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

Convolutional Neural Networks for Classification of German Blurbs

Erdan Genc, Louay Abdelgawad, Viorel Morari, Peter Kluegl

Averbis GmbH, Freiburg, Germany
{firstname}.{lastname}@averbis.com

Abstract

This paper presents the submission of the system AVERBIS_BOHB_CNN for the Shared Task on hierarchical classification of German blurbs (short texts) - GermEval 2019 Task 1. We optimized the hyperparameters of a CNN based on a fastText word embedding layer and combined it with a variant of a T-Criterion classification method. The model was able to achieve an F1 score of 0.834 (ranked 6th) for subtask a and of 0.644 (ranked 3th) for subtask b on the respective test sets.

1 Introduction

Hierarchical multi-label text-classification (HMC) is the task of classifying text into categories with an underlying hierarchical structure. As more and more text becomes available in digital form the need for such automated and robust text classification grows bigger. To foster research in the domain of HMC the organizers of GermEval 2019 called for participation in Shared Task 1 - hierarchical classification of German blurbs. In this work, we describe our submission to the task.

The task consists of two subtasks (subtask_a, subtask_b) in which the participants are challenged to classify short German text snippets advertising books into one or multiple of 8 non-hierarchically (a) and 343 (b) hierarchically structured categories, respectively.

We approach the subtasks with a convolutional neural net (CNN) (Kim, 2014), using word embeddings trained with fastText (Bojanowski et al., 2017). In order to optimize the involved hyperparameters, we used BOHB (Falkner et al., 2018). BOHB is a hyperparameter search technique, which combines Bayesian optimization with bandit-based methods to find good configurations in feasible time. The hierarchy of labels in subtask_b was not incorporated in the learning process, i.e. the labels' hierarchical structure was flattened. To further improve the predictions based on the probability distributions, we adapted the T-Criterion (Boutell et al., 2004) classification method. The approach reaches F1 scores of 0.850 on the validation set of subtask_a and 0.664 on subtask_b.

The rest of the paper is structured as follows. The next Section (2) describes the provided dataset. Section 3 introduces the preliminaries. In Section 4, the system description is layed out. Section 5 illustrates the results and Section 6 summarizes the learnings and gives an outlook for future work.

2 Dataset

The dataset is a collection of short German text sequences, so called blurbs, advertising German books. Figure 1 shows an example of a blurb. Each instance features different fields such as *title*, *body*, *copyright*, *categories*, *authors*, *published*, *isbn*, *url*. The *body* is the main text field and contains the advertising description.

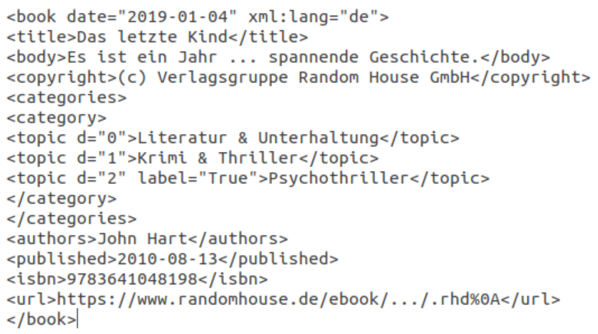
The *categories* are the target classes of the classification tasks. Each blurb can be classified into one or multiple *categories*. Each *category* consists of one or many hierarchically structured *topics*.

For subtask_a the blurbs need to be classified into one or multiple of the eight first level classes, i.e. root level topics ($d = 0$):

- Literatur & Unterhaltung
- Sachbuch
- Kinderbuch & Jugendbuch
- Ratgeber
- Ganzheitliches Bewusstsein
- Glaube & Ethik
- Künste
- Architektur & Garten

For `subtask_b`, the blurbs need to be classified into multiple hierarchical classes, i.e. topics ($d \in \{1, 2\}$). In total there are 343 topics with a maximum level depth of three.

The complete dataset contains 20,784 instances of German blurbs. 14,548 labeled instances for training (`train`) as well as 2,079 labeled instances for local validation (`dev`). All results in this work are based on these two sets. A third set of 4,157 unlabeled instances was made available for result submission (`test`).



```
<book date="2019-01-04" xml:lang="de">
<title>Das letzte Kind</title>
<body>Es ist ein Jahr ... spannende Geschichte.</body>
<copyright>(c) Verlagsgruppe Random House GmbH</copyright>
<categories>
<category>
<topic d="0">Literatur & Unterhaltung</topic>
<topic d="1">Krimi & Thriller</topic>
<topic d="2" label="True">Psychothriller</topic>
</category>
</categories>
<authors>John Hart</authors>
<published>2010-08-13</published>
<isbn>9783641048198</isbn>
<url>https://www.randomhouse.de/ebook/.../.rhd%0A</url>
</book>
```

Figure 1: Example blurb taken from the `dev` set.

With concatenated fields, the average length of a blurb is 231 words with a standard deviation of 64. The smallest blurb counts 66 words, the largest 1017.

The distribution of classes in the `train` set for `subtask_a` are shown in Figure 2. As we can see, the distribution is highly imbalanced with more than half of the instances being labeled with the topic *Literatur & Unterhaltung*.

3 Preliminaries

This section introduces the preliminaries. The support vector machine that serves as baseline in Subsection 3.1, word embeddings (3.2), convolutional neural networks (3.3) and hyperparameter optimization (3.4).

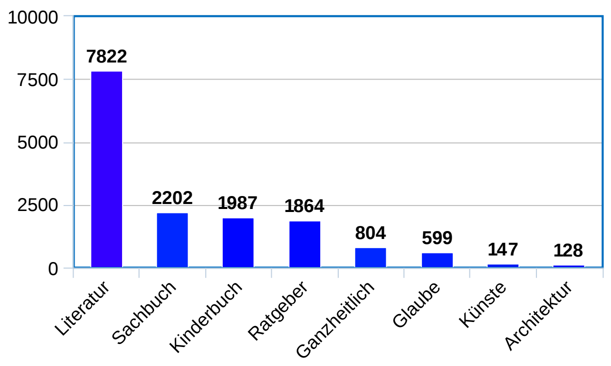


Figure 2: Number of samples per class for `subtask_a` in the `train` set.

3.1 SVM

Support Vector Machines (SVM) are a well established machine learning approach for text classification (Manevitz and Yousef, 2001). In this work, the multi-label classifier is implemented by several one-vs-all linear SVMs (Fan et al., 2008). The features of the single SVMs consist of a bag-of-stems[1]. The stems are prefixed to encode the information to which field it belongs to, e.g. *title_stem* or *body_stem*. The features are weighted using logarithmic frequencies with redundancy (Leopold and Kindermann, 2002) and are L2-normalized. At prediction time the T-Criterion classification method with a threshold of 0.5 is used for all labels. Opposed to state-of-the-art neural classifiers in which the input is encoded using distributed representations for words, e.g. word embeddings, SVMs with a bag-of-words feature representation neglect all sequential information.

3.2 Word Embeddings

A word embedding describes a mapping that translates single words or phrases taken from a vocabulary into an n-dimensional real-valued vector space. Using the context of the words, it is usually the rationale to find a representation which preserves syntactic and semantic attributes and can be passed on to a machine learning algorithm.

The choice of an effective word embedding depends on a large variety of parameters, e.g. the model itself, embedding size, the corpora used for training or the action taken for unknown words. Many standardized and well-described word embeddings, trained on different corpora, can be found online.

[1]https://snowballstem.org/, accessed September 18, 2019

For this work, fastText (Bojanowski et al., 2017) was selected as it additionally considers n-gram sub-words as input instead of whole words as atomic units. Apart from the increased training time, it enables the model to embed unseen words. The word ”Propene”, for example, might be unseen in the training corpus, yet fastText is able to assign a vector near the vector of the seen word ”Propylene” which can be advantageous in domains with complex language.

3.3 CNN

Early improvements in text classification with deep learning started with the Dynamic Convolutional Neural Networks (DCNN) method (Kalchbrenner et al., 2014). The authors first adopted CNNs, a model well-received in the computer vision domain, to the field of NLP. Following this work, Yoon Kim created another CNN architecture (Kim, 2014). The main improvement was to embed the input words using pre-trained word embeddings (Mikolov et al., 2013) before passing them into the neural network. Contrary to other convolutional networks, Kim’s CNN uses a single stage of wide parallel convolutions instead of several stacked convolutions on top of each other. This architecture has been selected as main approach in this work as it has been proven to be versatile and efficient.

3.4 Hyperparameter Optimization

In general the parameters of a neural network can be divided into two categories: the normal and the hyperparameters. Normal parameters, such as weights and biases, are changed during training time. Hyperparameters, such as learning rate and batch size, are set before the training begins. The right choice of parameters can effect the performance of a neural network significantly (Henderson et al., 2017). Therefore, hyperparameter optimization aims to determine a set of hyperparameter values such that a objective function is maximized. These techniques range from simple random search to more sophisticated, efficient methods such as Bayesian Optimization (BO). In this work, BOHB, a state-of-the-art hyperparameter optimization technique developed by Falkner et al., is used to tune the parameters. It combines the best of two worlds leveraging the strong performance of BO while maintaining the speed of hyperband (HB).

4 System Description

This section presents a detailed description of the used system and how the experiments were conducted. It introduces the preprocessing steps, used word embeddings, the model itself as well as the hyperparameter optimization steps and the used classification methods that turn the probabality distributions into predictions.

4.1 Preprocessing

This subsection describes the applied preprocessing steps for the experiments. The different fields of the blurbs are first concatenated and then tokenized using JTok[2]. The text is normalized by lower-casing, removing all non-alphabetic characters and reducing all multi-spaces to a single white space. As the types of CNNs used in this work require a constant sized input the blurbs are truncated/padded to a fixed length. This sequence length is optimized as a hyperparameter.

4.2 Word embeddings

As mentioned before, fastText (Grave et al., 2018) was used to obtain the word embeddings. In this work the pre-trained German model is used.[3] The number of words, i.e. the top-n most frequent words in the dataset that are embedded, is optimized as a hyperparameter.

4.3 Model

The main focus of this work relies on optimizing CNNs for the task of classifying German text blurbs. The preprocessed text is fed to the network through the earlier described word embedding lookup, which converts word IDs to vectors represented in a high-dimensional vector space.

Afterwards, a convolutional layer is applied on top of the embedding layer. The layer is specified by two central hyperparameters: region sizes and number of filters.

A region size can be understood as the 1-D convolution window size in the domain of computer vison or the n-gram size in the domain of NLP. Each region has a number of filters with different weights. The weights of each filter are adjusted during training time to detect different lexical features in the text.

[2]https://github.com/DFKI-MLT/JTok, accessed July 30, 2019

[3]https://fasttext.cc/docs/en/crawl-vectors.html, accessed August 05, 2019

In order to reduce the output of the different filters at each position in the text, max-over-time pooling is applied. The pooling returns the highest value for each filter with respect to all positions; i.e. after the pooling step, a layer with $|region\ sizes| \cdot number\ of\ filters$ neurons is obtained. These neurons are fully connected to the final sigmoid layer which turns the input into a probability distribution over the label classes.

In order to mitigate overfitting, a Dropout rate is established between these last two layers. This way the network is forced to not solely rely on the activation of single filters.

Besides the CNN's specific hyperparameters, it is also recommended to optimize the learning rate and batch size.

To adjust the weights during learning, Adam optimization (Kingma and Ba, 2015) is used.

4.4 Applied Hyperparameter Optimization

In this section, the results of applying Hyperparameter Optimization by using BOHB (Falkner et al., 2018) are investigated. The hyperparameters are optimized by sampling a random 20% train/test split at the beginning of each run, which is necessary to mitigate overfitting. In total BOHB evaluated 30 iterations to find the final configuration. The search hyperparameter space is shown in Table 1. The final hyperparameters are shown in Table 2. The parameters are very different for both subtasks. It can be seen that the best parameters for the fine-grained classification of `subtask_b` span a more complex network than for `subtask_a` which intuitively makes sense. The sequence length is more than twice as long ($191 \rightarrow 410$), the number of words almost reaches the upper limit of the range ($29,524 \rightarrow 48,736$) and the number of filters per region is also highly increased ($560 \rightarrow 975$). The region size on the other hand drops ($15 \rightarrow 7$), which may indicate that smaller text sequences play a more important role for the fine-grained classification.

4.5 Classification Methods

A special classification method based on the T-Criterion (Boutell et al., 2004) was used to further improve the classification results based on the probability distribution over the label classes. In the standard above threshold classification method, all class probabilities with a value above a pre-set threshold (0.5), are considered as positive classes. There are two problems with this approach.

First, given the Closed World Assumption that every blurb belongs to at least one class, if there is not a single label with a confidence greater than 0.5, there will be no prediction. This problem is solved by using the T-Criterion, if all confidences are lower than the pre-set threshold the label with the highest confidence is assigned if the overall confidence entropy is above a minimal threshold (0.1).

Second, as we flatten the hierarchy structure of `subtask_b` for the training of the CNN, using T-Criterion alone may result in positive classes for a single node in the hierarchy. Therefore, a reconstruction step is applied which additionally predicts the classes in the path from root to predicted node. This classification method will be described as `T-Criterion`.

Table 1: Hyperparameter search ranges.

Parameter	**Range**
Sequence Length	[50; 1,000]
Number of Words	[10,000; 50,000]
Regions Size	[3; 18]
Number of Filters	[200; 2,000]
Dropout Rate	[0.0; 0.7]
Learning Rate	[$1e^{-4}$; $5e^{-2}$]
Batch Size	[16; 256]

Table 2: Best found hyperparameter configurations for both subtasks.

	`subtask`	
Parameter	`a`	`b`
Sequence Length	191	410
Number of Words	29,524	48,736
Regions Size	[15]	[7]
Number of Filters	560	975
Dropout Rate	0.10	0.03
Learning Rate	0.0017	0.0022
Batch Size	32	64

5 Results

Table 3 illustrates the results for both subtasks per approach based on the `dev` and `test` set. As expected, the F1 scores for `subtask_a` (0.850) are higher than the scores for the HMC `subtask_b` (0.664) with 343 classes. The

Approach	dev subtask_a	dev subtask_b	test subtask_a	test subtask_b
SVM	0.783	0.577	-	-
AVERBIS__BOHB_CNN	0.837	0.641	-	-
AVERBIS__BOHB_CNN + T-Criterion	0.850	0.664	**0.834**	**0.644**

Table 3: F1 scores for both subtasks per approach.

AVERBIS__BOHB_CNN outperforms the baseline SVM approach (+0.054). Moreover, applying the T-Criterion classification method further improves AVERBIS__BOHB_CNN (+0.013). Unfortunately, the scores deteriorate comparing dev to test set. This could indicate overfitting on the test + dev set and therefore question the reliability of the found hyperparameters. For more robust results we suggest to tweak the hyperparameter ranges and increase the iteration budget of the optimization.

6 Conclusion

In conclusion we have shown that a strong classifier for German blurbs can be build using a CNN with optimized hyperparameters. Yet, especially the HMC subtask_b is a challenging problem that requires further work. The found hyperparameters for both subtasks illustrate how the complexity of a CNN, i.e. number of trainable parameters, grows from a simple text-classification task with 8 labels to the hierarchical task with 343 labels, given the same input.

As deployment into production is an important factor for us, we plan to compare our results to more recent approaches in terms of accuracy, training and inference time; e.g. Acharya et al. report on decreased model sizes and inference times while maintaining high accuracy by compressing the word embeddings.

References

Anish Acharya, Rahul Goel, Angeliki Metallinou, and Inderjit Dhillon. 2019. Online embedding compression for text classification using low rank matrix factorization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6196–6203.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: robust and efficient hyperparameter optimization at scale. *CoRR*, abs/1807.01774.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2017. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423–444.

Larry M Manevitz and Malik Yousef. 2001. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.