ULMFiT at GermEval-2018: A Deep Neural Language Model for the Classification of Hate Speech in German Tweets

Kristian Rother
Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm
`kristian.rother@hshl.de`

Achim Rettberg
Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm
`achim.rettberg@hshl.de`

Abstract

This paper describes the entry hshl_coarse_1.txt for *Task I (Binary Classification)* of the *Germeval Task 2018 - Shared Task on the Identification of Offensive Language*. For this task, German tweets were classified as either offensive or non-offensive. The entry employs a task-specific classifier built on top of a medium-specific language model which is built on top of a universal language model. The approach uses a deep recurrent neural network, specifically the AWD-LSTM architecture. The universal language model was trained on 100 million unlabeled articles from the German Wikipedia and the medium-specific language model was trained on 303,256 unlabeled tweets. The classifier was trained on the labeled tweets that were provided by the organizers of the shared task.

1 Introduction

Hate speech is on the rise in online communication and can come in different forms but usually follows certain patterns (Mondal et al., 2017). Additionally social media serves as a breeding ground for deviant behavior following real world incidents (Williams and Burnap, 2015).

Hate speech has psychological consequences for the victims such as fear, anger and vulnerability (Awan and Zempi, 2015) as well as the worry that online threats may become a reality (Awan and Zempi, 2016). Additionally, hate speech can be the harbinger of actual violence. Hate speech towards a group can serve as a predictor of violence towards that group (Müller and Schwarz, 2018a) and Twitter use can fuel hate-crimes (Müller and Schwarz, 2018b).

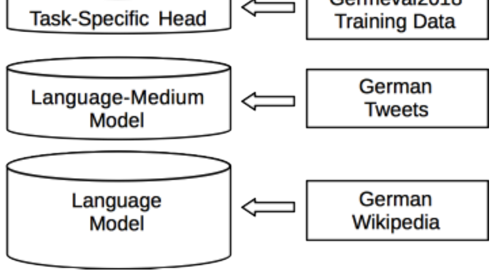
Institutions and legislators have reacted to this trend towards hate speech. The European Commission and multiple social media companies agreed to a code of conduct on countering illegal hate speech online (European Commission, 2016). Germany passed the *Network Enforcement Act* on September 1st 2017 to enforce fines of up to 50 million Euros against social media companies that fail to delete illegal content (German Bundestag, 2017). The law specifically includes hate speech (§§130, 166 and 185-187 of the Criminal Code).

Due to the negative impact of hate speech and the amount of social media data that is generated every day, automated detection and classification of hate speech has been studied widely. Recent overviews can be found in (Schmidt and Wiegand, 2017) and (Fortuna and Nunes, 2018). However, with some exceptions such as (Ross et al., 2017) and (Van Hee et al., 2015), the scope of the studies is often limited to the English language. Therefore, this paper tries to contribute to the improvement of the state of the art in German hate speech detection by describing the entry hshl_coarse_1.txt which participated in the binary classification task at Germeval 2018.

2 Experimental Setup

The overall setup closely follows the ULMFiT method (Howard and Ruder, 2018) as depicted in figure 1.

Figure 1: ULMFiT Overview.



The general idea is to split the training process into three parts. First, a *language model (LM)* is trained from a large corpus of unlabeled data.

This model is used as the basis to train a *language-medium model (LMM)* from unlabeled data that matches the desired medium of the task (e.g. forum posts, newspaper articles or tweets). Finally, a *task-specific head (TSH)* like a hate speech classifier or a sentiment classifier is trained on top of this model from a labeled dataset. This approach facilitates the reuse of pretrained models for the lower layers.

2.1 Technical Resources

All experiments were conducted in Jupyter Notebooks (Kluyver et al., 2016) running Python 3 (Python Software Foundation, 2018) kernels with the following libraries:

- pytorch (Paszke et al., 2017)
- fast.ai (Howard and others, 2018)
- pandas (McKinney, 2010)
- numpy (Oliphant, 2006)
- scikit-learn (Pedregosa et al., 2011)
- matplotlib (Hunter, 2007)
- spaCy (Honnibal and Montani, 2018)

All models were trained on a desktop computer with an Intel i7-6850 CPU, 32 GB of RAM and a GTX 1080 GPU with 8 GB of RAM. A fixed seed was used for the random number generators.[1]

2.2 Data

To train the entire model end to end, three data sources were used. The language model was trained on a dump of the entire German Wikipedia. Only the top 100 million articles (with a character length of at least 2,000) were kept and the vocabulary was limited to 50,000 tokens. Because this is the same approach as Wikitext-103 (Merity et al., 2016) the model will be called W103-DE-50k.

The language-medium model was trained on 303,256 unlabeled tweets[2] that were collected with a custom script using the Twitter-Streaming-API.

Finally, to train the task-specific head, the 5,009 labeled tweets that were provided by the Germeval 2018 competition organizers were used. The data is summarized in table 1 and the distribution of the labels for the binary classification task is shown in table 2

Model	Medium	Items	Type
TSH	Twitter	5,009	Labeled
LMM	Twitter	303,256	Unlabeled
LM	Wikipedia	100,000,000	Unlabeled

Table 1: Training Data.

Category	Items	Percent
Offensive	1,688	33.7
Other	3,321	66.3

Table 2: Frequencies of the Categories in the Training Set.

spaCy was used to tokenize the data and some additional preprocessing as in (McCann et al., 2017; Johnson and Zhang, 2017) was applied. Both the Wikipedia and Twitter data was sanitized with a custom function by replacing html-code and other unwanted characters with sensible ones (e.g., replacing nbsp; with a space or <br /> with a newline). For the Wikipedia data, labels for the beginning of an article and for the beginning of a paragraph were added. For the tweets only a beginning of tweet token was added and all @username occurrences were replaced with the label x_user_mention and all urls were replaced with x_url_mention. Finally, special tokens for upper-case words, elongation, repetition, unknown and padding were inserted. For the language model, the vocabulary was capped at the most frequent 50,002 tokens with a minimum frequency of 5. The medium-language model has a vocabulary of 33,191 tokens.

All datasets were split into a training and a validation set by randomly separating 10% of the data from the rest.

2.3 Architecture

Due to the sequential nature of the task, a recurrent neural network (RNN) architecture was employed. Specifically, the weight-dropped AWD-LSTM variant (Merity et al., 2017) of the long short-term memory network (Hochreiter and Schmidhuber, 1997) and (Gers et al., 1999) was used. The chosen embedding size was 400, the number of hidden activations per layer was 1150 and the number of layers was 3. For the classifier, two linear blocks with batch normalization and dropout were added

[1]With this setup, the training of one epoch of the language model took approximately 2 hours and 50 minutes.

[2]A rule of thumb from correspondence at the forums hosted by one of the ULMFiT-authors is to use between 5x and 10x of the available training data for this step.

to the model with rectified linear unit activations for the intermediate layer and a softmax activation at the last layer (Howard and Ruder, 2018).

2.4 Hyperparameters

The hyperparameters are similar across all stages of the ULMFiT method. The batch size was limited by the available GPU memory and always set to the highest possible value. Back propagation through time (BPTT) was set to 70 for all models. Apart from these parameters, the models used different configurations for the learning rate (LR), weight decay (WD), dropouts, cyclical learning rates (CLR) (Smith, 2017) and slanted triangular learning rates (STLR) (Howard and Ruder, 2018). Additionally, gradient clipping (Pascanu et al., 2013) was applied to some of the models.

For the dropouts, the two configurations that are summarized in table 3 were used. They are taken from the Github repository[3] corresponding to (Howard and Ruder, 2018) and the Github repository[4] corresponding to (Merity et al., 2017). The dropout multiplier, when configured, is applied to all dropouts. For the CLR the four parameters are maximum to minimum learning rate divisor, cooldown percentage, maximum momentum and minimum momentum in that order and for the STLR the parameters are maximum to minimum learning rate divisor and cut_fract.

Dropout	Howard	Merity
Input Layer	0.25	0.6
General	0.1	0.4
LSTM's Internal	0.2	0.5
Embedding Layer	0.02	0.1
Between LSTM Layers	0.15	0.2

Table 3: Dropout configurations.

2.4.1 Language Model

To obtain a sensible learning rate, the learning rate finder (LRF) introduced by (Smith, 2017) was used. The graph for the LRF is depicted in figure 2.

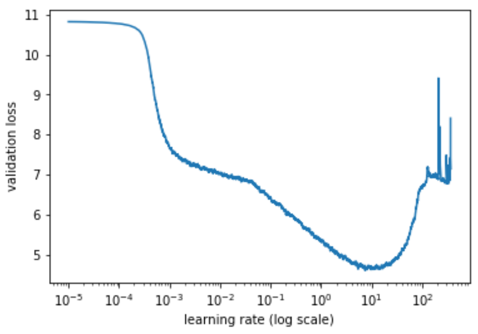
The hyperparameters for model C are directly transfered from (Howard and Ruder, 2018)[5]. The hyperparameters for model D are a variation of

[3]`https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts`

[4]`https://github.com/Smerity/awd-lstm-lm`

[5]Specifically, these up to date parameters were used: `https://github.com/fastai/fastai/tree/master/courses/dl2/imdb_scripts`

Figure 2: LRF - Language Model.



these parameters. The hyperparameters for model A, B and E were deduced from the learning rate finder and some short experiments.

Discriminative learning rates (Howard and Ruder, 2018) of [lr/6, lr/4, lr, lr] were used for models C and D for the four layer-groups. A fixed learning rate was used for all other models.

The batch size for all language models was set to 32 and a BPTT of 70 was used. A gradient clipping of 0.4 and 0.12 was applied to model B and D respectively. Model C used STLR with a ratio of 32 and a cut_fract of 0.1. Models A, B and E used CLR with the parameters 10, 10, 0.95, 0.85 and model C used CLR with the parameters 10, 33, 0.8, 0.7. Adam was used as the optimizer for models C and D and stochastic gradient descent was used for the other models. Table 4 summarizes the remaining hyperparameters.

Model	LR	WD	Dropout
A	2	1e-7	Howard * 0.5
B	1.4	1e-7	Howard * 0.4
C	3e-4	1e-7	Howard * 0.5
D	2e-3	1e-6	Merity * 0.2
E	5.12	1e-7	Merity * 0.5

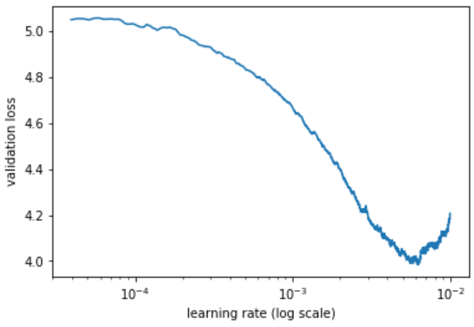
Table 4: Language Model Hyperparameters.

2.4.2 Language-Medium Model

A learning rate finder was used to determine suitable candidate learning rates. The graph is depicted in figure 3.

The batch size for all language-medium models was set to 32 and a BPTT of 70 was used. The weight decay was set to 1e-7 for all models and no gradient clipping was used. The model was gradually unfrozen (Howard and Ruder, 2018) by unfreezing the last layer first and then unfreezing all remaining layers. Slanted triangular learning

Figure 3: LRF - Language-Medium Model.



rates (Howard and Ruder, 2018) with a ratio of 32 and a cut_fract of 0.5 were used after the last layer was unfrozen and a ratio of 20 and a cut_fract of 0.1 was used when all layers were unfrozen. The hyperparameters of all four models are summarized in table 5. The columns LR-Last and LR-All refer to the learning rates for the runs were only the last layer was unfrozen and where all layers were unfrozen.

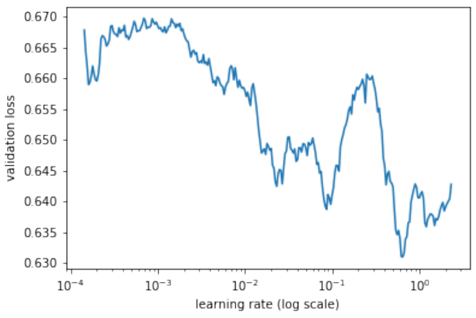
Model	LR-Last	LR-All	Dropout
LMM1	3e-3	3e-3	Howard * 0.7
LMM2	1e-3	1e-3	Howard * 0.7
LMM3	4e-3	3e-3	Howard * 0.3
LMM4	5e-3	1e-3	Howard * 0.3

Table 5: Language-Medium Model Hyperparameters.

2.4.3 Task-Specific Head

A learning rate finder (see figure 4) was used to find the learning rate of 3e-1. The batch size for the classifier was set to 52 and a BPTT of 70 was used.

Figure 4: LRF - Task-Specific Head.



The model was gradually unfrozen layer by layer with the same hyperparameters applied to each layer. The weight decay was set to 1e-7. Cyclical learning rates with the parameters 10,10,0.98 and 0.85 were used. The Howard dropouts were used with a multiplier of 1.8 and no gradient clipping was applied. The optimizer was stochastic gradient descent.

3 Experiments

The working hypothesis that an increased performance of the lower layers improves the results at the upper layers lead to the decision to try five different hyperparameter configurations for the language model. The models A-E were trained on unlabeled Wikipedia data for 25 epochs. Model A was trained for an additional 50 epochs and used as the basis for the language-medium model.

Four different hyperparameter configurations for the language-medium model (LMM1-LMM4) were trained on unlabeled tweets for 30 epochs each. Afterwards the best model (LMM1) was used as the basis for the hate speech classifier.

Lastly, The hate speech classifier was trained on the provided training data of labeled tweets. The hyperparameters were tuned during experimentation by picking a learning rate that lead to convergence with overfitting and regularizing via the other parameters until the model didn't overfit anymore.

4 Results

The perplexities for the language models are depicted in table 6. Models A, B and E outperformed the other models and converged while models C and D were underfitting. The perplexity of the best LM is 27.39 which is better than the best perplexity for a non-ensembled English language model on the One Billion Word benchmark (30.0) that was reported in the summary by (Jozefowicz et al., 2016) and the current state of the art (28.7) for the same corpus (Bakhtin et al., 2018). For comparison, the best published result for the English Wikitext-103 is 40.8 (Grave et al., 2016). To our knowledge, the best perplexity for a word level German language model that uses the Wikipedia is 36.95 (Van Hee et al., 2015).

After 30 epochs, the language-medium model LMM1 showed the best overall result with a perplexity of 17.64.

To get a feeling for the quality of the hate speech classifier, the labels for the validation set were predicted. Table 7 summarizes the results.

Model	Validation Loss	Perplexity
A	3.31	**27.39**
B	3.41	30.27
C	3.81	45.15
D	3.68	39.65
E	3.38	29.37

Table 6: Language Model Perplexities. Lower is better.

Class	Precision	Recall	F1	Support
Offensive	0.73	0.68	0.71	179
Other	0.83	0.86	0.85	322
Average	0.8	0.8	0.8	

Table 7: Results Binary Classification.

5 Conclusion

The paper presented the submission hshl_coarse_1.txt that was entered for the binary hate speech classification task of Germeval 2018. It used a deep recurrent neural net, specifically an AWD-LSTM architecture, to classify German tweets as offensive or non-offensive.

A three layered approach based on the ULMFiT method was used to train the classifier. First, a German language model was trained from unlabeled Wikipedia data. A language-medium model for German tweets was trained on top of this model from unlabeled tweets and served as the backbone to train the hate speech classifier on the provided labeled training data. This classifier achieved an average F1 score of 80 on the validation data.

All relevant code will be made available at one of the authors' Github repositories[6]. The German language model with a vocabulary size of 50,000 tokens achieved a perplexity of 27.39. It will be released as W103-DE-50k and a link will be added to the repository.

6 Outlook

The proposed approach towards hate speech classification can be improved in various ways. The working hypothesis that better lower layer results improve the classifier needs empirical support but assuming it holds, the overall results could be improved by improving the lower layers. Instead of relying on a single model at each stage an ensemble of models could be used. A good starting point would be turning all models into bidirectional models (Peters et al., 2017). Different architectures such as Quasi Recurrent Neural Networks (Bradbury et al., 2016) or Contextual LSTM (Ghosh et al., 2016) or general improvements like continuous caches (Grave et al., 2016) could improve the overall results further.

The idea of super-convergence (Smith and Topin, 2017) might also be worth investigating and some of the ideas outlined in the overview by (Schmidt and Wiegand, 2017) could be tried.

Lastly, hate speech dictionaries could be used to construct a keyword-filter for the Twitter-API to collect more data for the offensive category to improve the classifier by effectively increasing the size of the training set.

[6]`https://github.com/rother`

7 Acknowledgements

We thank the Behr-Hella Thermocontrol GmbH for supporting this research. We also thank all reviewers and the competition organizers.

References

Imran Awan and Irene Zempi. 2015. We fear for our lives: Offline and online experiences of anti-Muslim hostility. *Report,[online] available: http://tellmamauk. org/wp-content/uploads/resources/We% 20Fear% 20For% 20Our% 20Lives. pdf [accessed: 7 January, 2016].*

Imran Awan and Irene Zempi. 2016. The affinity between online and offline anti-Muslim hate crime: Dynamics and impacts. *Aggression and violent behavior*, 27:1–8.

Anton Bakhtin, Arthur Szlam, Marc'Aurelio Ranzato, and Edouard Grave. 2018. Lightweight Adaptive Mixture of Neural and N-gram Language Models. *arXiv preprint arXiv:1804.07705*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-Recurrent Neural Networks. *arXiv:1611.01576 [cs]*, November. arXiv: 1611.01576.

European Commission. 2016. Code of conduct on countering illegal hate speech online. `http://ec.europa.eu/newsroom/document.cfm?doc_id=42985`.

Paula Fortuna and Sérgio Nunes. 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, 51(4):85:1–85:30, July.

German Bundestag. 2017. Act to improve enforcement of the law in social networks (network enforcement act). `https://www.bmjv.de/SharedDocs/Gesetzgebungsverfahren/Dokumente/NetzDG_engl.pdf?__blob=publicationFile&v=2`.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, pages 850–855.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual LSTM (CLSTM) models for Large scale NLP tasks. *arXiv:1602.06291 [cs]*, February. arXiv: 1602.06291.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew Honnibal and Ines Montani. 2018. spaCy library. `https://spacy.io`.

Jeremy Howard et al. 2018. fast.ai library. `https://github.com/fastai/fastai`.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *arXiv:1801.06146 [cs, stat]*, January. arXiv: 1801.06146.

J. D. Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570. Association for Computational Linguistics.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Wes McKinney. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mainack Mondal, Leandro Arajo Silva, and Fabrício Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 85–94. ACM.

Karsten Müller and Carlo Schwarz. 2018a. Fanning the Flames of Hate: Social Media and Hate Crime. CAGE Online Working Paper Series 373, Competitive Advantage in the Global Economy (CAGE).

Karsten Müller and Carlo Schwarz. 2018b. Making America Hate Again? Twitter and Hate Crime under Trump.

Travis E. Oliphant. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS 2017 Workshop Autodiff*.

Fabian Pedregosa, Gal Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv:1705.00108 [cs]*, April. arXiv: 1705.00108.

Python Software Foundation. 2018. Python programming language. `https://python.org`.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Leslie N. Smith and Nicholay Topin. 2017. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. *arXiv:1708.07120 [cs, stat]*, August. arXiv: 1708.07120.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 464–472. IEEE.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Vronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 672–680.

Matthew L. Williams and Pete Burnap. 2015. Cyberhate on social media in the aftermath of Woolwich: A case study in computational criminology and big data. *British Journal of Criminology*, 56(2):211–238.