DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Transfer- and Multitask Learning for aspect-based Sentiment Analysis using 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 Google Transformer Architecture

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

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 [13], word-by-word Named-entity recognition (NER) [12] or handwritten image classification [5]). 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 [9].

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 [14]. 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 [8]. 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 [14]. 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 [11]. 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. Furthermore, 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 [11].

Rei makes use of this aspect 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 [10].

Similarily, Plank et al. show that learning to predict word-frequencies along with POS-tagging also improves the total model performance [7]. 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 ...

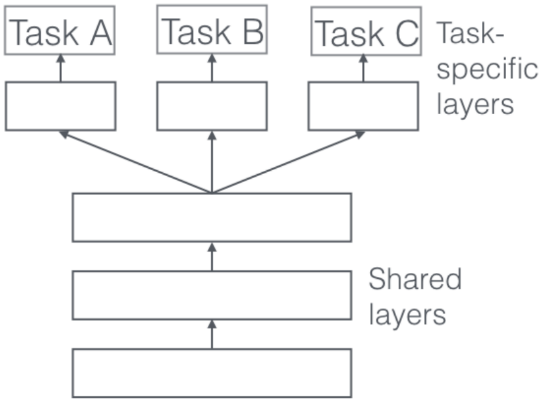

Figure 3.1: Hard parameter sharing. The first three layers are shared among tasks A, B and C. Each task also has one or more layers. Source: Ruder 2017 [11]

3.5 Transfer Learning

3.6 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

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 spell checker which is used for this step relies on the Levenshtein Distance [6] and a dictionary to determine if a word is spelled incorrectly and to make suggestions which word was meant originally.
slow
not good
edit distnace not good measure lot of false positives
Show table with results: out of vocabulary words before and after

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.

5.1.4 Stemming and Lemmatization

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

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 Multi-Domain Sentiment Dataset

[2]
The Multi-Domain Sentiment Dataset contains product reviews taken from Amazon.com from many product types (domains). Some domains (books and dvds) have hundreds of thousands of reviews. Others (musical instruments) have only a few hundred. Reviews contain star ratings (1 to 5 stars) that can be converted into binary labels if needed. This page contains some descriptions about the data. If you have questions, please email Mark Dredze or John Blitzer.
other applicatiosn: [1]

5.3 Training and Evaluation

5.3.1 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 5.1 for more details.

Table 5.1: 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

5.3.2 Docker

Docker is a virtualization framework ...
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.
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. As one can observe in figure XX

6 Discussion of Results

6.1 Hyper Parameter Optimization

6.1.1 Model Parameters

Aspect Heads

see [9]

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.
[9]

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

Acronyms

ACE Average Cross-Entropy Error.

API Application Programming Interface.

BoW Bag of Words.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

csv Comma Separated Values.

CUDA Compute Unified Device Architecture.

ETHZ Eidgenössische Technische Hochschule Zürich.

GB Giga Bytes.

GPS Global Positioning System.

GPU Graphics Processing Unit.

HTML Hypertext Markup Language.

IO Input / Output.

KNN K-nearest Neighbors.

MSE Mean Squared Error.

MTL Multi-Task Learning.

NER Named-entity recognition.

NLP Natural Language Processing.

POS Part-of-Speech.

RAM Random Access Memory.

std Standard Deviation.

SVM Support vector machine.

TUM Technische Universität München.

URL Uniform Resource Locator.

US United States.

List of Figures

List of Tables

Bibliography

[1] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. “Learning Bounds for Domain Adaptation.” In: *Advances in Neural Information Processing Systems 20 (NIPS 2007)*. Ed. by J. C. Platt and D. Koller and Y. Singer and S. T. Roweis. Curran Associates, Inc., 2008, pp. 129–136.

[2] 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.

[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] 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`.

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

[7] 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`.

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

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

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

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

[12] 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].

[13] 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.

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