DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Transfer and Multitask Learning for Aspect-Based Sentiment Analysis using the Google Transformer Architecture

Felix Schober

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Transfer and Multitask Learning for Aspect-Based Sentiment Analysis using the Google Transformer Architecture

Transfer- und Multitask-Lernen für aspektbasierte Sentimentanalyse mit der Transformer-Architektur von Google

Author:	Felix Schober
Supervisor:	PD Dr. Georg Groh
Advisor:	Gerhard Hagerer M.Sc.
Submission Date:	15.05.2019

I confirm that this master's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15.05.2019 Felix Schober

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Motivation

1.2 Outline

2 Related Work

2.1 Sentiment Analysis

2.2 Aspect Based Sentiment Analysis

3 Theoretical Background

This chapter attends to the theoretical background for the technologies used in this thesis.

3.1 Convolutional Neural Networks

3.2 Word Representations

3.2.1 Glove

3.2.2 FastText

3.2.3 Elmo

3.3 Google Transformer Architecture

3.3.1 Positional Encoding

3.3.2 Attention Mechanism

3.3.3 Pointwise Layer

3.3.4 Xavier Initialization

3.3.5 Adam

3.3.6 NOAM

3.4 Multi-Task Learning

Rich Caruana first introduced Multi-Task Learning (MTL) in 1993. Conventional machine learning approaches break a problem down in smaller tasks and solve one task at a time (e.g., word-by-word Part-of-Speech (POS)-tagging [21], word-by-word Named-entity recognition (NER) [20] or handwritten image classification [8]). In each

of these tasks a classification algorithm solves exactly one task (Assigning a ’part-of-speech’ or entity type to a word, or the classification of handwritten digits). Caruana shows that combining multiple related tasks improves model performance [4][3].
In Multi-Task Learning (MTL), multiple related tasks are learned in parallel and share a common representation. Generally speaking every machine learning model which optimizes multiple objectives for a single sample can be considered as Multitask Learning. This includes multi-label classification where one sample can have multiple labels as well as instances where different sample distributions or datasets are used for different tasks.
MTL is similar to how humans learn. Generally, humans learn new tasks by applying knowledge from previous experiences and activities. For instance, it is easier to learn ice skating when someone previously learned inline skating. This is because all the underlying important aspects of the tasks are very similar.
When tasks are related this also holds true for machine learning. When learning these tasks in parallel model performance is improved compared to learning them individually since the additional knowledge that a related task carries, can be used to improve on the original task [3].
There are four important aspects one can use to determine if MTL can bring performance boosts for a specific objective:

1. Multi Label Task: Multi Label classification task where one sample can have more than one label are almost always inherintly solved using MTL if labels are predicted by one model. Multiple authors show that adding tasks always improves performance compared to a separate model for each task as an alternative [17].

2. Shared low-level features: MTL only makes sense if the tasks share low level features. For instance, image classification and Natural Language Processing (NLP) do not share common features. In this case the model would not benefit from MTL because one task can not help to improve the other task. Therefore, it is important to choose tasks that are related to each other [24]. In most cases MTL will work with NLP tasks because they usually share at least some kind of sentence or word embedding as a common layer.

3. Task Data Amount: Several authors have suggested that it is important for the success of MTL training that the amount of data for the tasks is similar. Otherwise the model will mainly optimize for the task with most training samples.

4. Model Size: Finally, the multi-task model needs to have enough parameters to support all tasks [3].

3.4.1 Differentiation against Transfer Learning

Training samples from one task can help improve the other task and vice versa. This is important for the differentiation against transfer learning [15]. In MTL each task is equally important. In transfer learning the source task is only used to improve the target task so the target task is more important than the source task [24]. In addition, Transfer Learning uses a linear training timeline. First, the source task is learned and then after learning is completed this knowledge is applied to boost the learning process of the target task. MTL, in contrast, is learning both tasks jointly together instead of one after the other.

3.4.2 Improvements through Generalization

There are several reasons why the MTL paradigm performs so well. For instance, the generalization error is lower on shared tasks [4]. MTL acts as a regularization method and encourages the model to accept hypothesis that explain more than one task at the same time [19]. The model is forced to develop a representation that fits the data distributions for all tasks. In the end this creates a model that generalizes better because it must attend to different objectives.

3.4.3 Improvements through Data Augmentation

Secondly, Multi-Task Learning increases the number of available data points for training. All tasks share a common representation. While training one task all other tasks are also implicitly trained through the common representation.

Statistical Data Amplification

Each new task also introduces new noise. Traditionally, a model tries to learn by ignoring the noise from its data. However, if the model does not have enough training samples it will overfit because it focuses too much on the noise to explain the data. By introducing additional tasks, new data and therefore new noise is introduced which the model has to try and ignore [19]. This aspect is called *Statistical Data Amplification*[**Caruana1995a**].

Blocking Data Amplification

Blocking Data Amplification occurs when there is little or no noise. Consider the simple example from Caruana [**Caruana1995a**] that there are two tasks T and T' and common features F. The first task T is $T = A \land F$ and the second task T' is $T' = \neg A \land F$. For

$A = 0$ only T uses feature F and T' does not and for $A = 1$ it's the other way around. By training on both tasks F is used no matter what value A takes.
Rei makes use of these aspects and proposed a sequence labeling framework which uses a secondary, unsupervised word prediction task to augment other tasks such as NER or chunking. They show that by including the word prediction the auxiliary task performance is improved for all sequence labeling benchmarks they tried [18].
Similarly, Plank et al. show that learning to predict word-frequencies along with POS-tagging also improves the total model performance [14]. They argue that predicting word frequencies helps to learn the differentiation between rare and common words.

3.4.4 Architecture

The most common architecture for multitask learning is shown in figure 3.1. It is called hard parameter sharing and consists of at least one layer which is shared among all tasks. In addition, each task has at least one separate layer. This approach is also the one we used for our model which is described in chapter 4.
The easiest way to compute the loss for a hard parameter sharing MTL architecture is to take the sum of all losses for the individual tasks which is shown in equation ...

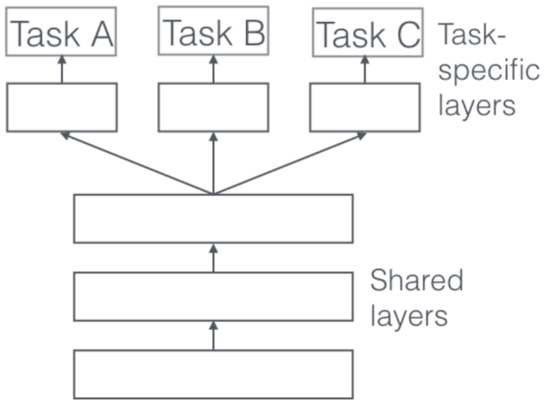

Figure 3.1: Hard parameter sharing. The first three layers are shared among tasks A, B and C. Each task also has one or more layers. Source: Ruder 2017 [19]

3.5 Transfer Learning

In 1991, Pratt et al. suggested to transfer information encoded in a neural network by reusing the network weights in a new network [16]. They show that even accounting

for the training time of the source network they achieved significant speedups when training a target network compared to random weight initialization.
Yosinski et al. provide a more modern definition: First, a base network is trained on a base dataset. Then, the learned features (the knowledge) of the base network is transfered to a second target network which is then trained on the target dataset and task [23]. This process works well if the base and target dataset and tasks are similar.
Goodfellow et al. give a more general definition. They define transfer learning as the transfer of previously learned knowledge from one or multiple sources to a target domain with fewer examples [5].
Figure 3.2 communicates those definitions.

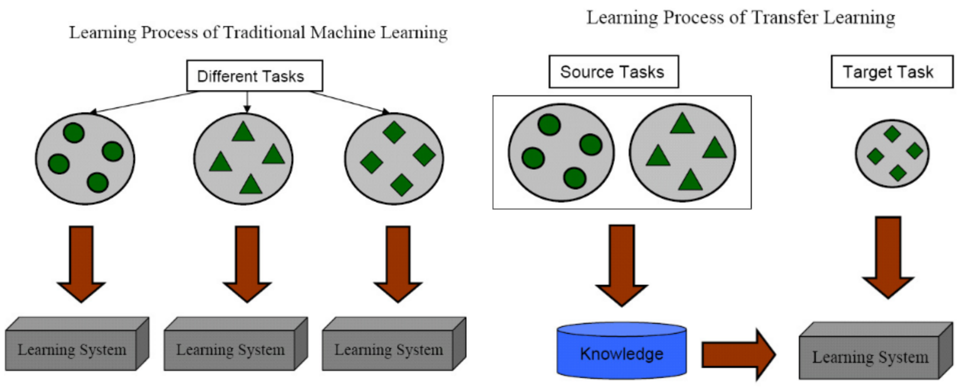

Figure 3.2: Difference in traditional machine learning were each model uses its own dataset and task. In contrast, in transfer learning a model is first trained on source tasks and part of the features are transformed to the target model to facilitate training. Source: Pan and Yang [13]

In practice it is very expensive to collect or recollect training data for every new domain. Transfer learning makes it possible to transfer knowledge from a larger dataset to a smaller dataset which greatly reduces the labeling effort [1]. When the target dataset has significantly fewer examples than the base dataset studies showed that it is possible to train large networks without overfitting [**Donahue2013**][**Zeiler2014**]. Usually, after the base model has been trained on the large dataset, the first n layers of the base model are copied over as the first n layers of the target model. The remaining layers of the target model are then randomly initialized and trained. The weights of the n layers from the base model can either be *frozen* or *finetuned* along the rest of the target model. If the target dataset has few samples compared to the number of parameters in the first n layers, finetuning can actually result in overfitting which is a reason why the error during target training is often not backpropagated to the first n layers [23].

Pre Training

The most common way to employ transfer learning is pre-training. Pre-training is often used in image recognition were interestingly, the first few layers generally form into the same feature regardless of the domain or task [23]. Consequently, researchers are able to exploit this by taking the first layers from a model which was previously trained on a large dataset like ImageNet [**Russakovsky2015**] and use these weights for their tasks which might have less examples.
This paradigm can also be applied on natural language processing. Understanding what words mean is the fundamental problem every NLP model has to solve. Therefore, it is sensible to use an embedding layer which has been pre-trained on large datasets like common crawl which contain petabytes of information [**commonCrawl**]. This pre-trained embedding layer can then be used in a model trained on a much smaller dataset.

3.6 Hyperparameter Optimization

3.6.1 Grid Search

3.6.2 Random Search

3.6.3 HyperOpt

3.7 Methodology

3.7.1 Performance Measurements

Precession - Recall

The most used measure for the precision of food classifiers is the average accuracy which is calculated by dividing the number of correct matches and the total number of samples. Accuracy, however, gives no information about the underlying conditions. It is a measure of overall performance. To have a higher chance of suggesting the correct items, future systems may present a list of options that the user can chose from. Intuitively, the accuracy is much higher if a classifier can present a list of items with high confidences instead of only one item because the problem is much easier. Accuracy, however, does not measures how easy a problem is. If a classifier were able to suggest all classes as options the accuracy would always be 100% although the results are not useful at all.
The combination of precision and recall objectively measures the actual relevance and performance of a classifier for a class of images because it includes the amount of

considered items and the correct predictions. In this case the amount of considered items changes based on how many items the classifier can suggest. Precision and recall is defined as:

$$Precision = \frac{T_p}{T_p + F_p} \quad Recall = \frac{T_p}{T_p + F_n}. \tag{3.1}$$

- True positives T_P is the number of correctly classified images of a class.
- False positives F_P are all images that the classifier predicted to be positive but are in reality negative. (Type I Error)
- False negatives F_N are all images that are positive (belong to the class) but are labeled as negative (do not belong to class) (Type II Error)

A high recall means that many images were matched correctly and a high precision denotes a low number of incorrectly classified images. The bigger the area under the Precision-Recall curve the better the classifier.

Null Error Rate

The null error rate is a baseline for any classification task that calculates the accuracy if a classifier would just predict the class with the most images.

Confusion Matrix

Confusion matrices are one of the most important metrics to understand why a classifier struggles with certain classes while getting a high precision with others. As the name suggests, a confusion matrix tells if the classifier "confuses" two classes.
A confusion matrix for n classes is always a $n \times n$ matrix where columns represent the actual images classes and rows represent the predicted image classes so if the diagonal of the matrix has high values this means that the classifier makes correct predictions.

Categorical Cross-Entropy

The categorical cross-entropy L_i is an error function that is used for the training of neural networks in classification tasks as the objective function. It is more versatile than the accuracy or the Mean Squared Error (MSE) because it takes the deviations of the predicted label $p_{i,j}$ and the actual label $t_{i,j}$ into account and weights the "closeness" of the prediction with the logarithm. For classification, cross entropy is more useful than

MSE because MSE gives too much emphasis on incorrect predictions. The categorical cross entropy function is defined as:

$$L_i = -\sum_j t_{i,j} \log(p_{i,j}) \tag{3.2}$$

The loss values that are used for the discussion of results for neural networks are the average values of the categorical cross-entropy (Average Cross-Entropy Error (ACE)).

3.7.2 Cross Validation

Cross validation is one of the most essential techniques to evaluate real-world classification performance. Classifiers like Support Vector Machines (SVMs) or neural networks are always better on data they have already seen. This is called overfitting (see section **??**). By training and testing on the same data the classification performance would be much better than the actual real world performance. To test if a classifier can actually work with samples it has not seen cross validation divides the dataset into different partitions.
For most tasks it is sufficient to divide the dataset into a training and a test set. The data in the training set is used to train the classifier and the test data is used to evaluate it with data is has not seen before.

k-fold Cross Validation

To make the classification evaluation even more robust, k-fold cross validation is used. By applying k-fold cross validation the dataset is randomly partitioned into k different parts. $k-2$ parts are used for training and two parts are used for the evaluation. This process is repeated k-times and after each iteration the parts are exchanged so that at the end, each sample was used for training and for validation. Calculating the mean of the k evaluations gives a much more robust measurement because the evaluation does not depend on the difficulty of the test partitions.

3.7.3 Early Stopping

4 Method

4.1 Architecture

4.1.1 Transformer

4.1.2 Aspect Heads

Linear Mean-Head

why mean? -> bring loss to similar value regardless of a) word length and b) aspect head choice (linear vs cnn)

Projection Mean-Head

CNN-Head

4.2 Multi-Task Learning

4.3 Transfer Learning

comparison to image first layer features which are very similar regardless of target domain. [23] -> Embedding layer, Transformer
However, last layer usually very dependet on domain and dataset -> good for model because last layer -> heads only domain relevant. So keep Embedding and transformer and exchange heads.

5 Experimental Setup

The following chapter describes the experimental setup for the discussion of results in chapter 6. The first section of the chapter deals with data preprocessing. Section 5.2 lists all datasets used for evaluations of the models and finally section 5.3 provides detail about the training and evaluation process used to generate the results.

5.1 Data Preprocessing

The following section describes the general data preprocessing steps which were taken for all datasets described in section **??**. Some of the preprocessing steps are specific to certain datasets and will be described there. All data preprocessing steps can be enabled or disabled to evaluate the impact on the performance of these preprocessing steps. Some of those results will be discussed in section 6.1.2 in chapter 6.

5.1.1 Text Cleaning

The main goal of the text cleaning step is

1. Reduce the number of words which are out of vocabulary
2. Keep the vocabulary size as small as possible.

without changing the semantics of the text.
The first step of the data preprocessing pipeline is the removal of all unknown characters which are not UTF-8 compatible. Those characters can occur because of encoding issues or words outside of the target language.

Contraction Expansion

Before we remove any special characters all contractions are expanded with the goal of reducing the vocabulary size and language normalization. Contractions are shortened versions of several words or syllables. In the English language, vowels are often replaced by an apostrophe. Especially in social media and spoken language a lot of contractions are used. *'I'll've'* and *'I will have'* have the same meaning but if they are

not expanded they produce a completely different embedding. *'I'll've'* will produce a (300)-dimensional vector (for glove and fasttext) whereas *'I will have'* will be interpreted as 3 300-dimensional vectors.
The contraction expansion is followed by the replacement of Uniform Resource Locators (URLs) with the token '<URL>' and e-mail addresses with the token '<MAIL>'. E-Mails and URLs are always out-of vocabulary and contain very little information that is worth encoding.
In addition any special characters are completely removed. Dashes ('-') are kept because there are compound-words which rely on dashes (e.g. non-organic).

Spell Checking

When writing comments in social media people tend to make spelling mistakes. Unfortunately, each spelling mistake is an out-of vocabulary word which we want to reduce as much as possible.
Therefore, a spell checker is used to prevent these mistakes. The first spell checker[1] which was evaluated relies on the Levenshtein Distance [9] and a dictionary to determine if a word is spelled incorrectly and to make suggestions which word was meant originally. Although, word replacement suggestions are good, the spell checking is slow especially with large dictionaries.
The second spell checker is called Hunspell developed by László Németh[2]. Hunspell is used in a variety of open- and closed sourced projects such as OpenOffice, Google Chrome or macOS. Hunspell also utilizes the Levenshtein Distance in addition to several other measurements. Both spell checkers suffer from false positives (word is incorrectly flagged as negative) as well as incorrect suggestions. Below are examples of Hunspells suggestions for words it did not recognize:

- taste/flavor -> flavorless
- GMOs -> G Mos
- Coca Cola -> Chocolate
- didn -> did

All of the above replacements are very bad because they change the meaning of the entire sentence.
Nevertheless, in terms of vocabulary size reduction they are clearly outperforming other techniques as table 5.2 demonstrates. Running Hunspell on the Amazon dataset

[1]PySpellchecker: https://pyspellchecker.readthedocs.io/en/latest/
[2]Hunspell: http://hunspell.github.io/

reduces the original vocabulary size of 1.6 Million by over 80% to about 311,000 unique words. In addition, as column *SP + TR-1* shows there are no tokens which only appear once. The reason for this is, that Hunspell always suggests something. Even words like *^_^b4* are replaced by new words even if it would make more sense to delete those words altogether.

Stemming and Lemmatization

Stemming were also briefly explored, however, they did not provide a significant performance improvement.

5.1.2 Comment Clipping

The transformer works with different input sequence lengths within one batch. Therefore, it is possible to group similar sequence lengths together and have arbitrary sequence lengths. Unfortunately, in each dataset there is a small percentage of sequences which are longer than other sequences. Due to the limited computational resources a batch of those long sequences does not fit into Graphics Processing Unit (GPU) memory. Therefore, all sentences are either padded or clipped to a fixed length. This is also a requirement for the CNN-based transformer aspect head since CNN-layers need a fixed number of input channels.

5.1.3 Sentence Combination

Some datasets feature sentence annotations instead of comment annotations. In this case important information for the aspect and sentiment classification could be encoded in previous sentences. Refer to figure XX for an example.
Therefore, n previous sentences are prepended to the current sentence where n is a hyper parameter which can be optimized. Similar to the clipping of comment wise annotations described in the previous section, these sentence combinations are also clipped and padded.
The process starts by repeatedly adding sentences to a stack. All $n-1$ sentences which are too long are cut at the front. The n-th sentence is cut in the back instead. This is done so that in the case of $n = 2$
See section **??** for the evaluation of this preprocessing step.

5.2 Data

5.2.1 Conll-2003 - Named Entity Recognition

5.2.2 GermEval-2017 - Deutsche Bahn Tweets

Bahn Name Harmonization

5.2.3 Organic-2019 - Organic Comments

5.2.4 Amazon Reviews Dataset

The Amazon Reviews Dataset consists of over 130 million Amazon product reviews from 1995 until 2015. Therefore, this dataset is one of the richest data sources for sentiment analysis or other related NLP tasks. The raw data is available directly through amazon.[3] The reviews are grouped into 45 product categories such as "Grocery", "Luggage" or "Video Games".
In 2013 McAuley and Leskovec compiled a subset of Amazon reviews [11]. This dataset contains 34,7 million reviews ranging from 1995 till 2013 grouped into 33 categories[4]. The authors also created a "Fine Food" Dataset from Amazon reviews [10] [5]. This dataset consists of 568454 Amazon reviews from 1995 till 2012. The domain of this specific dataset is related to the organic domain with 273 occurrences of the word 'organic'. Unfortunately, it does not contain predefined aspects so Aspect Based Sentiment Analysis (ABSA) is not possible without extensive pre-processing to generate aspects out of the reviews.
The datasets created in 2013 contains duplicates so McAuley et. al. generated an improved Amazon Reviews dataset in 2015 without duplicates [12][6]. This iteration of the dataset contains 142.8 million reviews from 1996 till 2014[6]. Due to the size of this dataset the authors provide a smaller dataset which only contains reviews from users who wrote exactly 5 reviews. This 5-core subset features 18 million reviews. The distribution of the domain categories is visualized in figure 5.1. As one can observe the dataset is substantially skewed towards the largest domain 'books' which makes up of 49% of the data.
To combat data imbalance and the sheer size of the dataset we propose a balanced subset of the 5-core dataset with 60000 reviews for each domain aside from *Musical Instruments*, *Amazon Instant Video*, *Automotive* and *Patio, Lawn and Garden*. These categories contain less than 50000 reviews so including them would skew the dataset again. In addition,

[3]https://s3.amazonaws.com/amazon-reviews-pds/readme.html
[4]Available through Stanford https://snap.stanford.edu/data/web-Amazon.html
[5]Available through Kaggle https://www.kaggle.com/snap/amazon-fine-food-reviews
[6]Available here: http://jmcauley.ucsd.edu/data/amazon/

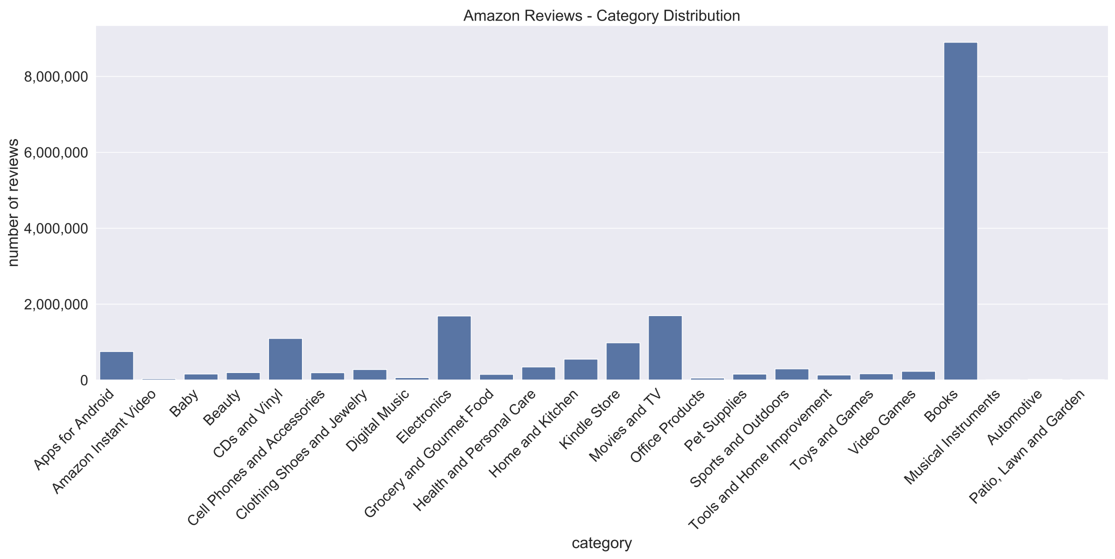

Figure 5.1: Number of reviews per domain category in the amazon review dataset by McAuley et. al. [12]

we also transformed the star-rating to the common negative-neutral-positive rating schema. Similar to Blitzer et. al. we interpret $1-2$ stars as negative, 3 stars as neutral and $4-5$ stars as positive sentiment [1].
To create a balanced dataset not only on domains but also on sentiment we sampled 20000 reviews for each sentiment for each domain. Overall, there are more positive reviews than neutral or negative reviews. Thus, some domains contain less than 20000 reviews per sentiment category. To prevent data imbalance, reviews from the remaining other sentiment categories are sampled so that each domain contains 60000 reviews in sum. This distribution and additional statistics about the dataset are documented in table 5.1.

Token Removal

There are over 145 million words in the dataset. These words combine into a vocabulary size of 1.6 million unique tokens and consequently into a very large embedding layer. (In comparison: the Organic2019 dataset has a vocabulary size of just 11,685.) Two techniques were used to reduce the vocabulary size:

1. Spell checking words
2. Removing rare tokens

	helpful	Pos.	Neu.	Neg.	stars	# words	
	mean	Count	Count	Count	mean	mean	std
Domain Category							
Apps for Android	0.22	20000	20000	20000	3.03	47	50
Baby	0.29	17012	17255	17012	3.33	105	106
Beauty	0.32	20000	20000	20000	3.10	90	94
Books	0.43	20000	20000	20000	3.08	176	201
CDs & Vinyl	0.44	20000	20000	20000	3.11	172	168
Cell Phones & Accessories	0.19	20000	20000	20000	3.06	93	138
Clothing Shoes & Jewelry	0.26	20000	20000	20000	3.11	67	70
Digital Music	0.53	47410	6789	5801	4.19	202	190
Electronics	0.43	20000	20000	20000	3.06	122	138
Grocery & Gourmet Food	0.33	28790	17514	13696	3.53	99	97
Health & Personal Care	0.35	20000	20000	20000	3.09	95	126
Home & Kitchen	0.44	20000	20000	20000	3.08	104	110
Kindle Store	0.35	20000	20000	20000	3.07	111	131
Movies & TV	0.39	20000	20000	20000	3.07	184	198
Office Products	0.29	45342	5060	2856	4.35	148	164
Pet Supplies	0.27	26412	15933	17655	3.35	91	96
Sports & Outdoors	0.30	20751	20000	19249	3.14	94	111
Tools & Home Impr.	0.40	39126	10769	10105	3.90	111	134
Toys & Games	0.32	11005	16357	11005	3.70	108	114
Video Games	0.41	20000	20000	20000	3.07	226	267
Total	0.35	506202	349677	337379	3.31	122	151

Table 5.1: Dataset statistics for the generated Amazon review subset for the domain categories. This table contains mean helpfulness rating; number of positive reviews; number of neutral reviews; number of negative reviews; mean star rating; mean number of words per review; standard deviation of the number of words per review

	Original	SP	SP + TR-1	TR-1	TR-2	TR-3
Word Count	148,129,490	-	0%	0.329%	0.389%	0.414%
Vocabulary Size	1,594,742	80.51%	80.51%	62.97%	74.41%	79.32%

Table 5.2: Different vocabulary size reduction techniques. This table shows the proportion of tokens that occur only 1, 2 or 3 times in relation to the total word count and the vocabulary size. *SP* is the spell checked dataset; *TR-n* is the token removal technique were *n* is the number times, tokens can occur in the dataset.

The process for the first technique is described in section 5.1.1. Another way to reduce the vocabulary size is by removing tokens, that only occur once or twice. These tokens make up the majority of the vocabulary size but only a small percentage of the overall word count. Table 5.2 shows the proportion of tokens which only occur 1, 2, or 3 times. As demonstrated in the table, infrequent tokens are very rare (all the tokens with one occurrence make up only 0.33% of the whole dataset). Yet, infrequent tokens make up over 74% of the total vocabulary size. Removing all tokens with one occurrence, therefore reduces the vocabulary size by 74% but only 0.33% of information is lost.
Most of these rare tokens are either incorrectly written (*nthis*), are part of structural elements such as headings (*review=======pros*) or are other unidentifiable characters and digits (*^_^b4*).

5.3 Training and Evaluation

5.3.1 Evaluation

The models that are used in this thesis are stochastic models since model parameters are randomly initialized. In addition, samples within the training batches are randomly shuffled. Therefore running the model multiple times leads to different results.
This means that it is necessary to collect model results multiple times. Unfortunately, k-fold cross validation is not possible for three out of the four datasets since the creators of the datasets provide a predefined split and changing the split during k-fold cross validation would prevent comparability with other results.
Therefore, for each dataset-result we repeat the experiment 5-times and report the mean and standard deviation. Iyer and Rhinehart suggest to run an experiment up to a 1000 times to get an optimal result [7]. However, this is not possible for our models due to computational constraints.
All experiments on hyper parameters are performed once with a fixed seed of 42. This

should make sure that all experiments on hyper parameters are reproducible. There are however some cudnn functions which are non-deterministic which means that even though a random seed is set the results could differ when running the same model with the same parameters multiple times.

GermEval 2017 - Evaluation

Wojatzki et al. [22] provide an evaluation script for their dataset GermEval-2017. All results from the GermEval 2017 challenge were evaluated using this dataset. Therefore, all results reported in this thesis also use the evaluation script to calculate the f1 score. This is done to be able to compare the results on this datasets to other approaches on this data.

Table 5.3: Example for GermEval-2017 evaluation. None sentiment is not shown. Document 1 is predicted correctly. Document 2 has a correct prediction for aspect A but an incorrect prediction for the sentiment of aspect B (in bold).

	Gold	**Prediction**
Document 1	A : negative	A : negative
Document 2	A : positive	A : positive
	B : **positive**	B : **negative**

Unfortunately, there are irregularities in the calculation of the micro f1 score. The evaluation script first creates every possible permutation of the combination of aspect and sentiment. If there are just two aspects (Aspect A and Aspect B) and four sentiments (n/a, negative, neutral, positive) this will generate 8 combinations (A-n/a, A-negative, ..., B-positive). This is used as the first input (*aspect_sentiment_combinations*) of the GermEval-2017 evaluation algorithm shown in 1.
In the next step, all gold-labels and predictions are paired together for each document based on the specific aspect-sentiment combination. The example in table **??** will produce the following combinations where the left side represents the gold labels and the right side the predictions. This would be the second input parameter *golds_predictions* for algorithm 1:

1. A:neg - A:neg (Document 1)
2. A:pos - A:pos (Document 2)
3. B:pos - B:n/a (Document 2)
4. B:n/a - B:neg (Document 2)

Using these inputs the algorithm will compute the following results:

- True Positives: 2
- False Positives: 2
- False Negatives: 2
- True Negatives: 26

which results in an f1-score of 0.5. In this example there is one misclassification where instead of predicting a pos. sentiment for aspect B the classifier predicted a neg. sentiment. When looking at the combination B:pos as the *'true class'* the model predicts a negative (NOT pos. sentiment) when in reality this is a positive (pos. sentiment) which is the definition of a *'False Negative'*. When looking at the combination B:neg as the *'true class'* the model predicts a positive (neg. sentiment) when in reality this is a negative (NOT neg. sentiment) which is the definition of a *' False Positive'*.
One could therefore argue that instead of producing two False Positives and two False Negatives the correct evaluation should be one False Positive and one False Negative.

5.3.2 Hardware

Training and evaluation of the models was done on four different machines. One of the servers belongs to the faculty of applied informatics, one is a local desktop machine and the last two are cloud instances. One is an Azure virtual compute instance with 8 Central Processing Unit (CPU) cores and 28 Giga Bytes (GB) of Random Access Memory (RAM) and the other is a Google Cloud GPU compute instance instance with an Intel Xeon E5-2670 processor, 15 GB of RAM and a NVIDIA Grid K520 GPU. See table 5.4 for more details.

5.3.3 Docker

Docker[7] is a framework for container virtualisation. Docker containers use the same kernel as the host system but an isolated file system with own system libraries.
Since training was performed on four different environments a Docker image was created which automates the installation of all required frameworks, environments, drivers and versions. An automated build pipeline builds a new image as soon as a new code version is pushed to the repository. Users can install or update an image directly from Docker Hub without rebuilding it every time locally.

[7]Docker: https://www.docker.com

Algorithm 1: GermEval-2017 Evaluation script.

```
Input  : aspect_sentiment_combinations: List of all possible combinations
         between aspects and sentiments including n/a, golds_predictions List of
         all comment wise pairs between gold labels and prediction labels
Output: ($tp$, $fp$, $tn$ $fn$)
tp = 0 fp = 0 tn = 0 fn = 0
foreach (aspect, sentiment) in aspect_sentiment_combinations do
    foreach (gold), (pred) in golds_predictions do
        if gold matches current aspect and sentiment then
            if gold matches prediction then
                tp++
            else
                fn++
            end
        else
            if prediction matches current aspect and sentiment then
                fp++
            else
                tn++
            end
        end
    end
end
return (tp, fp, tn, fn)
```

Table 5.4: Hardware used for model training

	OS	CPU	RAM	GPU
Schlichter 2	Ubuntu 12.04	Intel Core i7-3930K @ 3.20GHz	63 GB	NVIDIA Titan X
Schlichter 4	Ubuntu 14.04	Intel Xeon E5-2620 @ 2.00GHz	28 GB	-
Azure	Ubuntu 15.10	Intel Xeon E5-2673 v3 @ 2.40GHz	28 GB	-
Amazon AWS	Ubuntu 14.04	Intel Xeon E5-2670	15 GB	NVIDIA K520

The main concern of using docker for resource intensive task is the loss of performance due to the virtualization overhead. To evaluate this, epoch training time was measured with and without docker. The experiment was performed on machine-1 displayed in table 5.4. For both experiments a complete model was trained for 5 epochs on the Organic2019 dataset. Figure 5.2 visualizes the time each part of the training took. For both environments the mean execution time was around 195 seconds. This means that there is no difference between running a model inside a docker container or just locally. However, this is only the case when the host is running on a linux environment. On Windows and macOS, Docker has to virtualize part of the linux kernel. Therefore, there is no advantage of running on the exact same kernel as the host system. In addition, At the time of writing, the NVIDIA-runtime[8] is only supported for linux environments.

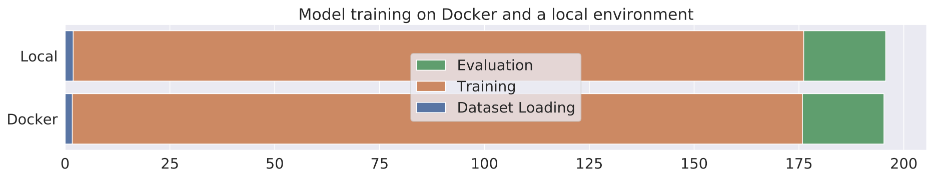

Figure 5.2: Docker vs. Local environment - Comparision of model training times.

[8]NVIDIA Docker Runtime: https://github.com/NVIDIA/nvidia-docker

6 Discussion of Results

6.1 Hyper Parameter Optimization

6.1.1 Model Parameters

Aspect Heads

see [17]

Pointwise Layer Size

Why smaller than model? ->
This dimensionality reduction is similar in moti- vation and implementation to the 1x1 convolutions in the GoogLeNet architecture (Szegedy et al., 2014). The wide lower layer allows for complex, expressive features to be learned while the narrow layer limits the parameters spe- cific to each task.
[17]

Transformer Architecture

Learning Rate Scheduler

Optimizer

Embedding

6.1.2 Data Preprocessing

Spell Checking

Stop Word Removal

Comment Clipping

??

6.2 Results for Named Entity Recognition

6.3 Results for Aspect-Based Sentiment Analysis

6.3.1 GermEval-2017

6.3.2 Organic-2019

6.3.3 Amazon Product Reviews

6.4 Impact of Multitask Learning

Difference to Multitask learning

6.5 Impact of Transfer Learning

7 Conclusion

7.1 Future Work

List of Figures

List of Tables

Bibliography

[1] J. Blitzer, M. Dredze, and F. Pereira. "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification." In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 440–447.

[2] R. Caruana. "Learning Many Related Tasks at the Same Time With Backpropagation." In: *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 657–664.

[3] R. Caruana. "Multitask Learning." Ph.D thesis. Carnegie Melon University, 1997.

[4] R. Caruana. "Multitask Learning: A Knowledge-Based Source of Inductive Bias." In: *PROCEEDINGS OF THE TENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING* (1993), pp. 41–48.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016, p. 785.

[6] R. He and J. McAuley. "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering." In: (2016). DOI: 10.1145/2872427.2883037. arXiv: 1602.01585.

[7] M. S. Iyer and R. R. Rhinehart. "A method to determine the required number of neural-network training repetitions." In: *IEEE Transactions on Neural Networks* 10.2 (1999), pp. 427–432. ISSN: 10459227. DOI: 10.1109/72.750573.

[8] Y. LeCun; B. Boser, J. S. Denker, R. E. Howard, W. Habbard, and L. D. Jackel. "Handwritten Digit Recognition with a Back-Propagation Network." In: *Advances in Neural Information Processing Systems* (1990), pp. 396–404. ISSN: 1524-4725. DOI: 10.1111/dsu.12130. arXiv: 1004.3732.

[9] V. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." In: *Insertions and Reversals. Sov* 6 (1966), pp. 707–710.

[10] J. McAuley and J. Leskovec. "From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews." In: *WWW '13 Proceedings of the 22Nd International Conference on World Wide Web*. Rio de Janeiro, Brazil: ACM, Mar. 2013, pp. 897–908. DOI: 10.1145/2488388.2488466. arXiv: 1303.4402.

[11] J. McAuley and J. Leskovec. "Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text." In: *RecSys '13 Proceedings of the 7th ACM conference on Recommender systems*. Hong Kong, China: ACM, 2013, pp. 165–172. ISBN: 9781450324090. DOI: `10.1145/2507157.2507163`.

[12] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. "Image-based Recommendations on Styles and Substitutes." In: (2015), pp. 1–11. arXiv: `1506.04757`.

[13] S. J. Pan and Q. Yang. "A survey on transfer learning." In: *IEEE Transactions on Knowledge and Data Engineering*. Vol. 22. 10. Piscataway, NJ, USA: IEEE Educational Activities Department, 2010, pp. 1345–1359. DOI: `10.1109/TKDE.2009.191`.

[14] B. Plank, A. Søgaard, and Y. Goldberg. *Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss*. Tech. rep. arXiv: `1604.05529v3`.

[15] L. Y. Pratt. "Discriminability-Based Transfer between Neural Networks." In: *Advances in neural information processing systems* (1993), pp. 204–211.

[16] L. Pratt, J. Mostow, and C. Kamm. "Direct Transfer of Learned Information Among Neural Networks." In: *AAAI-91 Proceedings*. AAAI, 1991, pp. 584–589.

[17] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande. "Massively Multitask Networks for Drug Discovery." Feb. 2015.

[18] M. Rei. "Semi-supervised Multitask Learning for Sequence Labeling." In: (Apr. 2017). arXiv: `1704.07156`.

[19] S. Ruder. "An Overview of Multi-Task Learning in Deep Neural Networks." In: (June 2017). arXiv: `1706.05098`.

[20] E. F. T. K. Sang and F. De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition." In: (2003). arXiv: `0306050 [cs]`.

[21] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. "Feature-rich part-of-speech tagging with a cyclic dependency network." In: 2007, pp. 173–180. DOI: `10.3115/1073445.1073478`.

[22] M. Wojatzki, E. Ruppert, S. Holschneider, T. Zesch, and C. Biemann. "GermEval 2017: Shared Task on Aspect-based Sentiment in Social Media Customer Feedback." In: *Proceedings of the GermEval 2017 – Shared Task on Aspect-based Sentiment in Social Media Customer Feedback*. 2017, pp. 1–12.

[23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems*. Ed. by Z. G. Weinberger, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Vol. 27. Curran Associates, Inc., 2014, pp. 3320–3328. arXiv: `1411.1792`.

[24] Y. Zhang and Q. Yang. "A Survey on Multi-Task Learning." In: (2017). arXiv: `1707.08114`.