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

3 Theoretical Background

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

3.1 Word Representations

3.1.1 Glove

3.1.2 FastText

3.1.3 Elmo

3.2 Google Transformer Architecture

3.2.1 Positional Encoding

3.2.2 Attention Mechanism

3.2.3 Pointwise Layer

3.3 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 [10], word-by-word Named-entity recognition (NER) [9] or handwritten image classification [3]). 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 [2][1].
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 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 the related task carries, can be used to improve on the original task [1].
Training samples from one task can help improve the other task and vice versa. This is important for the differentiation against transfer learning [5]. 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 [11].

3.3.1 Generalization

There are several reasons why the MTL paradigm performs so well. For instance, the generalization error is lower on shared tasks [2]. MTL acts as a regularization method and encourages the model to accept hypothesis that explain more than one task at the same time [8]. 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.

3.3.2 Data Augmentation

Secondly, Multi-Task Learning (MTL) 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 noise is learned instead this noise leads to overfitting. By introducing additional tasks, new data and therefore new noise is introduced which the model has to try and ignore [8].
Rei 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 auxiliary task performance is improved for all sequence labeling benchmarks [7].
Plank et al. show that the prediction of word frequencies along with POS-tagging can also improve the model performance [4].
While it has been shown multiple times that
MultiTask Learning NLP **[S\IeC {\o }gaard2016]**

Has been shown that learning more tasks instead of single tasks always improves performance see [6]
Good when there is insufficient data for the problem but multiple related tasks that each have a limited amount of data [11]
When does it make sense to use multi task learning:
1 shared lower level features (word embeddings + attention) 2 amount of data for each task is somewhat similar 3 neural network has to be big enough to do well on all the tasks. It can hurt performance if it is to small [1]
The most common architecture for

Data Augmentation

3.3.3 Architectures

Hard Parameter Sharing

Share hidden layers. Have several (one or more) layers per task

Soft Parameter Sharing

3.4 Transfer Learning

3.5 Methodology

3.5.1 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)).

3.5.2 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.

4 Method

4.1 Architecture

4.1.1 Pointwise Layers

5 Experimental Setup

5.1 Data

5.1.1 Data Preprocessing

Comment Clipping

Text Cleaning

Comment Clipping

5.1.2 Conll-2003 - Named Entity Recognition

5.1.3 SemEval-2016 - Restaurants and Laptops

5.1.4 GermEval-2017 - Deutsche Bahn Tweets

5.1.5 Organic-2019 - Organic Comments

6 Discussion of Results

6.1 Hyper Parameter Optimization

6.1.1 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.
[6]

6.1.2 Spell Checker

6.2 Results for Named Entity Recognition

6.3 Results for Aspect-Based Sentiment Analysis

6.3.1 GermEval-2017

6.3.2 GermEval-2017

6.3.3 Organic-2019

6.4 Impact of Multitask Learning

Difference to Multitask learning

6.4.1 Impact of Aspect heads on performance

see [6]

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.
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] R. Caruana. “Multitask Learning.” Ph.D thesis. Carnegie Melon University, 1997.

[2] 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.

[3] 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`.

[4] 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`.

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

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

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

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

[9] 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]`.

[10] 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`.

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