Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)

Logistic Regression and Naive Bayes for Hierarchical Multi-label Classification at GermEval 2019 - Task 1

Kristian Rother and **Achim Rettberg**
Hochschule Hamm-Lippstadt
Marker Allee 76-78
59063 Hamm, Germany
kristian.rother@hshl.de
achim.rettberg@hshl.de

Abstract

This paper describes an entry of two systems for Task 1 of GermEval 2019. Task 1 consists of classifying labels of genres for German books based on short advertisement text blurbs. It is split into Subtask A, a multi-label classification task with 8 classes and Subtask B a hierarchical multi-label classification task with 343 different classes. The submitted systems were used for both subtasks and combined Logistic Regression and Naive Bayes to build a classifier. To reach the final systems, different models and combinations of hyperparameters were explored empirically. The best submitted system reached micro-averaged F1-scores of 0.82 and 0.62 for the two subtasks respectively.

1 Introduction

Hierarchical multi-label classification (HMC) of blurbs is the task of classifying multiple labels for a short descriptive text, where each label is part of an underlying hierarchy of categories. The increasing amount of available digital documents and the need for more and finer grained categories calls for new, more robust and sophisticated text classification methods. Large datasets often incorporate a hierarchy which can be used to categorize information of documents on different levels of specificity. The traditional multi-class text classifcation approach is thoroughly researched, however, with the increase of available data and the necessity of more specific hierarchies and since traditional approaches fail to generalize adequately, the need for more robust and sophisticated classification methods increases (Remus et al., 2019).

To advance the state of the art in HMC, Task 1 of GermEval 2019 was launched. Task 1 consists of classifying labels of genres for German books based on short advertisement text blurbs. It is split into Subtask A, a multi-label classification task with 8 classes and Subtask B a hierarchical multi-label classification task with 343 different classes. For Subtask B, each label is part of an underlying hierarchy of categories. This subtask is thus a Hierarchical multi-label classification (HMC) problem.

HMC problems exist in a variety of domains from text classification tasks to the prediction of gene functions. The text classification research is mostly focused on problems in English. Typically, (deep) neural models are applied (Liu et al., 2017; Gargiulo et al., 2019; Shimura et al., 2018; Cerri et al., 2014). But there are also other approaches like decision trees (Vens et al., 2008) or genetic algorithms (Cerri et al., 2012; Gonçalves et al., 2018). An overview of multi-label classification algorithms can be found in (Sharma and Mehrotra, 2018).

To compete in GermEval 2019 and to contribute to the state of the art in German HMC, two systems, that only vary in the n-grams that they used, were submitted:

- *HSHL_LogisticRegression_NaiveBayes1* (*from here on out referred to as System 1*)
- *HSHL_LogisticRegression_NaiveBayes2* (*from here on out referred to as System 2*)

Apart from participating in GermEval 2019, the intended use case for the submitted systems is to serve as an introductory tool for machine learning in short talks or lectures and as a baseline for more complex systems. As such, some additional constraints were imposed on the systems that go beyond the scope of the GermEval task:

- End-to-end runtime for Subtask A around 10 minutes.
- End-to-end runtime for Subtask B around 45 minutes.

- No use of additional data.
- One system, no ensemble.

2 Data

The dataset (BGC-DE) was crawled from Random House and provided by the organizers. It was split into 70% training data, 10% validation data and 20% test data. The labels for the test data were not given to the participants because this data served as the means to judge the submissions. The dataset contains information on German books in XML-format.

Apart from the advertisement text blurbs and genres, some additional metadata (title, author, URL, date of publication) and the ISBN of the book were provided. Some quantitative characteristics of the dataset are summarized in table 1.

Number of samples	20,784
Average length of blurb	94.67
Total number of classes	343 (8, 93, 242)

Table 1: Characteristics of the BGC-DE dataset. The 343 classes form a hierarchy of 8 classes at the first level, 93 classes at the second level and 242 classes at the third level.

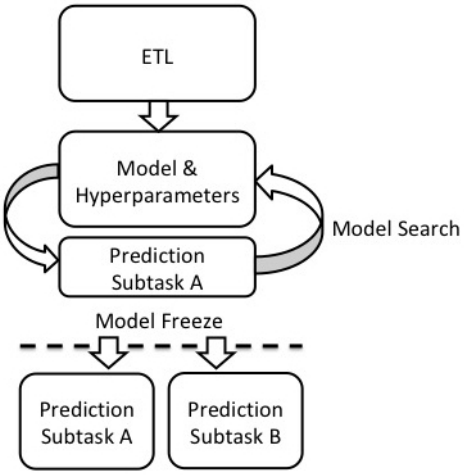
After an initial preparation phase in which a tiny sample of data was provided to check the file format, the data was released in two phases. In phase one, a validation set was released which consisted of labeled training data and corresponding unlabeled test data. After phase one ended, the labels for the training data of the validation set were released and the validation set essentially became the new training set for phase two with new unlabeled training data provided for this phase.

3 Experimental Setup

To produce a submission for the task, the provided data was first loaded, sanitized and converted to a format that is suitable for the use in machine learning, namely a pands DataFrame (ETL). Because some entries had no blurb, the book title was used as a filler value. Additionally the genre names were sanitized by lowercasing them and removing special characters and spaces so they could be used as labels for the DataFrame. No additional outside data was used.

To find a model and suitable hyperparameters, the provided training data from phase one of the competition was split into 80% training data and 20% development data. Afterwards, different models and hyperparameters were trained and evaluated for Subtask A on this new split (Model Search). This train-evaluate cycle was repeated iteratively until no further improvements could be made (Model Freeze). The chosen model was used to make the final predictions for both Subtask A and Subtask B. The overall process is show in figure 1.

Figure 1: Overall process for the competition.



3.1 Technical Resources

All experiments were conducted in Jupyter Notebooks, version 4.0.2 (Kluyver et al., 2016) running a Python 3.5.0 (Python Software Foundation, 2019) kernel with the following libraries:

- pandas 0.23.4 (McKinney, 2010)
- NumPy 1.11.3 (Oliphant, 2006)
- scikit-learn 0.20 (Pedregosa et al., 2011)
- spaCy 2.0.12 (Honnibal and Montani, 2019)

A fixed seed was used for the random number generators. All models were trained on an end of 2013 MacBook Pro with a 2 GHz Intel Core i7, 8 GB 1600 MHz DDR3 and an Intel Iris Pro 1536 MB GPU[1].

4 Model Search

In order to find a model and suitable hyperparameters, seven different models from the scikit-learn

[1]With this setup, the entire end-to-end training on the provided development set took 8:24 minutes for Subtask A and 46:55 minutes for Subtask B.

library were trained and evaluated on Subtask A. The selected models were Decision Tree, Random Forest, Multinomial Naive Bayes, K-nearest Neighbors (KNN), SVC, Linear SVC and Logistic Regression. To get a baseline for each model, the default values of the library were used. Afterwards, each model was optimized until no further improvements could be made. All parameters were found empirically by trying different combinations[2]. The chosen hyperparameters of the optimized models are summarized in the following chapters with a list of the used parameters in Python Code for each model.

4.1 Decision Tree

For the decision tree, a random splitter was used and the minimum number of samples was set to 15. The default "Gini-criterion" yielded better results than the "Entropy-criterion". Balanced class weights or changing the number of features for the split to anything other than the total number of features did not improve the results.

- splitter='random'
- min_samples_split=15

4.2 Random Forest

For the random forest, 200 estimators were used and the minimum number of samples required to split an internal node was set to 5. Additionally, no bootstrapping was used and balanced subsamples were used for the class weights.

- n_estimators=200
- min_samples_split=5
- bootstrap=False
- class_weight='balanced_subsample'

4.3 Multinomial Naive Bayes

For Multinomial Naive Bayes, only the alpha value for smoothing was tuned. An alpha value of 0.08 yielded the best results. The default of learning the class prior probabilities outperformed using uniform priors.

- alpha=0.08

[2]The exact experiments that were conducted can be found in the lab notes that accompany the code at https://github.com/rother/germeval2019

4.4 K-nearest Neighbors

For KNN, a total of 9 neighbors were used. Weighting points by the inverse of their distance instead of weighting them equally (uniform) provided the best results.

- weights='distance'
- n_neighbors=9

4.5 Support Vector Machine - SVC

For the SVC, tuning the C value for regularization proved most useful. Ultimately, it was set to 15,900. Furthermore, balanced class weights were used. No other experiments yielded improvements.

- C=15900.0
- class_weight='balanced'

4.6 Support Vector Machine - Linear SVC

Due to technical problems[3] the Linear SVC had to be constructed by passing kernel='linear' and probability=True to SVC instead of using LinearSVC directly. The only optimization that was performed was using balanced class weights, which improved the overall results.

- kernel='linear'
- probability=True
- class_weight='balanced'

4.7 Logistic Regression

For the logistic regression, a liblinear solver with a maximum number of 1000 iterations, automatic multiclass fitting and balanced class weights was used. L2 regularization with C=40.0 was applied. No dual formulation was used as the number of samples was bigger than the number of features.

- C=40.0
- dual=False
- multi_class='auto'
- max_iter=1000
- class_weight='balanced'

[3]Namely, that LinearSVC does not provide a predict_probab() method.

5 Results of the Model Search experiments

The micro-averaged F1-scores for all optimized models when evaluated on Subtask A are summarized in table 2. The table is ordered in ascending order of F1-scores. The Logistic Regression model performed best and was thus picked for the competition.

#	Model	Default	Optimized
1	Decision Tree	0.6049	0.6125
2	Random Forest	0.6125	0.6667
3	Naive Bayes	0.6253	0.7703
4	KNN	0.7164	0.7377
5	SVC	0.5127	0.7878
6	Linear SVC	0.7731	0.7882
7	**Logistic Regression**	0.6919	**0.7883**

Table 2: Overview of micro-averaged F1-scores for the default configuration of models and optimized versions. Bold indicates the best model.

Additionally, one mixed soft-voting ensemble that combined all models and ensembles that combined the two or three best models were created[4]. The ensembles that used the top models improved upon the best F-score as summarized in table 3. However, they were not used for the submission as explained in the introduction.

#	Ensemble Name	Models	F-score
1	All	1, 2, 3, 4, 7	0.7710
2	Top 2b	3, 7	**0.7967**
3	Top 3	3, 4, 7	**0.7950**
4	Top 2a	4, 7	**0.8001**

Table 3: Overview of ensembles with micro-averaged F1-scores. Bold indicates improvements over the best single model.

6 Model Freeze

Because the Logistic Regression model yielded the best results during Model Search, it was used for the submissions. Both submissions combined this Logistic Regression model with a Naive Bayes approach similar to (Wang and Manning, 2012)[5] to classify the genres of German books from advertisement text blurbs. spaCy was used as the tokenizer and unicode accents were stripped but the casing was kept for both submissions. Both lemmatiziation and stopwords lowered the results and thus were not used. Empty blurbs were replaced with the title of the book. The tokenization parameters are summarized in table 4.

Parameter	Value
Tokenizer	spaCy (German)
Accent Stripping	Unicode
Lowercase	No
Lemmatization	No
Stopwords	No
Replace-Empty Blurbs with	Title

Table 4: Tokenization parameters.

For the term-frequency matrices, only words that appeared in at least 4 documents were used and words that appeared in more than 40% of documents were ignored. Inverse document-frequency-reweighting, sublinear term frequency scaling and smoothing were applied. The parameters are summarized in table 5. System 1 and System 2 only differ in the n-grams that were used to construct the term-frequency matrices. For System 1, only unigrams were used and for System 2, bigrams were used.

Parameter	Value
Minimum Number of Documents	4
Maximum Frequency of Documents	40%
Inverse document-frequency-reweighting	Yes
Sublinear Term Frequency Scaling	Yes
Smoothing	Yes

Table 5: Parameters for the term-frequency matrices.

For the logistic regression, the model from the Model Search stage was used. The hyperparameters are summarized in table 6.

Parameter	Value
Solver	liblinear
Maximum Iterations	1000
Multiclass	Automatic
Class Weights	Balanced
Dual Formulation	No
Regularization	L2
C	40.0

Table 6: Parameters for the logistic regression.

[4]Linear SVC and SVC were not included in these ensembles due to technical problems.

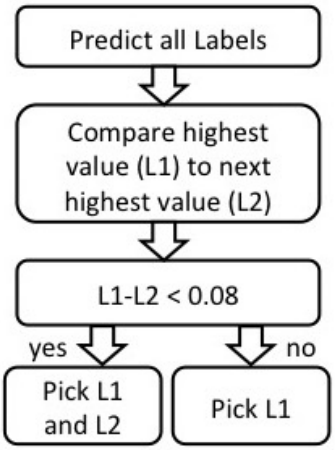
[5]See also https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline

7 Classification procedure

For Subtask A up to two labels were classified per blurb, because using more than two labels lead to worse results. For the second label a limit of 0.08 was found to be optimal. This limit means that a second label is only classified, if the difference between the probabilities of the top two predicted labels is less than this value. This is depicted in figure 2.

For Subtask B, the system started with the classified level 1 labels from Subtask A. Based on these labels it moved up one level on the hierarchy and added exactly one additional level 2 label per level 1 label if a cutoff value was met. Afterwards, the system once again moved up a level on the hierarchy and added level 3 classifications. This process is depicted as Algorithm 1. The exact procedure is explained in more detail below.

Figure 2: Subtask A classification process.



On the second level, a list of all level 2 labels that follow the classified level 1 labels on the hierarchy was generated. The level 2 label with the highest probability was picked and added as a classification if the probability was higher than a provided cutoff value. Note that on level 2, only upto one label per level 1 label was picked.

For the third level, the level 2 classifications were used as a basis. If there was a classified level 2 label, a list of all level 3 labels that follow the classified level 2 label on the hierarchy was generated. If there was only one level 2 label (and thus also only one level 1 label) the system added as many level 3 labels as met the provided multi-cutoff for level 3. If there was a second level 2 label (and thus also a second level 1 label) a single level 3 label was added, if it met the provided cut-

Algorithm 1 Subtask B classification process.

$Level1Labels \leftarrow SubtaskA$
for all $Level1Labels$ **do**
  $Level2Probabilities \leftarrow get\_from\_l1()$
  $max\_l2 \leftarrow Level2Probabilities(0)$
  if $max\_l2 > 0.09$ **then**
    $Level2Labels \leftarrow get\_label\_l2()$
  end if
end for
if $len(Level2Labels) \neq empty$ **then**
  $ProbA \leftarrow get\_from\_l2(Level2Labels(0))$
  $ProbB \leftarrow get\_from\_l2(Level2Labels(1))$
  for all $ProbA$ **do**
    if $probability > 0.15$ **then**
      $Level3Labels.append(get\_label\_l3())$
    end if
  end for
  if $ProbB \neq null$ **then**
    $max\_l3 \leftarrow ProbB(0)$
    if $max\_l3 > 0.7$ **then**
      $Level3Labels.append(get\_label\_l3())$
    end if
  end if
end if

off value. This ensured that the lower confidence prediction from the earlier level can get at most one additional level 3 label.

The cutoff values were found empirically and are summarized in table 7. On the third level, a multi-cutoff value of 0.15 was used for the first level 2 label and a cutoff of 0.7 was used if there was a second level 2 label.

Level	Cutoff Value	Number of Labels
2	0.09	0 to 1 per level 1 label
3	0.15 / 0.7	0 to n per level 2 label

Table 7: Cutoff values and maximum number of labels for the classification procedure.

8 Results

Unigrams (System 1) performed slightly better than bigrams (System 2). Both systems participated in Subtask A and Subtask B of the competition. The final results are summarized in table 8 with the best results highlighted in bold.

System	F1-score A	F1-score B
System 1	**0.8201**	**0.6161**
System 2	0.8163	0.6063

Table 8: Final results. F1-scores are micro-averaged.

9 Conclusion

This paper presented two submission (System 1 and System 2) that were entered for Task 1 at GermEval 2019.

Both submissions use a combined Logistic Regression/Naive Bayes approach to classify genres of German books from advertisement text blurbs. spaCy was used as the tokenizer and unicode accents were stripped but the casing was kept for both submissions. The first submission uses unigrams and the second submission uses bigrams. The other hyperparameters are the same.

For the term-frequency matrices, only words that appeared in at least 4 documents were used and words that appeared in more than 40% of documents were ignored. Inverse document-frequency-reweighting, sublinear term frequency scaling and smoothing were applied.

For the logistic regression, a liblinear solver with a maximum number of 1000 iterations, automatic multiclass fitting and balanced class weights was used. L2 regularization with C=40.0 was applied. No dual formulation was used as the number of samples was bigger than the number of features.

The cutoff values for the classification procedure are summarized in table 7.

To facilitate the reproduction of the results, all code is made available as a Jupyter Notebook at one of the authors Github repositories[6]. Additional lab notes on the conducted experiments will also be made available at the same repository.

The end-to-end execution time on a consumer grade laptop was 8:24 minutes for Subtask A and 46:55 minutes for Subtask B and can be reduced further by storing and reloading intermediate results.

The unigram version (System 1) performed best and reached micro-averaged F1-scores of 0.82 and 0.62 for the two subtasks respectively. When only the best entry for each teams is counted, this means compared to the other participants rank 7 of 9 was reached for Subtask A and rank 5 of 6 was reached for Subtask B. The winning scores were 0.87 and 0.68 for the two tasks respectively which means that the submission was 5.75% worse than the winning entry for Subtask A and 5.88% worse than the winning entry for Subtask B.

As one of the intended use cases for the submitted systems was to serve as a baseline system in the future it is worth noting that they outperformed the provided baseline by the organizers (a linear SVN that scored 0.80 and 0.53 on the tasks) for both tasks.

[6]https://github.com/rother/germeval2019

Acknowledgments

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

References

Ricardo Cerri, Rodrigo C. Barros, and Andre CPLF de Carvalho. 2012. A genetic algorithm for hierarchical multi-label classification. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 250–255. ACM.

Ricardo Cerri, Rodrigo C Barros, and André CPLF De Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39–56.

Francesco Gargiulo, Stefano Silvestri, Mario Ciampi, and Giuseppe De Pietro. 2019. Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, 79:125–138.

Eduardo Corrêa Gonçalves, Alex A Freitas, and Alexandre Plastino. 2018. A survey of genetic algorithms for multi-label classification. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Matthew Honnibal and Ines Montani. 2019. spaCy library. https://spacy.io. Accessed: 2019-08-07.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and et al. 2016. Jupyter notebooks - a publishing format for reproducible computational workflows. In *ELPUB*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

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

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

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

Python Software Foundation. 2019. Python Programming Language. https://python.org. Accessed: 2019-08-07.

Steffen Remus, Rami Aly, and Chris Biermann. 2019. Germeval-2019 task 1: Shared task on hierarchical classification of blurbs. In *Proceedings of the GermEval 2019 Workshop*, Erlangen, Germany.

Seema Sharma and Deepti Mehrotra. 2018. Comparative analysis of multi-label classification algorithms. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 35–38. IEEE.

Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. Hft-cnn: Learning hierarchical category structure for multi-label short text categorization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 811–816.

Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, pages 90–94. Association for Computational Linguistics.