiT model on an imbalanced dataset – with two classes unbalanced

The previous experiment showed that the model handles quite well even dataset, with half of the classes significantly smaller. However, the sum of the smaller classes still represented a significant portion of the training data. So to further test the potential problems described in the previous section, in this experiment I have lowered the number of unbalanced classes to two, as this should further encourage the model to learn to overlook them. Otherwise, the setup remained unchanged.

The results in Figures 5.4 to 5.5 show similar behaviour as before. Even when each of the unbalanced classes made up only about 0.72 % of the
5.4. Performance of the ULMFiT model on an imbalanced dataset – with two classes unbalanced

Figure 5.3: The change in the misclassification rate of the model on each class compared to the error rate of the model trained on the full dataset. The red marks the classes from which the data has been gradually removed (leaving $\frac{3}{4}$, $\frac{1}{2}$, and $\frac{1}{4}$ of the original data).

In the training dataset, the model was able to classify them correctly with 60.32 and 67.86 percent accuracy respectively. Both of the unbalanced classes were also unbalanced in the previous experiment (Section 5.3), and comparison of these two experiments shows that the error rates on those are very similar, which indicates that the number of unbalanced classes does not have a significant impact on the behaviour of the model.

Figure 5.4 shows that as in the previous experiment, most of the unbalanced classes tend to be assigned to some of the thematically related class; `comp.windows.x` often being confused with other newsgroups from the `comp.*` hierarchy and the `soc.religion.christian` newsgroup being mistaken for `alt.atheism` and `talk.religion.misc`. 

...
5. Experiments

Figure 5.4: Confusion matrix, obtained after training the model on a dataset where artificial imbalance have been introduced. From the classes highlighted in red, only \( \frac{1}{8} \) of the original training data has been left. On those classes the misclassification rate is notably higher compared to the results of the model trained on full the dataset, shown in Figure 4.1. However, most of the error happens on thematically overlapping classes.
Figure 5.5: The change in the misclassification rate of the model on each class compared to the error rate of the model trained on the full dataset. The red marks the classes from which the data has been gradually removed (leaving $\frac{3}{4}$, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ of the original data.)
Chapter 6

Conclusion

The ULMFiT promised a universal method for text classification, easily applicable on any dataset. In my experiments, I tried to simulate some conditions commonly found in real-life situations.

As large labeled datasets are in practice very rare, the ability of the model to learn the task from a small amount of training data is crucial. Since, the pretrained ULMFiT already possess some domain knowledge, it was able to achieve 80.85 % validation accuracy, even when trained with less than 150 text examples per class, only a quarter of the original dataset, compared to the 87.92 % achieved on the full dataset. Moreover, if the model is allowed to use unsupervised fine-tuning on the rest of the dataset, its accuracy improves by 3.89 %.

Furthermore, when trained on just 70 texts per class (1411 texts in total), the difference in measured accuracies compared to the previous setup grew to 11.59 %. The results confirm conclusions presented by Howard and Ruder [8] as they show, that the model can successfully gain knowledge from the unsupervised Language Model fine-tuning on the target task dataset. This, in combination with the supervised classifier training, makes it a viable semi-supervised method for text classification on datasets with a small number of labeled examples, but with access to a larger amount of unlabeled data.

As unbalanced training datasets pose a significant challenge in all classification tasks, it was essential to evaluate the ULMFiT model in such situations. In the experiments described in Sections 5.3 to 5.4 I’ve introduced an artificial class imbalance to the training dataset, by deleting data from selected sets of classes. As expected, the error on these has increased, but they were misclassified mostly to the thematically related classes.

Overall the model proved its dexterity and robustness over all tested tasks. The good qualities of the model are further supported by the reports of its deployment in a real-world production environment [19].

6.1 Future Work

Using pre-trained Language Models for text classification has great prospects. Possible ways of improving this method might involve usage of more complex models trained on even larger datasets, like the one presented by OpenAI [15].
However, as Howard and Ruder [8] showed, the main challenge lies in the careful transfer of the model to the target task. But with the use of newly gained knowledge, it might be viable, and it is definitely an interesting direction for possible study.

Another possibility is expanding these methods to other languages. As the ULMFiT model was trained on texts from Wikipedia, which is available basically in every language, obtaining the training dataset, would be easy. Nonetheless, the relative sizes of non-English Wikipedias compared to the English version can vary greatly. For example, the number of articles on Czech Wikipedia is only about one-thirteenth of the ones on its English variant [26], while being generally much shorter. But as Czech is the 14th most common language on the internet, according to the W3Techs [24], web scraping might be a possible approach of obtaining extensive Language Modeling dataset.

However, one of the possible challenges of using ULMFiT model on other languages might be the much more complicated grammar and sentence structure.
Appendix A

Detailed results of the experiments

The following tables contain more complete results of the experiments described in Chapter 5.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0.75</th>
<th>0.5</th>
<th>0.25</th>
<th>0.125</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>87.160835</td>
<td>87.914632</td>
<td>85.092499</td>
<td>80.308195</td>
<td>69.227414</td>
</tr>
<tr>
<td>1</td>
<td>88.223817</td>
<td>87.523094</td>
<td>86.423896</td>
<td>80.806657</td>
<td>71.161864</td>
</tr>
<tr>
<td>2</td>
<td>87.807642</td>
<td>86.651873</td>
<td>86.996770</td>
<td>83.227470</td>
<td>71.298274</td>
</tr>
<tr>
<td>3</td>
<td>88.207577</td>
<td>88.134526</td>
<td>86.752232</td>
<td>79.188221</td>
<td>72.445672</td>
</tr>
<tr>
<td>4</td>
<td>87.947029</td>
<td>88.178198</td>
<td>86.602650</td>
<td>80.259772</td>
<td>70.791102</td>
</tr>
<tr>
<td>5</td>
<td>87.938089</td>
<td>87.058472</td>
<td>83.267818</td>
<td>82.513815</td>
<td>68.779007</td>
</tr>
<tr>
<td>6</td>
<td>88.187609</td>
<td>87.406726</td>
<td>87.131258</td>
<td>80.458106</td>
<td>73.018356</td>
</tr>
<tr>
<td>7</td>
<td>88.056997</td>
<td>88.213705</td>
<td>87.196342</td>
<td>79.690593</td>
<td>66.413281</td>
</tr>
<tr>
<td>8</td>
<td>87.692025</td>
<td>88.361035</td>
<td>86.544611</td>
<td>80.874767</td>
<td>70.388140</td>
</tr>
<tr>
<td>9</td>
<td>88.000813</td>
<td>87.304882</td>
<td>86.577849</td>
<td>81.171910</td>
<td>65.811565</td>
</tr>
<tr>
<td>avg</td>
<td>87.922243</td>
<td>87.674714</td>
<td>86.258593</td>
<td>80.849951</td>
<td>69.933468</td>
</tr>
<tr>
<td>std</td>
<td>0.302300</td>
<td>0.542363</td>
<td>1.142954</td>
<td>1.158793</td>
<td>2.266158</td>
</tr>
</tbody>
</table>

Table A.1: Accuracies measured in the experiment described in Section 5.1.
A. Detailed results of the experiments

<table>
<thead>
<tr>
<th>Relative amounts of training data</th>
<th>1</th>
<th>0.75</th>
<th>0.5</th>
<th>0.25</th>
<th>0.125</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>87.160835</td>
<td>87.383913</td>
<td>85.794978</td>
<td>83.668693</td>
<td>81.707504</td>
</tr>
<tr>
<td>1</td>
<td>88.223817</td>
<td>87.916595</td>
<td>86.762542</td>
<td>85.688351</td>
<td>80.610170</td>
</tr>
<tr>
<td>2</td>
<td>87.807642</td>
<td>87.178675</td>
<td>86.570723</td>
<td>84.768064</td>
<td>80.977576</td>
</tr>
<tr>
<td>3</td>
<td>88.207577</td>
<td>87.462099</td>
<td>87.182601</td>
<td>83.416468</td>
<td>82.076774</td>
</tr>
<tr>
<td>4</td>
<td>87.947029</td>
<td>87.265326</td>
<td>86.869639</td>
<td>85.731833</td>
<td>80.338175</td>
</tr>
<tr>
<td>5</td>
<td>87.938089</td>
<td>87.383555</td>
<td>87.151738</td>
<td>84.760113</td>
<td>81.144016</td>
</tr>
<tr>
<td>6</td>
<td>88.187609</td>
<td>87.307489</td>
<td>86.673243</td>
<td>83.840033</td>
<td>82.111950</td>
</tr>
<tr>
<td>7</td>
<td>88.056997</td>
<td>88.248552</td>
<td>87.508883</td>
<td>84.917638</td>
<td>80.923438</td>
</tr>
<tr>
<td>8</td>
<td>87.692025</td>
<td>87.362567</td>
<td>86.841745</td>
<td>85.943018</td>
<td>83.100916</td>
</tr>
<tr>
<td>9</td>
<td>88.000813</td>
<td>86.971771</td>
<td>87.763805</td>
<td>84.733168</td>
<td>82.261575</td>
</tr>
<tr>
<td>avg</td>
<td>87.922243</td>
<td>87.448095</td>
<td>86.913333</td>
<td>84.746738</td>
<td>81.521668</td>
</tr>
<tr>
<td>std</td>
<td>0.302300</td>
<td>0.350518</td>
<td>0.516665</td>
<td>0.840956</td>
<td>0.819113</td>
</tr>
</tbody>
</table>

Table A.2: Accuracies measured in the experiment described in Section 5.2

<table>
<thead>
<tr>
<th>Class</th>
<th>Relative amounts of training data</th>
<th>1</th>
<th>0.75</th>
<th>0.5</th>
<th>0.25</th>
<th>0.125</th>
</tr>
</thead>
<tbody>
<tr>
<td>alt.atheism</td>
<td>19.56</td>
<td>21.57</td>
<td>24.33</td>
<td>37.05</td>
<td>46.58</td>
<td></td>
</tr>
<tr>
<td>comp.graphics</td>
<td>17.74</td>
<td>17.28</td>
<td>17.22</td>
<td>19.28</td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>23.05</td>
<td>22.89</td>
<td>22.39</td>
<td>21.78</td>
<td>20.00</td>
<td></td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>21.17</td>
<td>18.57</td>
<td>20.66</td>
<td>14.80</td>
<td>17.96</td>
<td></td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>13.45</td>
<td>16.52</td>
<td>16.52</td>
<td>26.130</td>
<td>30.75</td>
<td></td>
</tr>
<tr>
<td>comp.windows.x</td>
<td>17.81</td>
<td>17.40</td>
<td>20.51</td>
<td>26.28</td>
<td>34.59</td>
<td></td>
</tr>
<tr>
<td>misc.forsale</td>
<td>9.28</td>
<td>10.92</td>
<td>12.92</td>
<td>23.74</td>
<td>29.74</td>
<td></td>
</tr>
<tr>
<td>rec.autos</td>
<td>10.68</td>
<td>9.06</td>
<td>11.59</td>
<td>8.41</td>
<td>11.19</td>
<td></td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>4.72</td>
<td>6.98</td>
<td>6.83</td>
<td>10.20</td>
<td>15.27</td>
<td></td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>4.84</td>
<td>4.53</td>
<td>4.84</td>
<td>4.53</td>
<td>4.94</td>
<td></td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>1.80</td>
<td>2.01</td>
<td>1.80</td>
<td>1.95</td>
<td>2.41</td>
<td></td>
</tr>
<tr>
<td>sci.crypt</td>
<td>7.12</td>
<td>6.01</td>
<td>6.26</td>
<td>5.51</td>
<td>6.41</td>
<td></td>
</tr>
<tr>
<td>sci.electronics</td>
<td>19.03</td>
<td>18.02</td>
<td>22.39</td>
<td>31.55</td>
<td>37.86</td>
<td></td>
</tr>
<tr>
<td>sci.med</td>
<td>8.74</td>
<td>10.05</td>
<td>11.21</td>
<td>17.68</td>
<td>22.58</td>
<td></td>
</tr>
<tr>
<td>sci.space</td>
<td>7.21</td>
<td>6.60</td>
<td>7.06</td>
<td>12.64</td>
<td>17.46</td>
<td></td>
</tr>
<tr>
<td>soc.religion.christian</td>
<td>7.19</td>
<td>7.69</td>
<td>8.54</td>
<td>14.82</td>
<td>19.35</td>
<td></td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>10.88</td>
<td>10.33</td>
<td>13.46</td>
<td>16.32</td>
<td>18.74</td>
<td></td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>11.91</td>
<td>13.94</td>
<td>11.65</td>
<td>12.02</td>
<td>11.06</td>
<td></td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>38.77</td>
<td>37.55</td>
<td>37.10</td>
<td>36.57</td>
<td>37.75</td>
<td></td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>27.57</td>
<td>23.98</td>
<td>28.05</td>
<td>21.99</td>
<td>23.51</td>
<td></td>
</tr>
</tbody>
</table>

Table A.3: Average error per class with 10 classes unbalanced (in bold) as described in Section 5.3
### Table A.4: Average error per class with 2 classes unbalanced (in **bold**) as described in Section 5.4

<table>
<thead>
<tr>
<th>Class</th>
<th>Relative amounts of training data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>alt.atheism</td>
<td>19.56</td>
</tr>
<tr>
<td>comp.graphics</td>
<td>17.74</td>
</tr>
<tr>
<td>comp.os.ms-windows.misc</td>
<td>23.05</td>
</tr>
<tr>
<td>comp.sys.ibm.pc.hardware</td>
<td>21.17</td>
</tr>
<tr>
<td>comp.sys.mac.hardware</td>
<td>13.45</td>
</tr>
<tr>
<td><strong>comp.windows.x</strong></td>
<td></td>
</tr>
<tr>
<td>misc.forsale</td>
<td>9.28</td>
</tr>
<tr>
<td>rec.autos</td>
<td>10.68</td>
</tr>
<tr>
<td>rec.motorcycles</td>
<td>4.72</td>
</tr>
<tr>
<td>rec.sport.baseball</td>
<td>4.84</td>
</tr>
<tr>
<td>rec.sport.hockey</td>
<td>1.80</td>
</tr>
<tr>
<td>sci.crypt</td>
<td>7.12</td>
</tr>
<tr>
<td>sci.electronics</td>
<td>19.03</td>
</tr>
<tr>
<td>sci.med</td>
<td>8.74</td>
</tr>
<tr>
<td>sci.space</td>
<td>7.21</td>
</tr>
<tr>
<td><strong>soc.religion.christian</strong></td>
<td></td>
</tr>
<tr>
<td>talk.politics.guns</td>
<td>10.88</td>
</tr>
<tr>
<td>talk.politics.mideast</td>
<td>11.91</td>
</tr>
<tr>
<td>talk.politics.misc</td>
<td>38.77</td>
</tr>
<tr>
<td>talk.religion.misc</td>
<td>27.57</td>
</tr>
</tbody>
</table>
Appendix B

Contents of the CD

/  
  |__ src  
  |     |__ model_structure  
  |     |__ scripts  
  |     |__ fastai_scripts  
  |     |__ fresh_models  
  |     |__ notebooks........Jupyter notebooks used for data evaluation  
  |     |__ experiments........Files generated during the experiments  
  |     |     |__ exp1  
  |     |     |__ exp2  
  |     |     |__ exp3  
  |     |     |__ exp4  
  |     |__ thesis.........................LaTeX codes of this thesis  
  |     |__ figures  
  |     |__ chapters
Appendix C

Bibliography


