Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Universität Hamburg
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften

Hierarchical writing genre classification with neural networks

im Arbeitsbereich Language Technology
Prof. Dr. Chris Biemann

Bachelor Thesis

Autor: Rami Aly
MatNr. 6796161

Studiengang: Informatik B.Sc.
Version: October 18, 2018

1. Betreuer: Prof. Dr. Chris Biemann
2. Betreuer: Steffen Remus

Abstract

The use of online bookstores has significantly increased in recent years, raising the demand for automatic knowledge organization systems by categorizing books in their respective writing genres. In addition to the difficulty of finding appropriate features for classification, in a realistic task the writing genres are often structured in complex hierarchies. Hitherto, writing genre classification has only been executed sparsely on few and unorganized writing genres. This work introduces two datasets for the English and German language that consist of a large number of blurbs (small advertising texts) and hierarchically structured genres. The potential of using blurbs as a means to classify books is evaluated by applying several neural network architectures to these datasets. We introduce the use of capsule networks to the domain of hierarchical classification. They showed to predict rare label combinations more reliably than traditional convolutional neural networks and recurrent neural networks with long short-term memory units. This is relevant for multi-label classification tasks, as often only a subset of all label combinations are contained in the training data. This especially applies when the classes are structured in a hierarchy, since it naturally causes label co-occurrences. Blurbs showed to be a reliable source of information for writing genre classification with capsule networks displaying promising results.

Contents

List of Figures

List of Tables

Nomenclature

$\boldsymbol{x}_i$	Word vector of i-th feature of an example
$\boldsymbol{f}$	Feature vector of an example
$\boldsymbol{y'}$	Prediction vector for an example
$\boldsymbol{y}$	Labels of an example
$\mathcal{L}$	Set of class labels
F	Collection of examples
f_i	The i-th feature of an example
s	Number of features extracted from an example
T_σ	Activation threshold
X	Word vectors for each feature of an example
y'_i	Prediction for i-th label for an example
y_i	i-th Label of an example
CNN	Convolutional neural network
HMC	Hierarchical multi-label classification
LSTM	Long short-term memory
ML	Machine learning
MLC	Multi-label classification
PRH	Penguin Random House
SVM	Support Vector Machine
TC	Text classification

1. Introduction

The book store sales in the U.S are rapidly declining since 2009[1] while the total number of sales remains relatively constant[2] leading to the conclusion that the popularity of purchasing books online is highly increasing.

The methods to organize books in an online bookstore differs from its traditional counterpart. While the latter requires sophisticated methods like the Dewey Decimal Classification to manage the physical arrangement of books in order to create a useful ordering, an online store only needs to organize books regarding their content (Bhattacharya and Ranganathan, 1971). Thus, the most important aspect of knowledge organization for books in an online store is it's categorization into semantic similar groups. The writing genres books are assigned to are normally ordered in a hierarchical structure. It is also common that multiple writing genres are assigned to to a book.

Therefore, the process of ordering and inserting books in an online bookstore is highly automatable if there is a system that can evaluate and classify the content of a book in a feasible amount of time. However, research specifically done on automated writing-genre classification has been conducted very sporadically. A variety of feature sets have been used and were mostly restricted to single-label classification, ranging from using book-covers and titles (Chiang et al., 2015) to social networks (Holanda et al., 2017). Jordan (2014) uses 100 manually selected summaries and a simple bag-of-words method to classify books into one out of five categories. The drawback of utilizing summaries is the limited availability of labeled datasets. Additionally, these small datasets are mostly build with subjective criteria (Santini, 2007; Jordan, 2014). Hettinger et al. (2015) uses different features of whole German books written between the 16th to 20th century to classify them into two genres (educational and social) with 132 labeled instances. In a recent approach, blurbs have been used in combination with other features in a single-label sentiment classification task using mainly decision trees on a dataset consisting of 236 records and 6 classes (Franzoni et al., 2017). Approaches to classify writing-genres have been relatively limited presumably because it is difficult to find proper features that can be used. To the best of our knowledge, there is currently no promising approach to classify books automatically in a real world scenario.

[1] February 2018: `https://www.statista.com/statistics/197710/annual-book-store-sales-in-the-us-since-1992/`

[2] January 2018: `https://www.npd.com/wps/portal/npd/us/news/press-releases/2018/npd-bookscan-recaps-the-year-in-books-2017/`

This work proposes a solution for classifying books into their respective hierarchically structured writing genres by using advertising texts of books - so called blurbs - as the main source of information. A common blurb as the following:

> "Boys and girls ages 3-7 will love this Little Golden Book retelling Disney's Mulan, available just in time for the release of the Platinum Edition DVD and Blu-Ray in spring 2013! In order to save her father's life, Mulan disguises herself as a soldier and takes his place in the army. Will Mulan bring shame to her family for what she has done? Or will she help save China from the Huns?"[3],

contains useful information about the book's content and subject. Additional to traditional neural network architectures that could use these blurbs, recently, capsule networks have been introduced in text classification by Zhao et al. (2018) showing promising results, especially in multi-label classification because of their transitive property. Therefore, the following questions arise:

- How well are blurbs suited as a source of information to classify books?
- Which neural networks architectures can successfully be explored for this hierarchical classification task?
- How do the novel capsule networks perform on a hierarchical classification task compared to other neural network architectures?

To address these questions, a dataset that consists of blurbs will be introduced in this work for the English and German language. Since many online bookstores list books and their categories with a blurb, a big annotated dataset is buildable without additional human assistance. To the best of our knowledge there are no openly available datasets for writing-genre classification with book blurbs.

We then apply several neural network architectures to this classification task, in order to evaluate the potential of neural networks and blurbs to be used for classifying categories into hierarchically structured writing genres. We further introduce capsule networks to this hierarchical classification task so that the capabilities and the potential of capsule networks for a hierarchical classification task can be explored in this work. There has not been an evaluation of capsule networks on neither a hierarchically structured dataset nor on a dataset with a substantial amount of classes.

Complementary to the introduced datasets, the source code is available on request to enable reproducibility and replicability of results[4].

[3] Penguin Random House, `https://www.penguinrandomhouse.com/books/23859/mulan-disney-princess-by-jose-cardona-illustrated-by-golden-books`, accessed on 15.10.18

[4] Email address: rami.aly@outlook.com

2. Literature Review

2.1. Classification Task

Automated classification of text (Sebastiani, 2002) is a very broad domain of natural language processing. Applied to a variety of different tasks, which are for example document filtering, indexing, dictionaries, automated meta-data generation and word sense disambiguation, it recently received a lot of attention in domains like sentiment analysis, classification of protein functions among many others.

Writing genre classification, belongs to the task of content-based text classification and is hence a form of supervised learning. Clustering methods with the intention to discover semantically similar groups belong to unsupervised approaches and are not further discussed in this work. They do not need to conform with human made categories as the representation can be very abstract and therefore not as useful when ordering books for human access. Content-based classification does only take the content of a document into account but not search-queries or other meta information of users.

Formally, the underlying problem can be formalized by a target function $\phi : F \times \mathcal{L} \rightarrow \{True, False\}$, with F being the collection of blurbs and $\mathcal{L}$ the set of class labels. A classifier tries to find an approximation ϕ_c to the target function ϕ so that they coincide as much as possible. For an example the prediction vector $y' = \phi_c(\boldsymbol{x})$ is calculated based on the input $\boldsymbol{x}$.

The simplest text classification (TC) task is the assignment of exactly one label out of $\mathcal{L}$ to a document, also called single-label classification. The simplest classification problem is $|\mathcal{L}|= 2$, also referred to as a binary classification problem. A task is classified into a multi-class problem if $|\mathcal{L}|> 2$. If $y \subset \mathcal{L}$ labels are assigned to a document, we refer to it as a multi-label classification (MLC) problem. Therefore, while a multi-class problem only assigns one label out of multiple classes, MLC tasks allow to assign any combination of labels to a document. This problem has an exponentially combinatorial difficulty since the latter task has $2^{|\mathcal{L}|}$ possible assignments.

A related task to MLC is ranking. Cast as ranking, the task is to order labels in decreasing amount of applicability; this task could be trained on multi-label data.

In contrast to the above mentioned approaches, this work focuses on the hierarchical multi-label classification (HMC) of writing genres in books. Most online bookstores use a hierarchical categorization scheme for their books. This structure allows to separate subject information based on its specificity . Therefore, even if only more general genres were classifiable with a high confidence, it would still yield relevant information which can certainly be beneficial to a variety applications. For instance,

if new books were added to a library, partially correct information would help and accelerate the annotation process and thus allow better knowledge inferring (Jordan, 2014).

A HMC task consists of labels that are organized in a hierarchy. The structure of a hierarchy ranges from a directed acyclic graph to a collection of tree-components — called *forest*. As specified by Wu et al. (2005) the relationship of a tree is asymmetric, anti-reflexive and transitive. Moreover, a tree assigns only one parent to each child. The root of a tree is the most general label while the leaf consists of most concrete ones. We only focus on tree-structured hierarchies, in particular forests since the class of introduced datasets are structured that way (Section 3). A forest consist of components that are trees. Finally, multiple labels of the same level in a hierarchical structure can be assigned to a book.

2.2. Classification algorithms

Mainly two different approaches for TC have been shown to be useful. Mostly used in the late 1980s, knowledge engineering uses a manually selected list of fixed rules to categorize text (Sebastiani, 2002). It does not need a pre-classified dataset. However, in addition to relying on experts of the domain, this method only works well when categories are well defined and easily distinguishable. Furthermore, it does not scale well since rules have to be manually added, possibly causing inconsistency. Writing-genres are not well separable (Section 2.4) as there is no agreed mutually exclusive definition of each genre, making it very difficult to classify books based on pre-set rules.

Therefore, this work is focusing on machine learning (ML) approaches defined by learning the characteristics of respective categories. In contrast to knowledge engineering, the focus on ML approaches shifts from the construction of a classifier itself to the process of learning a classifier — the so called learner (Sebastiani, 2002). The learner iteratively uses labeled documents as its training set to create the classifier for the given task. The trained classifier is then used to classify unseen documents, the test set.

TC based on Support Vector Machines (SVM) (Cortes and Vapnik, 1995), Linear Regression and Multinomial Naïve Bayes in many variations have become a common technique (Russell and Norvig, 2016). The latter technique and decision trees, especially the C4.5 Algorithm (Quinlan, 1993), find application because the results are more comprehensible in comparison to an SVM. However, SVM's results seem to be generally better as they produce accurate results on a variety of text classification tasks (Russell and Norvig, 2016).

Moreover, the use of ensemble methods (Zhou, 2012) is relatively common in TC tasks as well. An ensemble method uses a multitude of classifiers. Bagging trains multiple classifiers on a subset of the training set in parallel. It uses the average result of each individual classifier on the test set. RandomForest (Breiman, 2001) is a bagging method that uses decision trees and a random selection of features. Boosting, i.e. AdaBoost (Freund and Schapire, 1996), is an ensemble learner that uses weighted data and models to sequentially train each sub-classifier.

The increase of computing power and accessibility of data has led to a growth in interest for neural network architectures in recent times. Although their practical use in TC has been introduced relatively late, most state-of-the-art results on TC tasks were achieved by using these so called *deep learning models* (Goodfellow et al., 2016).

A basic feedforward neural network is a network of nodes with non-linear activations that are organized as layers. Only the first and last layer, the input and output layer are respectively visible. All other layers are hidden. *Convolutional Neural Networks* (CNNs) have mostly been used in image recognition, but recent work shows their effectiveness in TC tasks (Kim, 2014). Character-level CNNs have been used for text classification with competitive results (Zhang et al., 2015).

Recurrent Neural Networks (RNN) connect their output back to the input in order to learn dependencies between words over time. Although the learning process of CNNs are more comprehensive and faster than RNNs, the latter have the advantage that they can cover and detect long-range semantic relationships. The comparative study by Yin et al. (2017) on these architectures with the use of different benchmarks showed that RNNs are especially useful for approaches in tasks that benefit from word dependencies. In contrast, CNNs excelled at comparably short documents and at extracting key words. It showed that in TC tasks like sentiment and relation classification, RNNs, specifically with *Gated Recurrent Unit* (GRU) cells and *Long-Short Term Memory* (LSTM) cells, performed better (Yin et al., 2017).

The introduction of a routing algorithm for *capsule networks* (Sabour et al., 2017), originally introduced by Hinton et al. (2011), showed that shallow capsule networks applied to an image classification task with dynamic routing learn more robust representations for each class as the capture parent-child relationships more accurately. They reached on-par performance with more complex CNN architectures, even outperforming them in several classification tasks, such as the affNIST and MultiMNIST dataset (Sabour et al., 2017). The first dataset consists of digits with small affine transformations in order to test the capabilities of a network to learn robust representations and keep spatial information. The second dataset is a collection of overlapping digits of different classes. This dataset highlights the attention mechanism of the dynamic routing algorithm.

Capsule networks have been applied to text classification by Zhao et al. (2018) showing promising results. He exploits the aforementioned property of capsule networks by showing that capsules outperform traditional neural networks by a great margin, when learning single-labeled and testing multi-labeled documents on the Reuters-21578 dataset. This indicates a greater transitive property because of the attention mechanism and maintained spatial information which helps to keep necessary indicators for multiple classes.

However, the use of capsule networks has been restricted to datasets with only few classes and only on flat classification problems. As hierarchical classes naturally consist of different classes that have overlapping properties, we expect to further benefit from capsules properties. Hence, this work is going to introduce a capsule network model designed for a hierarchically structured dataset with a substantial amount of classes and evaluate its performance with common neural network models for text classification.

2.3. Hierarchical classification

There are several ways to explore the hierarchical structure of a dataset. A HMC task is converted into a MLC problem by exclusively classifying into a subset of $\mathcal{L}$ so that only the most specific genres of a document are retained. That problem is being frequently referred to as *flat classification* problem and makes no use of the underlying hierarchical structure. Although in general less classes have to be classified, it is a 'hit or miss' method as it is not possible to retain information of possibly correct parent classes. That would be unfavorable considering the benefit of also collection partially correct information.

The *local classifier* approach (Silla and Freitas, 2011) trains a classifier either per node, per parent node or per level. During the testing phase, the system uses multiple classifiers, each one narrowing down the number of choices for the consecutive classification. The approach *per node* is the most popular one since most binary classifiers can be used for that purpose. Furthermore, it enables a simple implementation to classify documents so that the most specific node can be at any level of the hierarchy. This is achievable by adding a threshold to the confidence for every local classifier. In case the confidence is below the threshold, the classification is stopped for that document. However, this may lead to either a falsely blocked classification on a very high level of the hierarchy or to classification inconsistencies.

Major disadvantages of local classifiers are that errors in higher levels will propagate down, making it very important that the accuracy of more general classifiers is as high as possible. Moreover, the need to train multiple classifiers independently results in an extension of the training time, making complex models unsuitable for

this approach. In addition, in the worse case, the number of hyperparameters increases exponentially in respect to the level for the common *per node* approach since the classifier for each class needs to be optimized. Hence architectures with many hyperparameters like neural network are especially difficult to use in that approach. Regardless of the disadvantages, neural network architectures have been recently applied to small hierarchies in the local classifier approach as well. Cerri et al. (2014) introduced the use of concatenated mutli-layer perceptrons. It incrementally uses the output of one perceptron as the input for the next level. A hierarchical network architecture called *HDLTex* presents a concatenated technique using either a CNN, a RNN or a deep neural network for each parent node (Kowsari et al., 2017). The HDLTex in particular has only been tested on a hierarchy with a maximum level of 2.

This work focuses on the *global classifier* or *big-bang* approach. Mainly because of the reasons mentioned above, the use of neural networks in the HMC task seems to be most effective using that approach. Moreover, the properties of capsule networks should be beneficial in the implementation of a global classifier. Their use in a local classifier seems currently rather infeasible because of their inherent complexity. A global classifier uses a single classifier that takes the underlying class hierarchy all at once into account. This is why in comparison to the local classifier approach, a global classifier cannot dynamically learn specific parts of the hierarchy which would be helpful asset for extending the hierarchy easily in a real case scenario because it does not necessarily require the training of a new model.

Classifiers that adjust learning algorithms to a specific problem are called *algorithm adaptation* methods (Tsoumakas and Katakis, 2006). However, a very frequent approach is the *transformation* method. The HC task is being cast in a MLC problem that involves every node of the hierarchy. Kiritchenko et al. (2005) casts a HMC problem to a MLC problem by adding every parent label of an assigned label to $\mathcal{L}$ and train the new dataset with Adaboost. The hierarchy is corrected as a post-processing step in order to create consistent classifications that do not contradict the hierarchical structure. Generally speaking, the transformation to a MLC task allows the usage of most techniques that have been introduced to this domain. For example Nam et al. (2017) introduced an improvement in the subset accuracy in a MLC problem by using a sequence-to-sequence prediction algorithm.

However, this work will mainly focus on neural network approaches that make use of the actual relationships between labels since they highly dominate in a HMC task and should be hence explored separately.

2.4. Blurbs and Writing genre

There has been extensive research regarding blurbs and their properties. Blurbs are categorized into the same linguistic genre as advertisements since the main communicative purpose of a blurb is persuasion (Bhatia, 2014). The goal of a blurb is to create a positive response from readers and to make them interested in purchasing the book. Most blurbs consist of at least two to three components (Valor, 2005). The first mandatory part provides information regarding the book's content. It is often combined with subjective evaluations of its content. The second component contains praise mostly written by newspapers and is included in form of extracts. This component only consists of positive reviews and tries to convey that the book has many qualities. A section in which the respective author with information about awards, interests, place of residence and sometimes previous publications can sometimes be found as well. It is crucial to point out that blurbs display information to allure customers. A blurb does not have the objective of actually representing the content of a book correctly as it could try to amuse, misinform or warn the readers to create attention (Valor, 2005).

Common linguistic strategies are being applied to attract the customer. The extensive use of adverbs and superlative constructions is common, especially in the review section. To gain trust,the expressive words are toned down by modal adverbs like *probably*. It is also common to compliment the author and the book, for example by underlining the awards and mostly by using elliptical syntactic patterns. The main purpose is to attract the reader's eye in addition to imitate real speech. These parts could require further knowledge to be understood completely.

Many of the just described characteristics are observable in the blurb in Section 1 and in Listing 1. While both definitely try to allure the reader, the latter also consists of short praise.

It is important to emphasize the difference and distinction between *writing genre* and genre. The definition of the term genre is not well defined in literature (Santini, 2007; Bhatia, 2014), however it normally refers to properties that are complementary to topics, like function, style and text type (Van der Wees et al., 2015). Examples would be *FAQ* or *editorials* for web genres. In contrast to that, the term writing genre is more similar to the concept of *topics*, class or category as the latter describes the subject of a text ranging from broad to detailed (Van der Wees et al., 2015). Since the common term to distinguish topics of book is referred to as *genre* (on both Penguin Random House pages as well), we decided to stick with the term genre for the rest of this work and will use genre classification and writing genre classification interchangeably.

The annotation of books is based on subjective criteria on which most people would intuitively agree. Rule-based approaches that automatically identify genres, as introduced by Santini (2007), are quite inaccurate since there are no agreeable rules on categorizing text into writing genres. The quality of automatically annotated datasets is insufficient, making the use of manually annotated books inevitable. The inconsistency of a genre's definition results in slightly different genre taxonomies and classification schemes across publishing groups. This should be considered when applying already trained classifiers on different datasets.

2.5. Dataset

TC benchmarks range from single-label sentence to large-scale web-pages genre classification tasks. The classic Reuters-21578 collection[5] is seen as the main benchmark in text categorization and is for example used by Zhao et al. (2018). The articles are being categorized into 118 (flat) categories with the possibility of multiple assignment. A newer benchmark is the Reuters-RCV1 dataset (Lewis et al., 2004) consisting of 800,000 documents categorized into several hierarchical category sets. The topic category consists of 103 categories.

A hierarchically structured dataset used by Kowsari et al. (2017) is the Web of Science Dataset: WOS-11967, WOS-46985 and WOS-5736 with 35, 134 and 11 categories and 7,7 and 3 parent categories respectively.

Two annotated datasets with relatively small hierarchies were introduced that consist of labeled documents as well as sentences. Baker et al. (2015) introduced an annotated dataset based on the Hallmarks of Cancer (Hanahan and Weinberg, 2011) with a total of 37 classes. Larsson et al. (2017) introduced a dataset for chemical risk assessment with a hierarchy of 32 classes.

This list is not complete and mostly serves as an overview to different hierarchical datasets for TC.

[5] `https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection`

3. Blurb Datasets

The datasets *BlurbGenreCollection-DE* and *BlurbGenreCollection-EN* are being introduced as part of this work for the German and English language respectively.

3.1. Collection Policy

Both datasets were created with certain requirements in mind in order to create essential properties for our task. It is highly beneficial for a dataset to use a policy that is widely known to make the introduced dataset more accessible and easier to use.

We adopt well-accepted policies from RCV1 why are very frequently used in HMC tasks. All of RCV1's properties have been explained by its authors in detail and should therefore be known to the community.

Both polices that RCV1 introduced are applied to our datasets as well.

1. The Minimum Code Policy requires the assignment of at least one writing-genre to each document of the collection.
2. The Hierarchy Policy ensures that every ancestor of a document's label is assigned as well.

3.2. Collection creation

3.2.1. Collection Process

Both datasets are created with data of the (Penguin) Random House webpage[6].

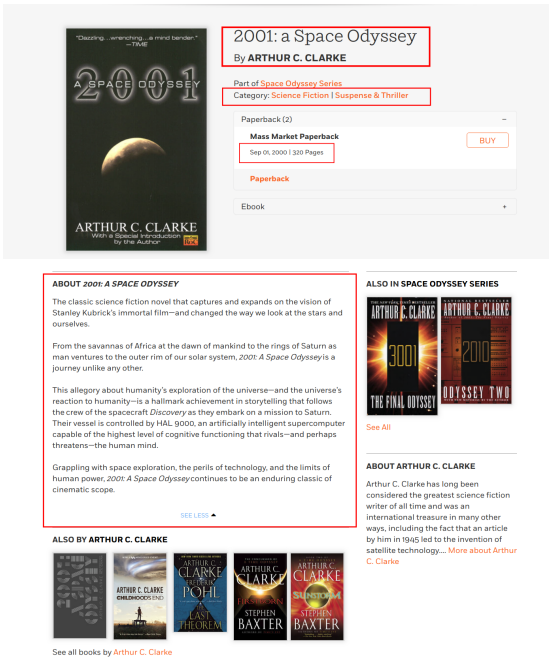
PRH is the biggest publisher group in the world and has therefore an enormous profile of books. PRH owns the rights to its blurbs as they were written by PRH and its content creators. Hence, the publication rights had to be requested. We received permission from the German and American department to publish the datasets that include the blurbs.

The detailed annotation process is not transparent. Most certainly, manual annotations were used in order to categorize the blurbs since assigned writing genres of books should be as accurate as possible to increase the customer's experience. RCV1 for example applied a pipeline of multiple steps. After automatically classifying the

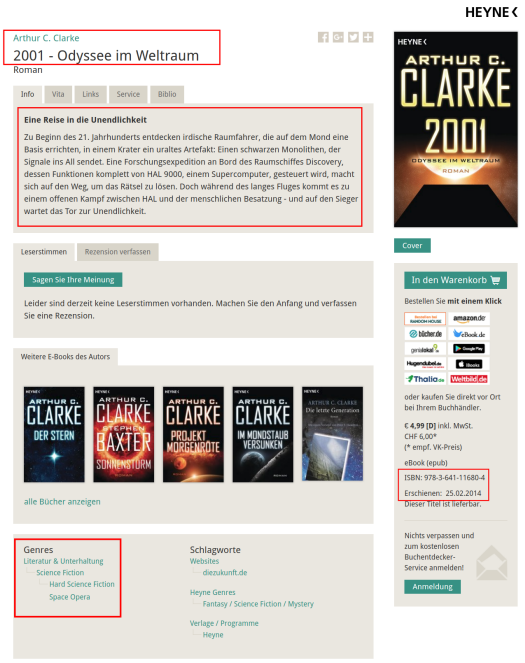
[6]German: `https://www.randomhouse.de/`
US-English `https://www.penguinrandomhouse.com/`

blurbs by using simple rules, human annotators adjust labels in the subsequent process. It seems quite reasonable that a comparable system is being utilized by book publisher groups like PRH.

Since both websites are structured differently, two separate crawling methods were applied to gather and evaluate the data.



(a) English PRH page



(b) German PRH page

Figure 1: Snippets of websites the data was collected from[7]. The specific parts are highlighted in red boxes. Snippets taken in October 2018.

Both pages list every book with its respective blurb. We extracted the *about* and *Info* sections of the English and German dataset respectively as highlighted in Figure 1. These sections frequently consist of praise, hence conforming with the attributes of blurbs as described in Section 2.4. Praise citations from newspaper are listed separately. However, these were not extracted for the dataset.

The genres a book belongs to are also listed. For both datasets we further decided to extract title, author, url, ISBN and date of publication. The date of publication normally refers to the publication date of the specific version of the book. The English dataset additionally includes the number of pages. Other information like *about the author*, *reader rating* and *other books of same author* were not included because special permissions would be required for its usage. Moreover, using this information would go beyond the scope of the purpose this dataset was designed for.

[7]English: Penguin Random House`https://www.penguinrandomhouse.com/books/325356/2001-a-space-odyssey-by-arthur-c-clarke/`, accessed on 14.10.18
German: Random House GmbH `https://www.randomhouse.de/ebook/2001-Odyssee-im-Weltraum/Arthur-C-Clarke/Heyne/e436608.rhd`, accessed on 14.10.18

The crawling of the actual pages was done with a web crawling framework for python called *Scrapy*[8]. XPath and CSS were mainly used to extract necessary information.

BlurbGenreCollection-EN In the English dataset, only the most specific genres are listed, shown in 1a, hence we had to add the ancestors in a post-processing step in order to fulfill the hierarchy policy. Firstly, the books on the page are dynamically loaded via Javscript when the bottom of the page is reached. Therefore, a static extraction of the HTML content was not suitable. To overcome this problem an auto-scroller was applied to export the static, completely loaded HTML content. Secondly, the ancestors of labeled genres had to be added to the labels of a book. We analyzed the structure of writing genres on the American PRH page and manually created a tree to describe the relationships between the genres.

BlurbGenreCollection-DE As seen in Figure 1b, the German PRH page already lists all genres with their ancestors. Although each title by itself has a clear hierarchy, simply combining the extracted relationships would result in ambiguity. This is caused by the assignment of identical genre names to different categories allowing the formation of cycles as well as children to have multiple parents. The genres were systematically renamed by concatenating the parent's and child's genre name in order to create a forest. Despite these steps, noise and redundancy still occurred in the dataset mainly caused by errors on the page itself so that we decided to use a combination of automatic and manual hierarchy extraction to create the taxonomy for the genres. We manually checked all relations and merged or removed some labels to extract a pruned hierarchy.

3.2.2. Post-processing and Pruning

Besides adding all ancestors to the label of a book, further post-processing steps were applied to the collected data for the German and English language. Some categories that are used on the web pages should be neither part of the dataset nor of the TC task as they capture properties that do not rely on content but on the shape or form of a book, which cannot be predicted by a model. *Children's books* for examples were pruned. We kept three out of ten sub-genres since most of them specify the presentation and shape of a book and not the content, for example Picture Book, Boxed Sets, Board Books and so on.

[8] `https://scrapy.org/`

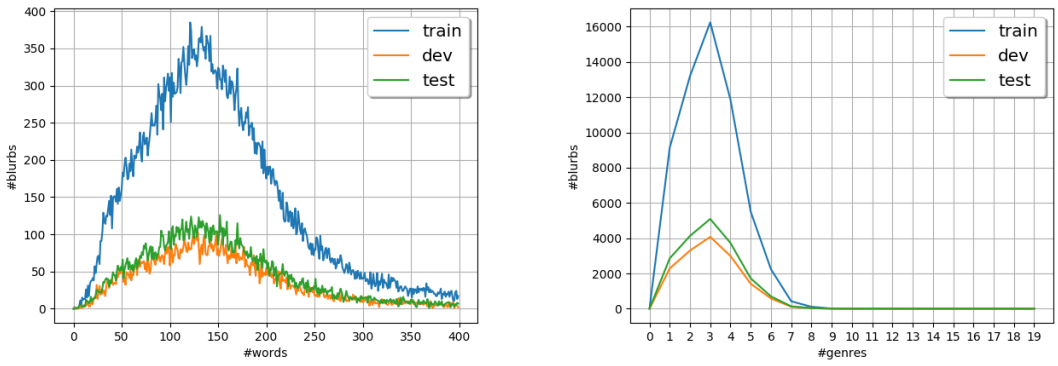
BlurbGenreCollection-EN The genre *Audiobooks* and all its descendants were removed. The genre *Audiobooks* does list many books that are in other book genres as well, therefore referencing again to the type of medium rather than the content's genre.

BlurbGenreCollection-DE In addition to the *Hörbuch Formate*(Audiobook) genre we removed the *Preishits* (special offers) genre since it does not actually classify a book based on its content.

As the last preprocessing step, we remove every book that has assigned genres (combinations) that appear less than five times in the complete dataset. The motivation behind this is relatively straight forward, as it is not feasable to learn classifications on only one or two training samples since at least two samples are needed for evaluation (development and test set). That step removed in total 3552 blurbs from the German and 5943 blurbs form the English dataset, which also indicates that the German dataset has more sparsely distributed genres.

Books which were removed during this process with classes that occur in the dataset are added as an additional asset to the dataset since they can be used to evaluate the transitive properties of a model.

Finally, the preprocessed dataset is divided into three subsets - a training, validation and test set. The validation set is used as a feedback system to evaluate the quality of training. The test set is only used for testing the performance of the final classifier. The total data is split into a training, development and test set in the common ratio of 64%, 16% and 20% (±0.2), respectively. We further applied stratified sampling, to ensure that a random split will not disfigure the distribution. This can also be seen in Figure 2 for the German dataset.



(a) Number of words blurbs have (b) Number of genres blurbs have

Figure 2: Statistics of the German Dataset splits

3.3. Dataset Properties

3.3.1. Documents

The datasets have been analyzed in order to gain insight into their fundamental properties as well as to ensure their correctness. Since every ISBN and URL appears only once, all blurbs should be in theory unique, however in exceptional cases book blurbs can be very similar, for example if a book is part of a series. There are some other anomalies in the documents, e.g. no available author and inaccurate publication dates. Furthermore, it appears that multiple publications of books cause blurbs to be different, although the author and title is identical and vice versa. However, only around 1% of all blurbs are effected by these anomalies. Whether these are problematic depends on the topic of research the dataset is being used for. We are confident that the number of problems is small enough to be ignored. Listing 1 shows an entry of a book and its associated information, extracted from the English webpage that is shown in Figure 1.

```
<book date="2018-08-18" xml:lang="en">
<title>2001: a Space Odyssey</title>
<body>The classic science fiction novel that captures and expands on
    the vision of Stanley Kubrick's immortal film-and changed the way
    we look at the stars and ourselves. From the savannas of Africa at
    the dawn of mankind to the rings of Saturn as man ventures to the
    outer rim of our solar system, 2001: A Space Odyssey is a journey
    unlike any other. This allegory about humanity's exploration of
    the universe-and the universe's reaction to humanity-is a hallmark
    achievement in storytelling that follows the crew of the
    spacecraft Discovery as they embark on a mission to Saturn. Their
    vessel is controlled by HAL 9000, an artificially intelligent
    supercomputer capable of the highest level of cognitive
    functioning that rivals-and perhaps threatens-the human mind.
    Grappling with space exploration, the perils of technology, and
    the limits of human power, 2001: A Space Odyssey continues to be
    an enduring classic of cinematic scope.</body>
<copyright>(c) Penguin Random House</copyright>
<metadata>
<topics>
<d2>Suspense & Thriller</d2><d0>Fiction</d0><d1>Science
    Fiction</d1><d1>Mystery & Suspense</d1>
</topics>
<author>Arthur C. Clarke</author>
<published>Sep 01, 2000 </published>
<page_num> 320 Pages</page_num>
<isbn>9780451457998</isbn>
<url>https://www.penguinrandomhouse.com/books/325356/2001-
  a-space-odyssey-by-arthur-c-clarke/</url>
</metadata>
```

```
</book>
```

Listing 1: Example entry of a book in the dataset BlurbGenreCollection-EN

An extensive comparison of quantitative characteristics is shown in Table 1. The English dataset is significantly larger than the German one, with 91,895 and 17,796 books respectively. Furthermore, a blub in the English dataset has on average nearly twice the length of an average German blurb. The combination of both properties can be seen in Figure 3a, which shows that even the most frequently occurring length of a blurb with approximately 80 words consists of less samples than the English dataset although the maximum is at around 130 words. Despite that, the German dataset has more genres in total while having a more shallow hierarchy at the same time.

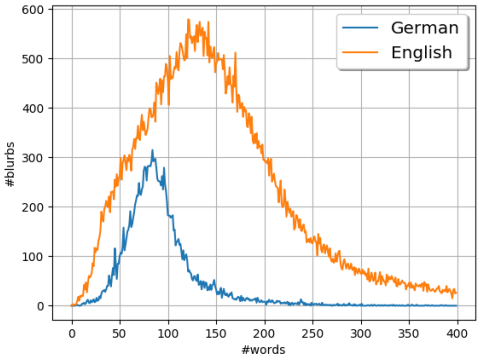
Dataset	BlurbGenreCollection-EN	BlurbGenreCollection-DE
Number of samples	91,892	17,796
Average length of blurb	157.51	92.84
Blurb length 90% confidence	267	137
Total number of classes $\|\mathcal{L}\|$	146	273
Classes $\|\mathcal{L}\|$ on level 1;2;3;4	7; 46; 77; 7	9; 85; 180; 0
Genre Occurrence $\|\mathcal{L}\|$ on level 1;2;3;4	97,218; 113,225; 62,611; 3485	18,064; 20,620; 9635; 0
Average number of genres per book	3.01	2.72
Average number of samples per co-occurrence	68.47	36.79
Occurrence standard deviation σ	389.93	99.28
Average Jaccard similarity (percent)	9.73	5.23

Table 1: Quantitative characteristics of both introduced datasets. The average length of blurbs is measured in number of words. The 90% confidence interval states the length of the blurb in order to be at least as long as 90% of all samples.$|\mathcal{L}|$: h = 1;2;3;4 lists the number of genres on each level h of the hierarchy. The number of samples per co-occurrence lists the average frequency for every genre combination of the dataset. The occurrence deviation shows the divergence of the data from the mean occurrence of samples for each co-occurrence.

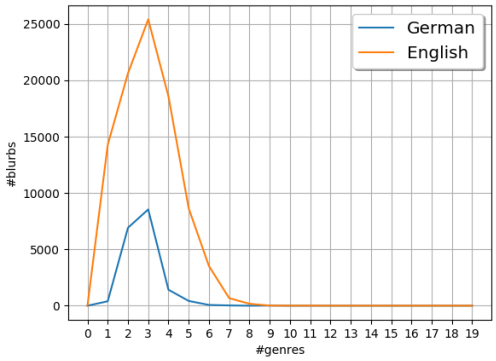
3.3.2. Categories

Both genre hierarchies are structured as forests with each component being a directed tree. Figure 4 shows the subtree for the root *Nonfiction*. It is noticeable that there are only few labels that are on level $h = 3$, although the exponential genre growth seems to be recognizable from level one to three. The full hierarchy for both datasets are visualized in Appendix A.

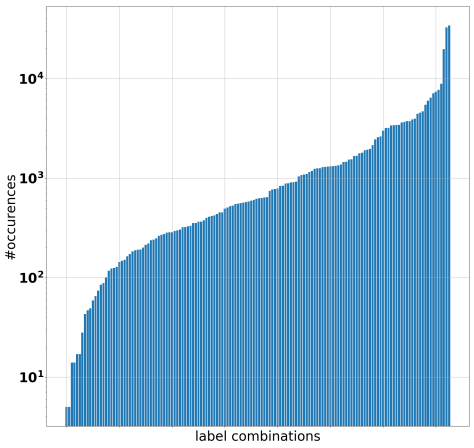
The hierarchy has a total of 146 genres for the English and 273 genres for the German dataset. It is important to note that the most specific genre of a book does not have to be a leaf. For instance, the most specific class of a book could be *Children's Books*, although Children's Book has further children genres, such as *Middle Grade books*.



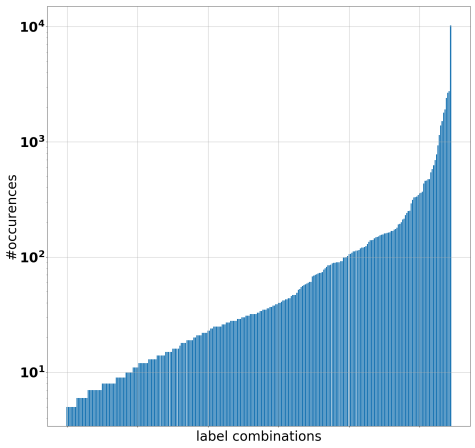
(a) Number of words blurbs have



(b) Number of genres blurbs have



(c) BlurbGenreCollection-EN



(d) BlurbGenreCollection-DE

Figure 3: Property illustrations of both datasets. 3a and 3b respectively compare the length of blurbs and the number of genres for samples. 3c and 3d show genre frequencies in log scale, sorted by size.

This is partly due to the removal of genres as described in Section 3.2.2. However, a great number of books are simply not classified into more special classes on the website.

Furthermore, both datasets are highly unbalanced. The standard deviation σ regarding the mean frequency of label combinations (Table 1) is almost four times as high for the English than for the German dataset. Figure 3 shows the occurrence of each genre combination on a log scale. Ideally, the resulting graph when sorted by occurrence, should increase roughly linearly, since the genres on a lower level in the hierarchy should appear exponentially more often than on higher level, assuming that the graph is not highly unbalanced. However, it is noticeable that while the German dataset shows this property except for the most frequent genres, the English dataset has a distribution in which some labels have disproportionately few or many examples.

To evaluate the correlation between labels, we employed the Jaccard similarity which is defined as $J(Y, \hat{Y}) = \frac{|Y \cap \hat{Y}|}{|Y \cup \hat{Y}|}$ with Y and $\hat{Y}$ being the set of labels of different examples in the dataset. The similarity of labels is almost twice as high in the English dataset.

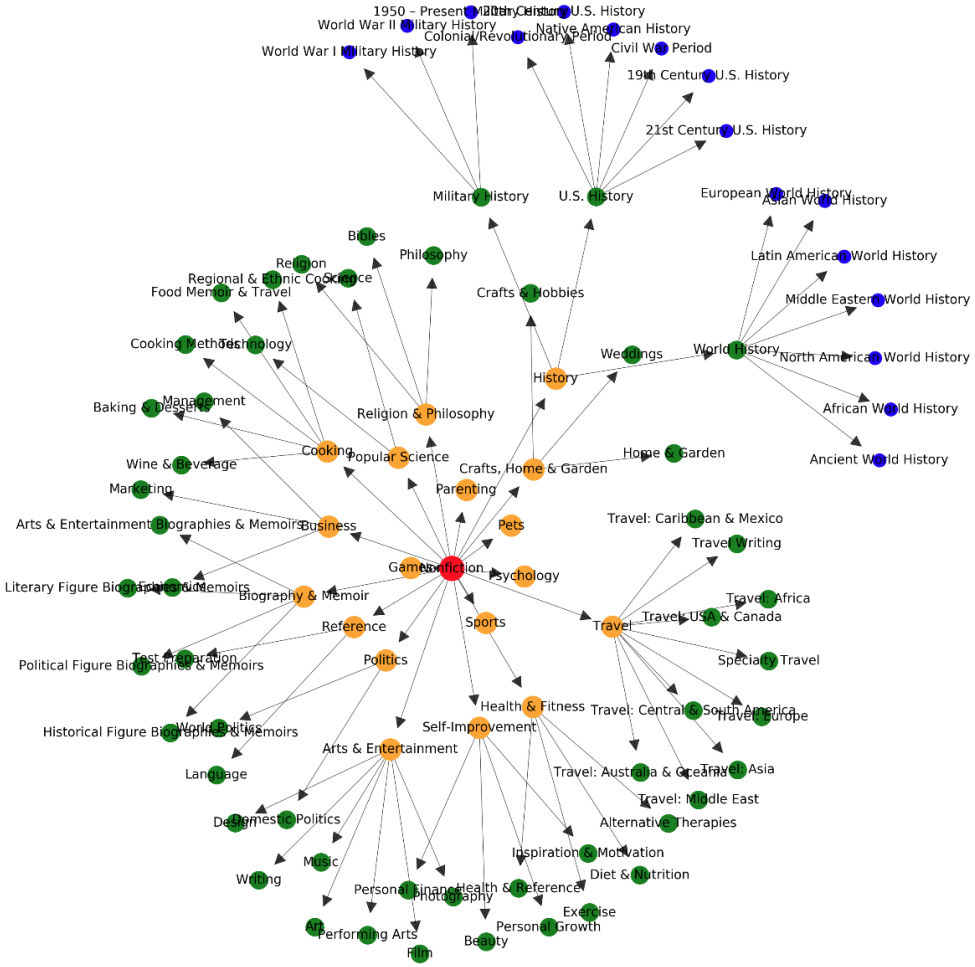


Figure 4: Visualization of a subsection of the English genre hierarchy. Every node on the same level is assigned the same color. The root for this subgraph is *Nonfiction*. The maximum level is four at leafs like *Ancient World History*.

3.4. Further Usage

Further application of the datasets could be a machine translation task as the blurbs for identical books in different languages share many similarities regarding their content. Since the ISBN of a book is created by a combination of information like publisher and region of the specific book publication, it is not trivially possible to create a mapping between the books of both datasets. However, the size of the intersection is estimated by looking at the author and the title. We took the minimum

number of books an author has published in both datasets. This is an upper bound of the actual intersection between both datasets but seemed to be quite accurate on a sample basis. Around 4200 candidates were found in total. Manually checking them should be doable but was not done, as machine translation is not part of this work.

4. Baseline

The baseline method was selected as a traditional, non-hierarchical classifier. To select the most appropriate method we applied a variety of classifiers on a trial dataset that consists of around 2600 documents of the English dataset that were selected randomly. The trained classifier were multinomial Naïve Bayes, logistic regression using l_1 and l_2 penalty, a SVM with and without a linear separator as well as several ensemble methods like bagging, random forest and AdaBoost. Evaluating the methods showed that the SVM with a linear separator yielded the most promising results, hence we decided to use it as a baseline for the actual datasets. This chapter elaborates on the functionality of an SVM and the design we choose to implement.

Support Vector Machine (SVM) First introduced by Cortes and Vapnik (1995) a Support Vector Machine (SVM) is a common and accurate model and has been the default method for classification tasks for many years (Russell and Norvig, 2016).

SVMs belong to the group of *nonparametric* models. *Parametric* models use a fixed set of parameters to summarize data. The number of parameters has to be selected carefully, since too many parameters can result in overfitting while an insufficient number could restrict the function too much. Nonparametric methods, however do not use a set of fixed parameters. They dynamically adjust the number of parameters to the number of documents it uses. Hence they are relatively robust to overfitting. Although the complete training data was used in SVMs, in practice only a small fraction of the documents was retained in SVMs thus combining the benefits of both mentioned approaches.

The standard SVM is used in a binary classification problem. The goal is to find the most appropriate separating line between two classes, i.e. genres of all training examples in the dataset. Every training example is transformed in order to be represented as a feature vector $\hat{\boldsymbol{f}}$ with s being the number of features. For every point $\boldsymbol{z}$ on the hyperplane $\text{sep}(\boldsymbol{z}) = \langle \boldsymbol{w}, \boldsymbol{z} \rangle + b$ is equal to zero, with $\boldsymbol{w}$ being a vector and b the distance to the intersection between $\boldsymbol{w}$ and a hyperplane orthogonal to $\boldsymbol{w}$. The feature vector $\hat{\boldsymbol{f}}$ for each example is assigned to either $y_i \in \{1, -1\}$ for genre i (binary) in separated spaces, depending on $\text{sep}(\hat{\boldsymbol{f}})$ being greater or lower 0. In order to define a separator that can generalize well, the generalization loss $|\boldsymbol{w}| = \sqrt{\sum_{i=1}^{n} (w_i)^2}$ (Euclidean norm) is applied. The minimization of that loss results in a hyperplane that maximizes the minimal distance of all examples to the separator while classifying them correctly. That separator is called maximum margin separator.

Since most problems are not linear separable we add a variable ξ to allow a non-perfect classification while ξ is still being minimized. C is the trade-off constant between minimizing the loss and the correct classification. It is a hyper-parameter and therefore has to be selected before training the model. Let F_{train} be the collection of feature vectors for training and their associated label y. The linear optimization problem with the parameters $\boldsymbol{w}, b$ can be described as:

$$\begin{aligned} &minimize \quad \frac{1}{2}|\boldsymbol{w}|+C\sum_{\hat{f}\in F_{train}}\xi_{\hat{f}} \\ &subject\ to \quad y_i(\langle \boldsymbol{w}, \hat{\boldsymbol{f}}\rangle + b) \geq 1-\xi_{\hat{f}}|\ \forall \hat{\boldsymbol{f}} \in F_{train} \end{aligned} \tag{1}$$

This optimization problem can be solved efficiently. Modern techniques solve the problem in Eq. 1 in its dual representation because all solutions of the dual problem are also solutions for 1. It can be transformed, as w can be represented by a linear combination of the training instances. Let $\hat{\boldsymbol{f}}$ and $\boldsymbol{p}$ be the feature vectors of two different examples of the dataset and $y_{i\hat{\boldsymbol{f}}}$ and y_{ip} the corresponding genre that is being assigned to $\hat{\boldsymbol{f}}$ and $\boldsymbol{p}$ respectively:

$$\boldsymbol{w} = \sum_{\hat{\boldsymbol{f}}\in F_{train}} \alpha_{\hat{f}} y_i \hat{\boldsymbol{f}} \tag{2}$$

$$max_{a_x} \quad \sum_{\hat{\boldsymbol{f}}\in F_{train}} \alpha_{\hat{f}} - \frac{1}{2}\sum_{\hat{\boldsymbol{f}}\in F_{train}}\sum_{\boldsymbol{p}\in F_{train}} \alpha_{\hat{f}}\alpha_p y_{i\hat{f}} y_{ip}\langle \hat{\boldsymbol{f}}, \boldsymbol{p}\rangle \tag{3}$$

The so-called support vectors have a weight $\alpha_{tf} \neq 0$ and are either exactly on the margin or within it, if $\xi \geq 0$, hence their name.
The optimization problem in its dual form (Eq. 3) depends on the similarity, formulated as the pairwise dot product between the feature vectors of two training examples $\langle \hat{\boldsymbol{f}}, \boldsymbol{p}\rangle$.

However, in practice most problems are not linearly separable in the original space. SVMs project to a higher dimensional space in order to tackle this issue. According to Covers' Theorem (Schölkopf and Smola, 2001) the number of possible linear separations then increases. Doing so, almost every problem becomes linear separable in the new space. Every training instance is transformed using a feature function ϕ. Applying the feature function on the dual problem transforms the scalar product to $\langle \phi(\hat{\boldsymbol{f}}), \phi(\boldsymbol{p})\rangle$. However, calculating the product plainly by applying the function to the data results in unnecessary computation steps. A so-called kernel function highly optimizes the calculation and is equivalent $k(\hat{\boldsymbol{f}}, \boldsymbol{p}) = \langle \phi(\hat{\boldsymbol{f}}), \phi(\boldsymbol{p})\rangle$. There are several kernel functions that can be applied to the original problem ranging from linear kernels to kernels with special function properties like $\varphi(\boldsymbol{z}) = \varphi(||\boldsymbol{z}||)$, also called radial bases function (RBF) kernel.

Task Adjustment There are a number of different implementations of SVMs in a multi-label classification task. The approaches can be split into two categories. The first being a transformation method that adjusts the problem in order to use the SVM without essential change. The alternative design is a algorithm adaptation method that extends the actual SVM. To maintain simplicity we decided to transform the problem by using a collection of binary classifiers, so called One-vs-Rest classifier, which can be applied to a multi-label classification problem by using a classifier for each class to make a prediction $y'_i = \phi_c(\hat{\boldsymbol{f}})$. The prediction for all labels of an test example is $\boldsymbol{y'} = \{i|y'_i = 1, i \in \mathcal{L}\}$ with a different SVM employed on each genre i.

5. Neural Networks

5.1. Feature Extraction

Although the dataset offers much more information that could be used for our task in interest, we decided to focus exclusively on classifying the books by its blurb to gain knowledge about the utility of this particular, previously unused kind of data.

Different feature extraction steps were applied to create the input for the neural networks. We try to limit the preprocessing steps of the data to preserve as much information for the neural networks as possible while reducing noise and the vocabulary size. This decision is made, since neural networks tend to benefit from minimal text cleaning[9].

The first step is the tokenization of the complete blurb. This includes the removal of stop-words and most punctuations. We kept special characters like exclamation marks as they can be frequently found in blurbs that have a younger target audience and hence could provide useful information.

We create a vocabulary of words that appeared in the training set and map each word to a unique id. Words in the test set which do not occur in the vocabulary were replaced by an unknown word token. A more sophisticated solution, although not as relevant for this work, would be the use of an additional neural network that learns character representations and creates an embedding for a word even though it did not appear in the data (Zhang et al., 2015). Words that occur only once in the training set are replaced with the unknown word token. The length of a blurb is highly variable, hence we decided to define a maximum length of $s = 100$ tokens which corresponds roughly to the average length of a blurb for the English and 90% confidence for the German dataset as shown in Table 1 (since stop-words are removed). While length limit is necessary for a CNN, it also showed to be effective for the implemented LSTM. Furthermore capsule networks are computationally very expensive, thus limiting the input length has been substantial, in order to explore the proposed capsule network in depth. In case a blurb is shorter than the limit, we padded the input with a special character at the end of the original data.

This preprocessed input vector $\boldsymbol{f}$ is processed by the SVM and the neural network architectures differently. A feature vector $\hat{\boldsymbol{f}}_{\boldsymbol{i}}$ of an SVM is created by calculating frequency-inverse document frequency (tf-idf) of all generated token ids $\boldsymbol{f}$ as inputs. Tf-idf includes the frequency of a word in respect to its appearance in all blurbs. A higher score is achieved if a word is unique in the training data or if a frequency of a word is higher inside one blurb (or both).

[9] `https://groups.google.com/forum/#!msg/word2vec-toolkit/jPfyP6FoB94/tGzZxScO0GsJ`

In comparison to that, the employed neural networks represent a feature f_i by a vector $\boldsymbol{x}_i$, so called *word embeddings* to create a feature matrix X. A vector representation $\boldsymbol{x}_i$ for each word is employed either with randomized weights or pre-learned word embeddings. These weights can be trainable by the network, called *non-static*, otherwise it is called *static* (Kim, 2014). Word embeddings based on the skip-gram model, also called word2vec (Mikolov et al., 2013), are most commonly used in this domain and should yield similar results to alternative embeddings like GloVe (Pennington et al., 2014). Words that do not exist in the vocabulary defined by the pre-computed embedding vectors were initialized with an embedding, that was drawn from a uniform distribution. In addition to approaches like the character embeddings introduced by Zhang et al. (2015), an approach to tackle the issue of missing words in the embedding is addressed by Bojanowski et al. (2017), commonly known as fastText. They present each word as a bag of character n-gram, thus learning subword representations in order to combine them for unknown words. We decided on using embeddings trained with the fastText approach, without the use of subwords. For a detailed explanation see Section 6.2.

The employed neural network infrastructures cannot inherently use hierarchically structured data. To tackle this issue, the labels are transformed into a multi-label classification task as described in Section 2.3. This decision was made with the hierarchical properties of our datasets in mind. As all ancestors are included in a book's label, the multi-label form of the task always contains a more special label with its parents. Therefore, the classifier has the possibility to learn the co-occurrences of labels through the hierarchical structure. Hence, we made use of the benefits that are ensued by the hierarchical dataset in a multi-label classification problem.

The labels are binarized so that the network is able to do a multi-label classification. As a result, there is a unique binary representation for every possible genre combination.

5.2. Convolutional Neural Network (CNN)

The first neural network architecture that is applied on the presented classification task is the convolutional neural network (CNN). They are built with their biological counterpart — the visual cortex in mind and have been successfully employed in image processing tasks (Simonyan and Zisserman, 2014). In recent years they found application in language processing tasks as well, especially in text mining and classification. CNN's require a very large amount of training data in order to be reach an acceptable performance.

Our architecture is based on the model described in Kim (2014). In the first step, the extracted features $\boldsymbol{f}$ of an example with feature length s are mapped on the word representation of each word i, denoted as $\boldsymbol{x}_i$, creating the embedding matrix X. The next step is a convolution operation. $m \in \mathbb{N}$ filters were applied to the vector representation of the features. Each filter $\boldsymbol{w} \in \mathbb{R}^{h \times 300}$ with a window size of h words created a new feature c_i for every possible window of words in the blurb. Therefore, a two-dimensional convolution is used as we simultaneously slide over the complete vector representation at once. Since we slide until the the end of the window reaches the last word, a filter creates a total of $\boldsymbol{c} = [c_1, ..., c_l]$, with $l = (n - h + 1)$ features. Formally, the operation of a filter can be formulated as:

$$c_i = relu(\boldsymbol{w} \cdot (\boldsymbol{x}_i \oplus ... \oplus \boldsymbol{x}_{i+h-1)}) + b), \tag{4}$$

with $relu$ being the rectifier function $relu(c) = max(0, c)$ and b being the bias. Hence, a filter spatially connects the embedding layer to the next layer.

In order to find the relevant features a pooling layer is applied. There have been approaches to create CNNs without pooling layers and rely solely on convolutional layers (Springenberg et al., 2015), however that mostly requires bigger strides (movement or steps of CNN) to limit the amount of parameters the model has to adjust. Otherwise training time and/or training data has to increase drastically (Section 5.4). In the pooling layer only the highest value $\hat{c} = max\{\boldsymbol{c}\}$ of a feature map is maintained — with the intention to keep the most significant feature. Pooling over the sequence of features for different windows is called over-time pooling as the pooling is executed over the time component of the matrix.

A recent study showed that a shallow-and wide CNN that uses word embeddings outperforms a deep CNN (Le et al., 2018). Hence, we decided to use only one convolutional and pooling layer, comparably to the network described by Kim (2014). All outputs of the pooling layer for the respective window sizes are concatenated and flattened, resulting in $\boldsymbol{z} = [\hat{c_1}, ..\hat{c_m}]$.

We then use $\boldsymbol{z}$ for the input of the dropout layer. The dropout layer is one of two regularization methods we applied. During the training phase it *drops* neurons by randomly setting weights to zero. Thus the network only updates a fraction of all neurons during backpropagation. This helps the network to learn more robust features and aims to prevent overfitting (Srivastava et al., 2014). Since previous studies showed that the amount of regularization is maximized at 0.5 (Baldi and Sadowski, 2013), the value is fixed to that in our networks. The l_2-Norm constraint is the second method which will scale the whole weight matrix down back to a fixed value if it is exceeded.

The final layer consists of a fully connected layer that has $|\mathcal{L}|$ output neurons. For the final layer's activation, the sigmoid function (Eq. 5) is used

$$\sigma(z) = \frac{1}{1 + \exp^{-z}} \qquad (5) \qquad\qquad \sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{\mathcal{L}} e^{z_k}}, \qquad (6)$$

instead of the usually used softmax function (6). The latter models the probability that a single label occurs as a multinomial distribution. However, a multi-label classification task requires to make the decision whether a genre is assigned independently. The final label prediction y_i' for genre i will be positive iff $y_i > T_\sigma$ with T_σ being a cut-off threshold. The prediction vector is therefore $\boldsymbol{y'} = \{j | y_i' \geq T_\sigma, j \in \mathcal{L}\}$. In our mode this threshold is a hyperparameter as we can not trivially determine the best threshold. More elegant solutions have been proposed to tackle this issue. Fürnkranz et al. (2008) uses a method called *calibrated label ranking* which adds an artificial label that separates the correct from the wrong labels. Nam et al. (2014) learns a threshold predictor based on the output values of each neuron. The correct prediction would be the one that maximizes the F_1-score. However, in this work we are not going to further address this issue and remain with a default value of 0.5 for all networks as it seemed appropriate during optimization.

The implemented model is illustrated in Figure 5.

The actual learning is done by adjusting all parameters of the model after an iteration of a fixed-sized collection of examples, called mini-batch, is finished. After the results for a batch are retrieved, a loss L is computed. The CNN uses binary cross entropy defined as

$$L(\theta, (\boldsymbol{f}, \boldsymbol{y})) = -\sum_{i=0}^{|\mathcal{L}|} y_i log(y_i') + (1 - y_i) log(1 - y_i') \qquad (7)$$

with θ being the model parameters.

We then use the derivative of the loss in respect to our current parameter to calculate where the loss decreases the most. Gradients for CNNs can be efficiently computed using backpropagation (LeCun et al., 1989).

Since every weight in the filter impacts the feature map, all changes created by the weight will effect the loss. Thus, the filter's derivative for the weights can be described by

$$\partial W_{c_i} = \begin{bmatrix} x_{11}\partial c_1 + ... + x_{l,1}\partial c_l & ... & x_{1,300}\partial c_1 + ... + x_{l,300}\partial c_l \\ ... & ... & ... \\ x_{h,1}\partial c_1 + ... + x_{n,1}\partial c_l & ... & x_{h,300}\partial c_1 + ... + x_{n,300}\partial c_l \end{bmatrix}, \qquad (8)$$

with $\partial c_i := \frac{\partial L}{\partial c_i}$ and $x_{i,j}$ being the j-th value of the word representation of the i-th word.

The derivatives are used in optimization methods which are applied to locate the minimum of L. The most basic method is Stochastic Gradient Decent (SGD) that has mostly been replaced by more sophisticated methods like RMSProp and Adam (Kingma and Ba, 2014; Ruder, 2016). SGD simply adjusts the weights according to a learning rate and the gradient of the loss. It can be extended by using momentum in order to achieve more stable and faster convergence. RMSProp and Adam both extend SGD by adapting the computation of the learning rate. They take the gradients of previous steps into account and scale the learning rate on previous iterations while the latter decay exponentially. Although the learning rate already decays in Adam, we use a further decay that scales the learning rate with respect to the epoch.

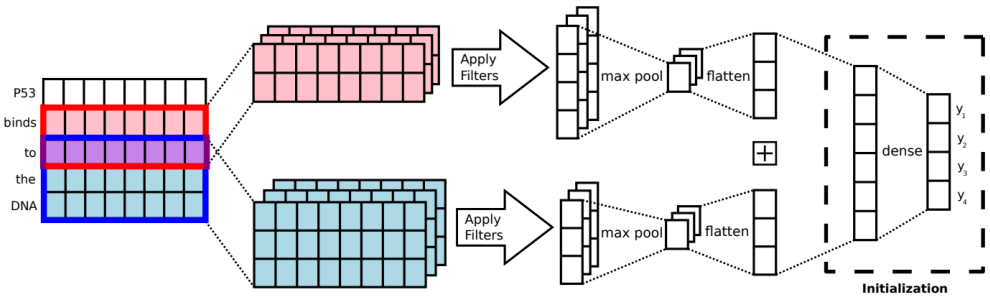


Figure 5: CNN architecture based on the model of Kim (2014), the initialization layer is outlined. Graphic taken from Baker and Korhonen (2017), p.4.

5.3. Long Short-Term Memory (LSTM)

The Long short-term memory (LSTM) is the second neural network architecture that we employed. Historically introduced by Hochreiter and Schmidhuber (1997) it did not find much resonance at its time because a lack of computational resources and data. However, this drastically changed in recent years and major achievements were reached with LSTMs. LSTMs belong to the group of recurrent neural networks and are therefore designed for processing sequential data. Hence, they are especially useful for language processing tasks because of the sequential nature of language. The main advantage of an LSTM over a traditional RNN is its ability to capture long-range dependencies and their ability to deal with the exploding and vanishing gradient problems (Bengio et al., 1994; Kowsari et al., 2017). There are many different variations of RNNs and LSTMs that perform slightly better or worse on certain tasks. We use a traditional LSTM (Kowsari et al., 2017) with minor tweaks.

The output of the embedding layer is given as the input to the actual LSTM. An LSTM consists of n concatenated units. Each unit takes multiple inputs: the vector representation $\boldsymbol{x}_i$ at the position/time step i of the feature set $\boldsymbol{f}$, the output from the previous unit $\boldsymbol{h}_{t-1}$ and the so called memory of the previous unit C_{t-1}. The memory functionality of an LSTM is its most distinguishable property. It generates an output and alters its memory. The memory and output of an LSTM is adjusted from the first to the last unit which then creates the actual output of the LSTM.

There are three gates that control the new memory C_t. First, the input memory C_{t-1} passes the so-called forget gate. It determines how much of the current memory is going to be kept. The gate does an element-wise multiplication, resulting in forgetting most of its memory if the gate's value is near 0. A one layered network with sigmoid activation function learns how much of the memory should be kept. The input for that network is $\boldsymbol{h}_{t-1}, \boldsymbol{x}_i$ and C_{t-1}.

This value is passed onto an input gate which defines which values should be updated. This input gate is implemented as a network with sigmoid activation. Additionally, a layer of neurons with *tanh* activation creates possible updates for the current memory. How much of these updates are added is determined by the input gate. These steps finally create $\boldsymbol{c}_t$.

The actual output $\boldsymbol{h}_t$ is calculated by a layer with the use of $\boldsymbol{c}_t, \boldsymbol{h}_{t-1}, \boldsymbol{x}_i$ and bias b:

$$\boldsymbol{h}_t = \sigma(W[\boldsymbol{h}_{t-1}, \boldsymbol{x}_i] + b) \times tanh(C_t) \tag{9}$$

with W being the weights of the layer (Kowsari et al., 2017). Therefore, in Eq. 9 the just computed new memory is applied to create the output.

Additional to early stopping and learning decay, the implemented LSTM also makes use of the dropout functionality. In contrast to CNNs that implement a dropout before the fully connected layer, it has been shown by Semeniuta et al. (2016) that dropout masks that are implemented between recurrent units provide the highest improvements. Thus we drop a fraction of the recurrent connection h_{t-1}, setting their output to zero. We apply the same mask at every time step for the recurrent layer. The mask does not change during the steps in order to prevent an amplification of noise over long sequences (Gal and Ghahramani, 2016).

Since the output dimensionality of the LSTM is n, a fully connected layer is added that is identical to the CNNs.

The LSTM also uses Eq. 7 as its loss function and computes its gradients by backpropagation.

5.4. Capsule Network

Capsules have been introduced by Hinton et al. (2011) and were refined in Sabour et al. (2017) through the use of dynamic routing. Capsules represent instances or entities and they output a vector with it's orientation representing so-called instantiation parameters like latent structures or as Zhao et al. (2018) assumes, local order and semantic representations of words. Furthermore, while separate logistic units were applied to learn probabilities when capsules were first introduced, in recent applications the length of the output vector defines the probability of an entity's current instantiation. Each capsule learns a part of the input and encapsulates an entity's instantiation parameters, hence it's name. In contrast to that, standard neural networks create a single scalar output for each neuron. It is being argued that an approach with capsules is biologically more accurate since small activities are detected when an entity cannot be found and vice versa (Sabour et al., 2017).

Capsules have shown to be useful in image recognition tasks, which they were introduced for. Exemplary instantiation parameters for that domain are pose or deformation as well as the existence of an entity in the image. An output in the form of a vector has several advantages over the current state of the art model for image recognition, the CNN. Convolutional Layers use filters to extract highly informative features from one certain part of the image and apply that extracted rule to the rest. Therefore, the ability of CNNs to detect positional variances is build in. However, to learn other affine transformations either the number of filters or the amount of training data has to increase exponentially with the number of dimensions (Zhao et al., 2018). This problem is addressed by capsule networks. A capsule makes the assumption that at most one instance of an entity exists at the capsule's location. That allows us to use a distributional representation to encode information of an entity at that location. Furthermore, by applying so called transformation matrices on these vector representations, spatial relationships can be modeled by a simple matrix multiplication. These resulting prediction vectors try to predict the representation of larger objects based on smaller objects. Hence, the representations are equivariant to changes (e.g. viewpoint), meaning while the probability (length) of a representation should remain the same the parameters (direction) should adjust accordingly. An example is shown in Figure 6. The prediction for the CNN is not affected by the modification of the smiley, since every feature that has been found on the original image on the left side is identifiable on the right as well. On the contrary, the capsule's prediction of the modification on the right picture clearly has a smaller length and thus confidence, that the final entity represents a face. This is due to a combination of using vector representations and the superior routing algorithm of the capsule network.

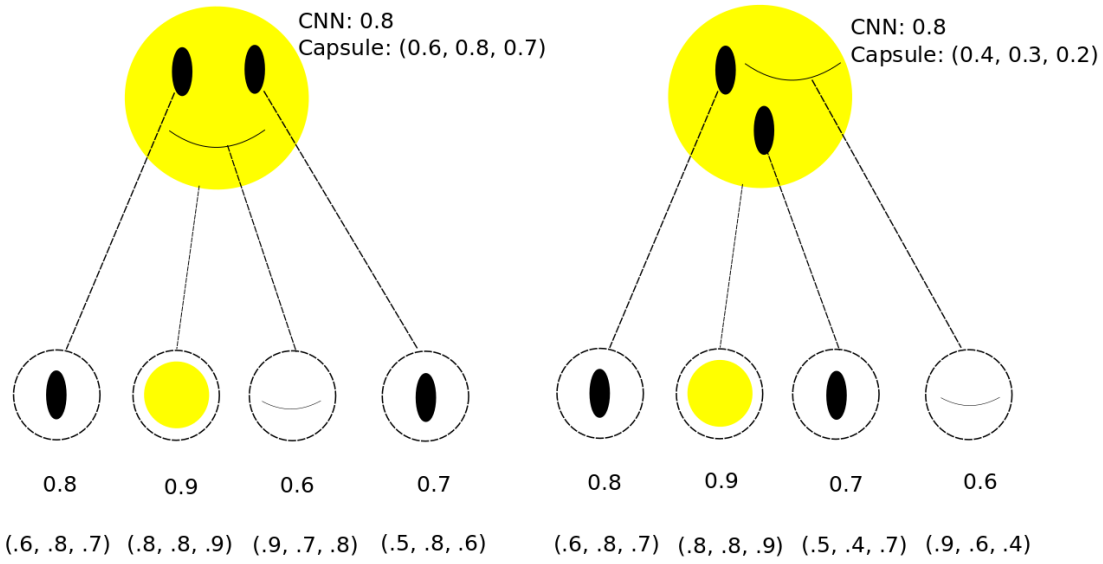


Figure 6: Illustration of CNNs and capsules for image recognition. Because convolutions combine scalars to detect features in pictures and replicate useful features on the whole image, the modification in the second image remains undetected. However, since capsules encapsulate instantiation parameters, information about position and rotation remain. The routing algorithm then uses the prediction vectors of the parts to confirm whether a majority of the entities agree about the whole — hence routing by agreement.

The benefit of keeping instantiation parameters appears to be more clear for image recognition than for TC. That is, because these parameters represent more abstract information like semantic representations in the domain of TC. Furthermore, the dimensional aspect is especially relevant for image recognition tasks, and not so much for TC tasks, since text is mostly one-dimensional and read from left to right.

Nonetheless capsules showed to have properties that could be highly beneficial for our task. Encapsulating terms (n-grams) in different capsules, which do not have the spatial limitations of CNNs, allows us to combine information that is being processed and encoded independently, similar to the image recognition domain.

Our CNN uses max pooling layers as a primitive routing and to reduce the dimensionality of the convolution. By doing so spatial information is lost, as only the most prominent feature is stored while the others are ignored. Therefore, CNNs are relatively restricted in encoding spatially rich data. This is also why the CNN is unable to detect the difference between two faces as both features can still be detected while their surroundings are ignored. This also leads to CNN's having more difficulty in distinguishing overlapping entities or classes.

The routing algorithm of a capsule network tries to recognize an entity and iteratively combines the prediction vectors of capsules of more general entities. This approach is also inspired by biology which assumes that humans recognize objects

by breaking them down into known, much simpler shapes. In image recognition tasks, these properties allow the network to be robust to reflection, rotation, scaling and translation. It is a heuristic which *routes by agreement* (Sabour et al., 2017). All capsules of smaller entities try to predict the output of a capsule that represents a larger object. Prediction of capsules that are more similar to the actual output have a higher impact for the next iteration and thus for the output of the bigger capsule. This method can partially be formulated as the minimization of a clustering loss function with a KL divergence regularization (Wang and Liu, 2018). A simplified illustration is shown in Figure 7.

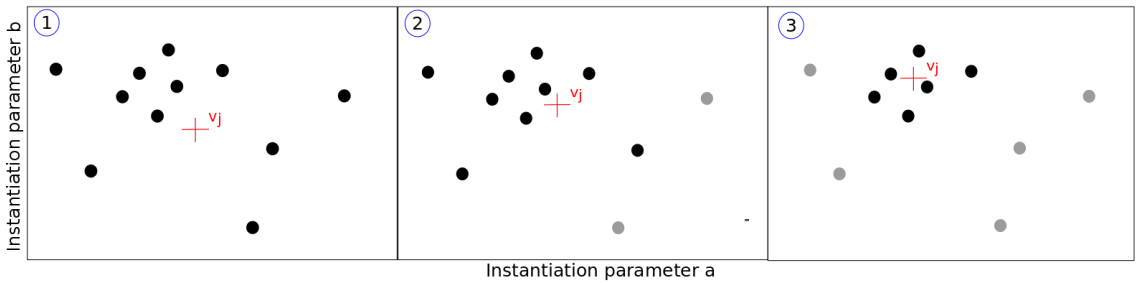


Figure 7: Simplified illustration of the routing algorithm with three iterations. The black circles represent the prediction vectors of capsules. The output of the more general capsule v_j is indicated with a red cross. All capsules create an output vector of dimensionality two. Iteratively, the output of v_j moves toward the cluster that is the closest to it. The relevance of a prediction vector to the output $v_{j|i}$ is updated after each iteration. A prediction vector that has a negligible relevance $c_{j|i}$ is colored in gray.

Let primary capsules be the capsules on the lower level and dense capsules the capsules on the higher level of the network, with $c_{primary}$ and c_{dense} being the respective number of capsules. c_{dense} is equal to the number of classes $|\mathcal{L}|$. Let further $\boldsymbol{u}_i$ be the output vector of the the primary capsule i with dimension $d_{primary}$. As a pre-step for the routing algorithm, u_i is multiplied by a weight matrix $W_{dense|primary}$ to create a prediction vector $\hat{\boldsymbol{u}}_{i|j}$ for every dense capsule j.
$W_{dense|primary} \in R^{c_{dense} \times c_{primary} \times d_{dense} \times d_{primary}}$ is responsible for the equivariance of vectors. Thus, every value of the vector receives its own weight. The prediction vector $\hat{\boldsymbol{u}}_{j|i}$ tries to predict the output that the capsule j has, hence its name. The routing algorithm iteratively updates the relevance $c_{j|i}$ of the prediction vector of capsule i to the capsule j by calculating the similarity $b_{j|i}$ between the prediction and the actual output of j, which is defined by $\boldsymbol{v}_j = squash(\sum_i c_{j|i} \hat{\boldsymbol{u}}_{j|i})$. $Squash(x)$ scales the output non-linearly between zero and one, so that its length can be interpreted as a probability of its instantiation. The similarity is then updated by calculating the scalar product:

$$\boldsymbol{b}_{j|i} \leftarrow b_{j|i} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j \tag{10}$$

The relevance is updated in the next iteration as well, using the softmax to scale the values to probability distribution $c_i \leftarrow \text{softmax}(\boldsymbol{b}_i)$. Since the output of the capsule j is influenced by the weighted prediction vector of i, an increase in the similarity between i and j results in an output of j that is even more similar to i in the next iteration. Therefore, a convergence is guaranteed. The pseudo code of the routing algorithm is shown in (1).

Routing algorithm
Result: $\boldsymbol{v}_j$
Initialization: $\forall i \in Primary. \forall j \in Dense : b_{j|i} \leftarrow 0.$
for *r iterations* **do**
  $\forall i \in Primary : \boldsymbol{c}_i \leftarrow \text{softmax}(\boldsymbol{b}_i)$
  $\forall j \in Dense : \boldsymbol{v}_j \leftarrow \text{squash}(\sum_i c_{j|i} \hat{\boldsymbol{u}}_{j|i})$
  $\forall i \in Primary. \forall j \in Dense : b_{j|i} \leftarrow b_{j|i} + \hat{\boldsymbol{u}}_{j|i} \cdot \boldsymbol{v}_j$
end

Algorithm 1: Routing algorithm after Sabour et al. (2017)

After testing different routing iterations, a common best-practice recommendation is to run three iterations of the routing algorithm.

In image classification, the benefits of the routing algorithm have been shown in a task, that tries to classify highly overlapping digits (around 80%) into mutual exclusive classes. It appears that the routing algorithm acts similar to a big attention network as the existence of every larger entity is is predicted independently, with all spatial information in place (Sabour et al., 2017). This observation has been confirmed by Zhao et al. (2018) for a TC task. They showed that capsule networks can combine information better than CNNs, having significantly higher scores when training on single data information and executing on multi-labeled data, using the Reuters-21578 dataset. Routing does not only take spatial relationships into account but also works like a parallel attention mechanism. That should be highly beneficial in multi-label and especially hierarchical classification, for multiple reasons. Firstly, the task consists of a very high number of classes that can be combined in exponential ways. Secondly, hierarchical tasks often consist of label combinations that are relatively rare, the properties of capsules combined with the attention mechanism could possibly increase the results. The test for overlapping digits also supports the assumption, as genres in an HMC are often not clearly separable, as different genres can share many features.

We experimented with different architectures. The experiments of Xi et al. (2017) and Afshar et al. (2018) favor the use of broad rather than deep capsule networks. Similar properties regarding CNNs were explained above (Section 5.2). Since the complexity of the routing algorithm increases significantly with the number of

classes, we decided to create a network that is simplistic while using the benefits of capsules and dynamic routing, following the kiss (keep it simple and smart) principle. Furthermore, we chose a design to maintain high comparability to the CNN to examine whether capsules and routing perform superior compared to pooling.

The architecture of our capsule network is shown in Figure 8. Similar to CNNs and LSTMs, every feature of a feature set $\boldsymbol{f}$ is mapped on a word vector $\boldsymbol{x}_i$. Then the first layer of capsules is employed, called primary capsule layer. It is a convolutional layer with filters $\boldsymbol{w} \in \mathbb{R}^{h \times 300}$. The filter convolves over the complete vector representation of a word, which results into a feature map of shape $\mathbb{R}^{l \times m}$. The output of the convolution is reshaped into capsules with output vectors $\boldsymbol{u}_i$ for capsule i and dimension $d_{primary}$. To combine the vector outputs of each capsule in the primary capsule layer the routing algorithm described in Sabour et al. (2017) is applied.

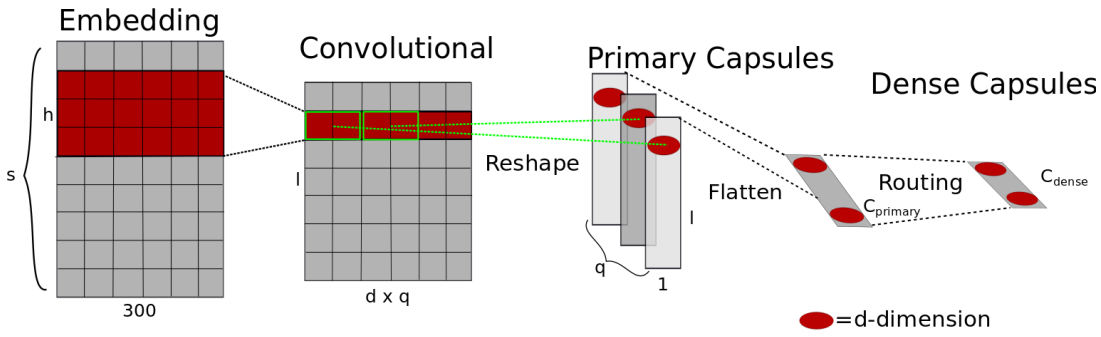


Figure 8: Architecture of the introduced capsule network. The capsules are constructed by dividing the feature map into q capsules, each with a dimension d. Since the convolutional layer convolves over the complete vector representation, reshaping the feature maps to capsules results to a matrix of shape $\mathbb{R}^{l \times 1 \times q \times d}$. These obtained capsules are flattened to $\mathbb{R}^{c_{primary} \times d}$ on which the routing is applied to generate the output of the dense capsules.

The output vectors of the dense-capsule layer of dimension d_{dense} are used to predict the probability of the classes. Since the length of a dense capsule is equal to the probability of a genre's occurrence, we use the length of the capsule's output $y_i = ||\boldsymbol{v}_i||$ for classification.

We further applied the loss function that has been introduced for capsule networks, called *margin loss*:

$$\begin{aligned} L(\theta, (\boldsymbol{f}, \boldsymbol{y})) \; &= \sum_{i=0}^{\mathcal{L}} \mathbb{1}(y_i' = y_i) \; max(0, 0.9 - ||\boldsymbol{v}_i||)^2 \\ &+ \; \lambda \; (1 - \mathbb{1}(y_i' = y_i) \; max(0, ||\boldsymbol{v}_i|| - 0.1)^2, \end{aligned} \tag{11}$$

with $\mathbb{1}$ being the indicator function. $\lambda = 0.5$ reduces the impact of absent classes to the loss, which is helpful at the initial learning as it prevents shrinking of capsule lengths.

Similar to the two aforementioned models, the gradient is computed by backpropagation.

Past applications of capsule networks have been restricted to relative small problems. Capsules are more effective in keeping spatial information than plain convolution without pooling layers (Sabour et al., 2017). However, depending on the number of capsules that are being routed, the capsule network needs a comparably long time to train. For image recognition tasks, the number of parameters should be significantly lower as for a CNN as explained above — thus compensating for the increasing computational time of the network. However, for TC tasks, the complexity benefit of capsules is not as great, since text is mostly one dimensional. Nair et al. (2018) applied capsules to more complex problems. While the applied problems were more challenging, the number of classes did not exceed ten in any dataset. Therefore, this work is the first to introduce an architecture of a capsule network to an extensive number of classes.

5.5. Leveraging Label Co-occurrence

5.5.1. CNN and LSTM

In order to make explicit use of the label hierarchy, we utilize the co-occurrence of hierarchical labels. In contrast to a simple MLC task and as explained in Section 3.1, genre co-occurrences are found particularly often in the HMC task since every book label also includes the ancestor labels. There are several ways to utilize the co-occurrence. Recently, Peng et al. (2018) introduced a recursive regularization method which adjusts the parameters in the final layer by adjusting the loss function to prefer labels that are close in the hierarchy. To utilize the co-occurrences, the final layer with n hidden units is initialized with the weight matrix $W_{uk} \in \mathbb{R}^{n \times |\mathcal{L}|}$ as proposed by Kurata et al. (2016) and applied to the HMC domain by Baker and Korhonen (2017). The initalization layer is outlined in Figure 5. For every co-occurrence a value w is assigned to the associated genres. Every other weight is initialized with 0 for a co-occurrence. w is the normalized initialization value proposed by Glorot and Bengio (2010) as it maintains activation variances and back-propagated variance. The weight is calculated by:

$$w = \frac{\sqrt{6}}{\sqrt{n + |\mathcal{L}|}}. \tag{12}$$

The goal of initializing the layer is to raise the relevance of specific genre combinations that co-occur in the dataset by only triggering the corresponding genres in the output layer.

5.5.2. Capsule Network

Since our capsule network does not have a final fully connected layer, the above mentioned approach cannot be applied. Therefore, this work proposes an adjustment for levering co-occurrences in capsule networks. Since the primary caps are initially weighted by W, we biased the prediction vectors by initializing W with co-occurrences. Since every value of the output vector has a weight to every other value of the output j, the weight can be initialized per capsule or per dimension. We decided on initializing the weight between one dimension of the primary caps and a complete capsule in the dense capsules layer to affect the dynamic routing without initializing too many values to zero, as shown in Figure 9. The initialization value is taken from a uniform distribution while all other values are set to zero. W is not applied as capsules do not use an activation function the way neurons do.

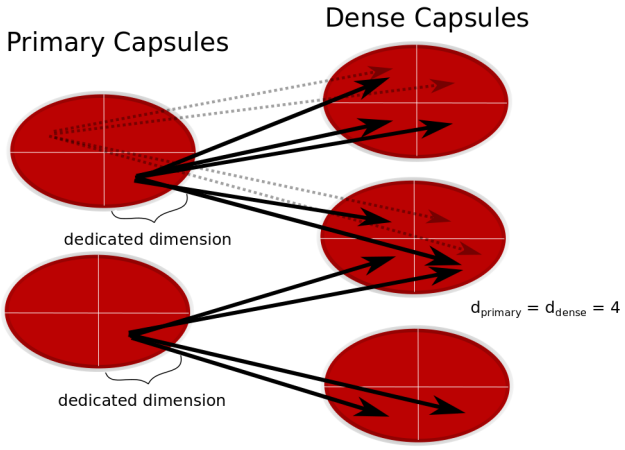


Figure 9: Illustration of weight matrix initialization in our capsule network. Co-occurrence is embedded in capsules by initializing the weight of one dimension of each primary capsule to all dimensions of the dense layer.

5.6. Label Correction

All neural network architectures may assign labels to genres that cannot be reached in the tree as the activation function looks at each genre separately. Inconsistencies in respect to the hierarchy structure of genres are corrected by a postprocessing step. In addition to improving the results, the goal of this step is to evaluate how consistent the predictions are. A score that did not improve much could indicate that the model learned the inherit hierarchy of the data or at least the combinations of genres the hierarchy allows

Let p be a sequence of genres in the hierarchy graph with root r: $p = r, v_1, ..., v_k$. Furthermore, let v_i be a node in p denoted with $v_i \in p$, and $r \leq i \leq k$. We employed three different ways of label correction.

Correction by extension: We add as many labels as needed to make the genres consistent again without removing any genre, hence $y'' = \{v_i | \exists v_j.(v_j \in p \wedge y'(v_j) > T_\sigma \wedge i < j)\}$.

Correction by removal: Adjust labels to $y'' = \{v_i | \forall v_j.(v_j \in p \rightarrow y'(v_j) > T_\sigma)\}$, resulting in the removal of every genre that is not connected to the root through nodes that are part of the prediction.

Correction by threshold: $y'' = \{v_i | \frac{1}{i} \sum_{j=r}^{i} y(v_j) > \psi\}$, therefore the average confidence for an inconsistent prediction y'_i and its ancestors is calculated and kept if the confidence is above a cutoff threshold ψ.

Comparable methods to the first two were used in Baker and Korhonen (2017). The threshold correction method was applied in Kiritchenko et al. (2005).

6. Evaluation

6.1. Experimental Datasets

The HMC task is conducted on blurbs of BlurbGenreCollection-DE and BlurbGenreCollection-EN which were introduced in Section 3. This experiment exclusively uses preprocessed blurbs as features. Despite meta-data such as title and author not being used for this task, this information is occasionally mentioned in a blurb. Thus, there is no perfect separation between the features.

To further evaluate the transitive properties of the different models, we perform a separate test on the books with classes that were below the minimum frequency threshold that has been described in Section 3.2.2.

6.2. Training and Hyperparameters

The training and validation set are used to find optimal values for hyperparameters and learn the model with these parameters in order to do the final evaluation on the test set. We tune the hyperparameters for each neural network architecture on the English dataset and adjust them marginally to the German dataset (e.g. the learning decay, as the German dataset is much smaller). Since the English dataset is relatively large, we decided to use a simple holdout method instead of a k-fold cross validation to optimize the parameters. As the created dataset already provides a validation set, the holdout method is applied to the existing splits.

Applying grid-search to every feature combination of all neural network models is computationally too expensive. Since the average training time for an epoch of our capsule network is on average 3.2 hours, compared to 1.9 minutes for a CNN, we decided against an extensive grid search approach. The optimization is mainly done by starting with the most impactful parameter, on which different values are tried out, while fixating all other parameters. The parameter value that provides the best result is kept as the value for the optimized parameter. This step is repeated for the next hyperparameter — now with the already optimized hyperparameter from the last iteration. Naturally, this greedy approach does not necessarily find the best set of hyperparameters. However, to our estimation it is a valuable trade-off between computation time and performance, since the search space is reduced to a linear complexity. Additionally, the search space for the capsule network is set significantly smaller than the LSTM and CNN ones. Final parameters and their order of optimization is shown in Appendix B.

To increase the confidence in the obtained results we run the LSTM and CNN models three times and report mean and confidence. The capsule network is run

twice. Running models multiple times increases the reliability of presented results. Therefore, the learning process of a network differs despite equal parameters. The confidence is measured by the 95% confidence interval.

During training we also apply early stopping, which stops training if our validation accuracy decreases during multiple epochs. This value is used as a reference for the final run.

The pre-learned embeddings we use are described in Bojanowski et al. (2017) and are openly accessible [10]. We choose these embeddings as they have been published for the English as well as German language giving us comparable embeddings. Furthermore, the embedding vocabulary covers best the training set vocabulary amount of words in the vocabulary are recognized with these embeddings. We are only interested in keyed vectors of words and not in the aforementioned combination by subwords since the use of it decreased the performance of our network. Both datasets use non-static embeddings as they tend to produce better results, as shown by Kim (2014). A detailed summary of embedding statistics can be found in Appendix B4.

6.3. Evaluation Metrics

Because of the problem transformation as described in Section 5.1, all multi-label classification metrics can be used. However, only a few can be adjusted to capture the property of hierarchical classification.

For evaluation, we usew recall R, precision P, and especially the F_1-score. We distinguish between macro and micro averaging. While macro averaging computes the score for each label individually and averages them, the latter computes it for each metric globally over all instances. Labels that occur more often have a higher impact on the micro average score, hence we decided to use the micro average as our main metric. That is mainly due to the hierarchy policy that makes more general classes occur more often as seen in Table 1. It is especially important for the more general classes to be correct since they describe more general attributes of a blurb. Therefore, it seems reasonable that their relevance to the score is implicitly higher than more specialized genres. The micro-average scores can be calculated by:

$$
\begin{aligned}
R_{micro} &= \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_{i,j}Y'_{i,j}}{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_{i,j}} \\
P_{micro} &= \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_{i,j}Y'_{i,j}}{\sum_{j=1}^{k}\sum_{i=1}^{n} Y'_{i,j}} \\
F_{1-micro} &= \frac{2\sum_{j=1}^{k}\sum_{i=1}^{n} Y_{i,j}Y'_{i,j}}{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_{i,j} + \sum_{j=1}^{k}\sum_{i=1}^{n} Y'_{i,j}}
\end{aligned}
\tag{13}
$$

[10] `https://fasttext.cc/docs/en/pretrained-vectors.html`

Silla and Freitas (2011) suggest the use of Eq. 13 for hierarchical classification tasks and even refer to them as hierarchical precision, recall and F_1. The metric is similar to the metrics recommended for MLC tasks by Sorower (2010).

To complement the above mentioned metrics we further employ the so called *subset accuracy*.

$$Accuracy_{Subset} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i' = y_i), \tag{14}$$

with $\mathbb{1}$ being the indicator function. Therefore, a training instance will be evaluated with 1 iff it is exactly the same as the correct prediction. In contrast to the F_1-score, which takes partially correct classifications into account, the subset accuracy is a very harsh metric as there is no distinction between partially correct classification and completely incorrect classifications. Nonetheless, the decision to use this metric was made as it captures how well labels are selected in relation to each other.

There are more metrics that could have been employed like the Hamming Loss, which computes the fraction of labels that are incorrectly predicted, or other proposed evaluation metrics for hierarchical classification tasks. Kowsari et al. (2017) use an accuracy metric which weights correct prediction differently on each level of the hierarchy. Cerri et al. (2014) use 13 with additional weights for each genre depending on its frequency in the dataset. We refrained from using these metrics since they do not provide essential new insights and would mostly cause an overload of metrics.

6.4. Model Implementation and Variations

All models have been implemented in Python. For the neural networks we use specifically the Keras and TensorFlow libraries. The basis of the dynamic routing implementation in Keras is taken from XifengGuo[11]. The evaluation of models and tokenization of blurbs is executed by applying Scikit-learn and SpaCy 2.0.10 with models *en_core_web_sm* and *de_core_news_sm*. The models were trained on an Nvidia 1080 Ti and on an Nvidia Titan X with 10.76GB and 11.75GB memory respectively.

We apply several variants of the proposed architectures:

- CNN, LSTM, Capsule - The basic model as described above. It uses randomized weights on the final layer and does not execute label correction.

[11] `https://github.com/XifengGuo/CapsNet-Keras`

- CNN_{init}, LSTM_{init}, Capsule_{init} - this version does initialize labels as explained in Section 5.5. A total of 1342 and 484 co-occurrence relationships were extracted for the English and German dataset respectively.

We apply label correction (Section 5.6) on the basic models in order to evaluate the effect of different correction methods and the consistency of models.

7. Results

7.1. Results

As explicit hyperparameter optimization was mainly done for the English dataset, the in depth evaluation is mainly performed on the English models. Table 2 and 3 show the results of each implemented model for the proposed English and German experiment, respectively. Identically to Section 6.2, each model is trained multiple times in order to increase the confidence and replicability in shown results. Regarding the English task, the capsule network performed the best out of all models on the main metric F_1. It also yields the highest recall of all models while the CNN achieved the highest precision and the LSTM with initialization showed the best result in subset accuracy. The best performance on the German dataset is achieved by the CNN with initialization. The capsule network performs the second best while the LSTM has again a significant lead on subset accuracy. The LSTM therefore seems to be more capable than convolutional based models to predict classes that are consistent with the hierarchy of the labels.

In both datasets all neural network architectures beat the baseline SVM model by a substantial margin. The initialization of weights lead to a larger improvement on models of the German dataset, albeit the improvement is still small.

The results of employed models are relatively consistent with exception of the LSTM as its confidence interval is relatively high with 2.92 on the accuracy (compare Table 3). That is mainly due to it being rarely stuck in a local minima during the start of the training.

Additionally, we added the results of SVM_{flat} that is trained on only the most special classes of a blurb, thus ignoring the hierarchy of genres. Although a better performance for the classifier that includes the hierarchy in our metrics was expected, the substantial difference in performance is noteworthy.

BlurbGenreCollection-EN				
Model	R	P	F_1	Acc
SVM_{flat}	42.00	80.02	55.08	34.48
SVM	61.11	**85.37**	71.23	35.79
CNN	64.58 ± 0.15	84.00 ± 0.29	73.02 ± 0.08	37.33 ± 0.43
CNN_{init}	64.75 ± 0.41	83.87 ± 0.09	73.08 ± 0.27	37.26 ± 0.52
LSTM	68.36 ± 2.17	75.16 ± 3.61	71.60 ± 1.07	37.66 ± 1.44
LSTM_{init}	69.12 ± 1.24	75.49 ± 3.54	72.16 ± 1.01	$\mathbf{37.99 \pm 1.52}$
Capsule	$\mathbf{71.85 \pm 0.23}$	77.02 ± 0.39	74.35 ± 0.29	37.15 ± 0.41
Capsule_{init}	71.73 ± 0.63	77.21 ± 0.54	$\mathbf{74.37 \pm 0.35}$	37.40 ± 0.68

Table 2: Performance results for the English dataset. All results are expressed in percent with the corresponding confidence interval.

BlurbGenreCollection-DE				
Model	R	P	F_1	Acc
SVM_{flat}	15.66	79.81	26.19	13.90
SVM	46.28	**86.49**	60.29	19.87
CNN	53.52 ± 0.93	83.42 ± 1.73	65.20 ± 0.33	23.72 ± 0.45
CNN_{init}	54.22 ± 1.01	82.23 ± 0.86	$\mathbf{65.35 \pm 0.54}$	24.08 ± 0.48
LSTM	59.02 ± 1.73	68.71 ± 4.91	63.50 ± 1.64	$\mathbf{26.91 \pm 2.92}$
LSTM_{init}	$\mathbf{60.35 \pm 0.97}$	68.05 ± 3.42	63.97 ± 1.99	26.88 ± 1.57
Capsule	56.15 ± 1.51	76.25 ± 1.48	64.68 ± 0.41	22.98 ± 0.92
Capsule_{init}	56.65 ± 1.74	75.98 ± 0.61	64.91 ± 0.59	23.09 ± 1.03

Table 3: Performance results for the German dataset. All results are expressed in percent with the corresponding confidence interval.

The evaluation of the label correction, executed on the results of the different English models, is shown in Table 4. First of all, the highest improvements are achieved with the threshold method. The threshold has not been optimized. We consider it a good starting point as the precision of all models is higher than its recall. Thus, adding correct labels is more rewarding than removing wrong ones. We can also observe greater improvements of the accuracy for the CNN and capsule than for the LSTM. Therefore, not as many corrections are needed as for the CNN and capsule model. In almost every case the F_1-Score increases very slightly as well.

	CNN		LSTM		Capsule	
Method	F_1	Acc	F_1	Acc	F_1	Acc
Base	73.01	37.18	71.66	37.14	74.35	37.15
Extension	73.15	37.68	71.72	37.44	74.34	37.42
Removal	72.83	37.70	71.59	37.43	74.39	37.78
Threshold 0.2	73.16	37.89	71.69	37.50	74.43	37.70

Table 4: Results of Label Correction for the English dataset. All results are expressed in percent. *Base result* is the result of the latest trained model without correction.

Since the dataset is highly unbalanced regarding the number of genres on each level of the hierarchy, an evaluation of the performance on each level of the hierarchy is needed to complement the employed metrics. This evaluation is shown for the English dataset in Figure 10. Although the performance on the highest level is significantly worse, the total score is only slightly decreasing when taking level four into account. Despite a performance decline on each subsequent level of the hierarchy, the capsule network performs best in terms of F_1 of all models on every level of the hierarchy. Furthermore, the score difference is higher in the lower hierarchies. In comparison to the difference of 1.2 points regarding the total result, the capsule network outperforms the CNN on average by 3 points F_1 on levels larger than one. The LSTM has a significant performance collapse on the lowest level.

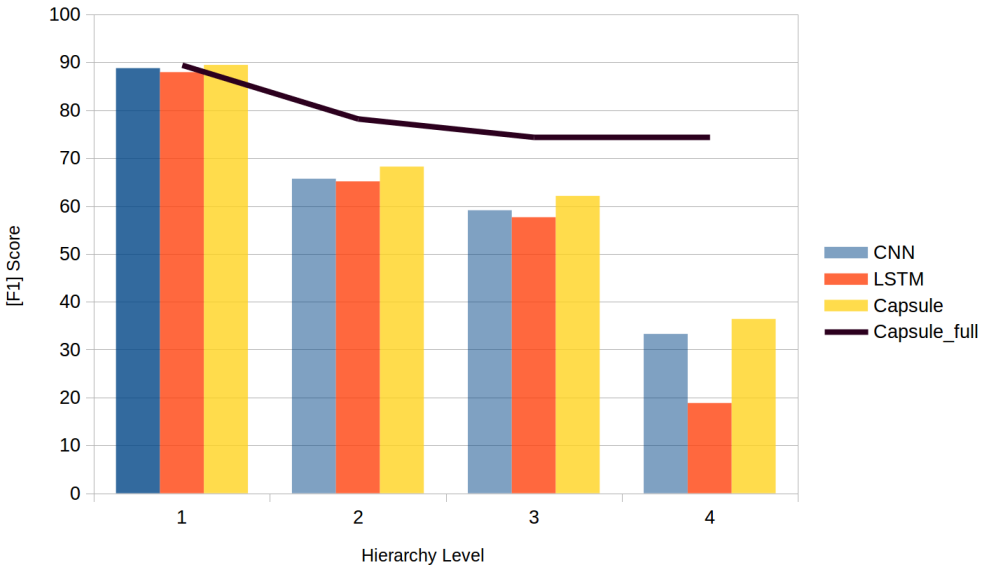


Figure 10: Test F_1-score of classifiers for the English dataset. X-axis shows the result only considering genres that are on the respective levels. The black line is the commutative score, with the value at x = 3 being the result in Table 2.

To check whether data imbalance could have caused the performance decline of the classifiers we further investigate the score for each label co-occurrence sorted by their frequency. As shown by the regression lines in Figure 11 the performance of each classifier increases with the number of examples in the dataset. Although the performance regarding the most frequent co-occurrences is almost the same for all models, the capsule network performs better on low frequency occurrences.

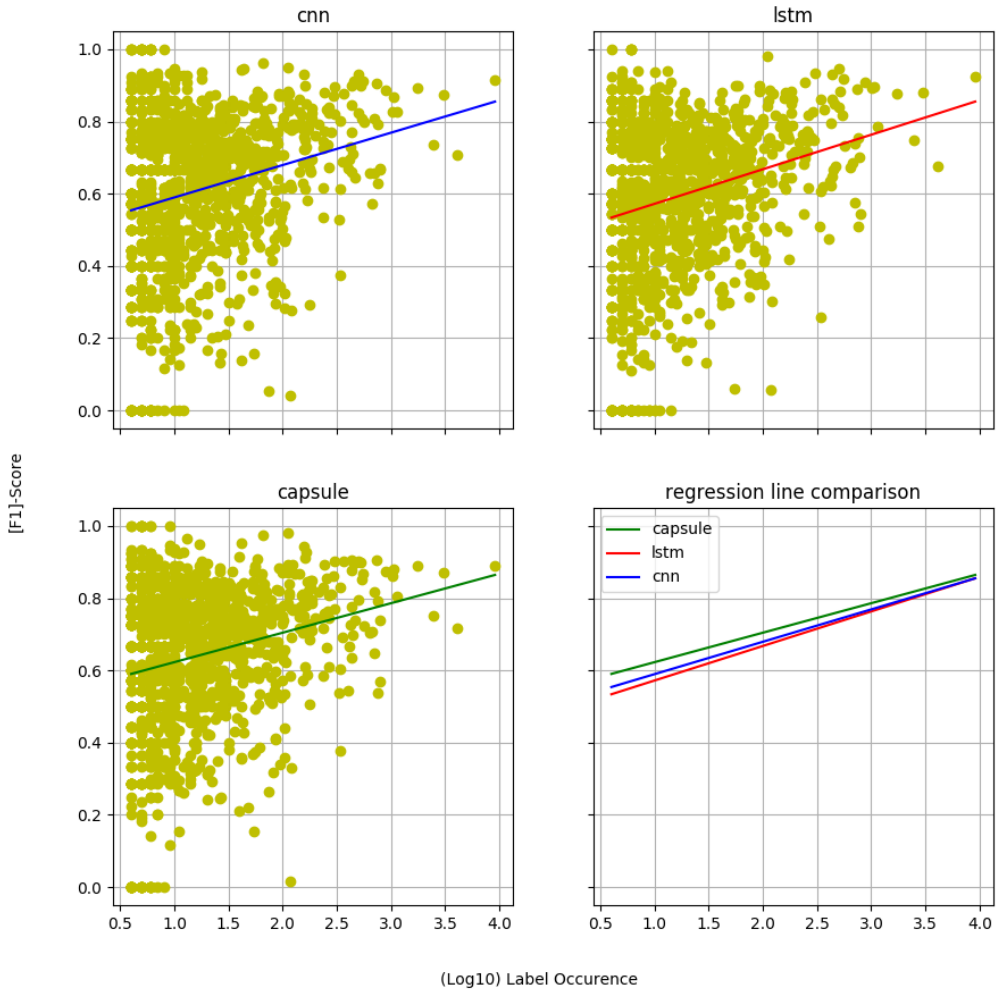


Figure 11: Test F_1-score of classifiers for each label co-occurrence on the English dataset. X-axis is sorted by frequency of genre scaled to $\log_{10}$ scale. The results are fitted to a line in order to emphasize the trend.

Furthermore, it could be highly interesting to explore which label combinations caused the classifier problems and whether some patterns can be explored. Hence, we created and evaluated the confusion matrix for every label combination as there is no standard confusion matrix for a MLC task. The confusion matrix for the most frequent 25 label combinations in the test set is shown in Figure 12. A general observation is that genre combinations which share many labels are more difficult to distinguish by the classifier as three small clusters are already identifiable in the first 25 label combinations. Some false classifications are due to the classifier only classifying a subset of all labels correctly. Most notably are the prediction *Fiction, Literary Fiction* being made frequently wrongly for *Fiction, Literary Fiction, Women's Fiction* and *Fiction, Literary Fiction, Historical Fiction* as well as *Fiction*.

However, if a book consists of only very general labels like *Fiction*, the prediction appears to have a very low accuracy.

Moreover, it is noticeable that predictions of no class at all occur very frequent as well (row with NaN values). The assumption that the classifier would have problems to differentiate between same genres of different age groups does not appear to be true.

The confusion matrices for the other networks and additional results can be found in Appendix C.

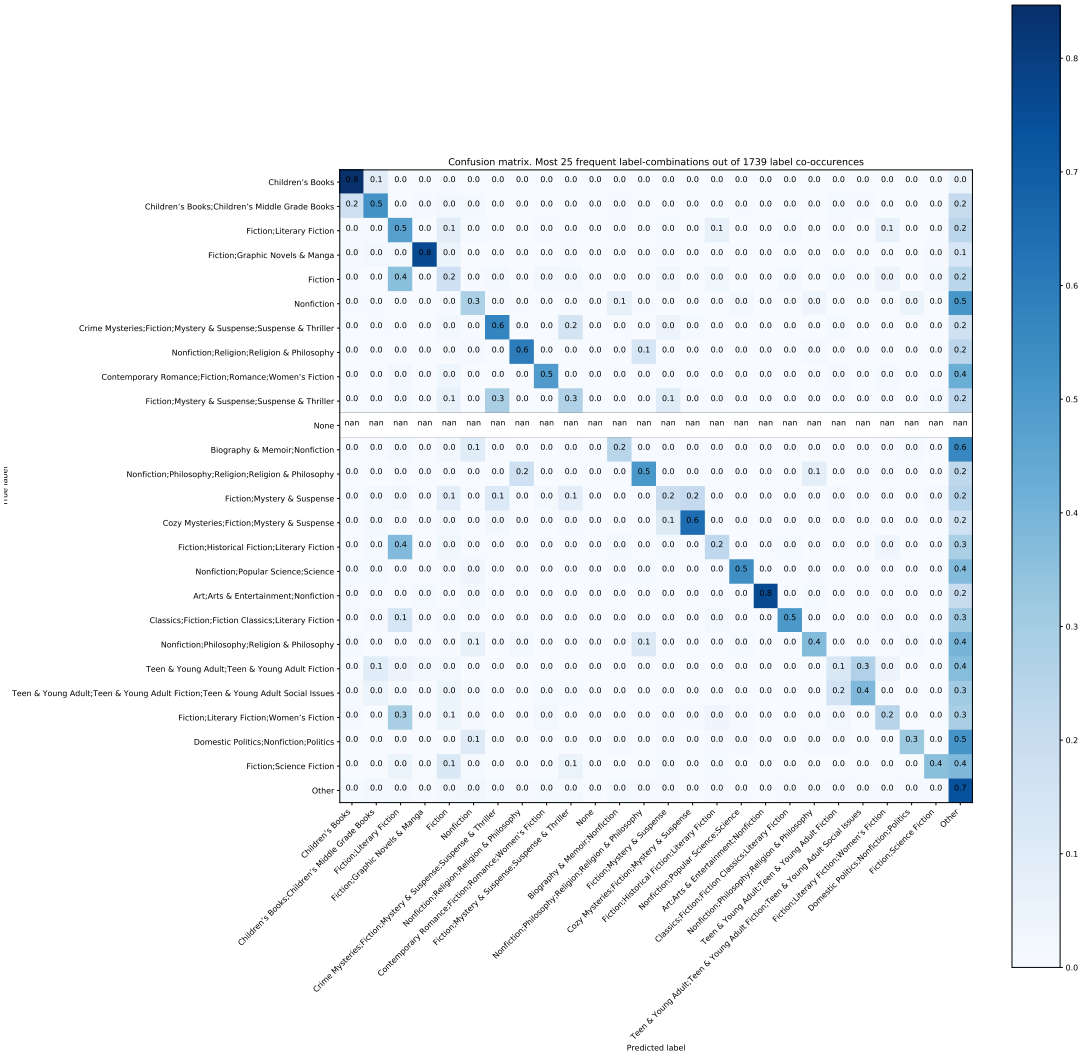


Figure 12: Confusion matrix on the 25 most frequent genre combinations of capsule network's predictions. On the y-Axis are the true labels. The results are normalized. Genre combinations that are not in the top 25 are labeled as *other*. Empty predictions are represented by *None*. Each genre of a combination is separated by a *;*.

Finally, a last experiment was conducted to evaluate the transitive properties of the network, as the network has not been trained on the specific combinations of

genres that are in that dataset. As shown in Table 5, the capsule network has the highest F_1-Score in both datasets. Although the capsule's F_1 in the German dataset is relatively similar to the LSTM's, the precision is approximately 10 points higher. Moreover, it is interesting to notice that the LSTM in the English and the CNN in the German dataset performed surprisingly poor in terms of F_1 because of the higher divergence between its precision and recall score. The subset accuracy is not listed as all networks performed very poorly.

	BlurbGenreCollection-EN			BlurbGenreCollection-DE		
Model	R	P	F_1	R	P	F_1
CNN	46.21	68.95	55.34	29.34	75.47	42.26
LSTM	45.79	60.48	52.13	35.29	60.66	44.63
Capsule	53.30	61.21	**56.98**	33.22	68.70	**44.78**

Table 5: Performance results on test set with low frequency occurrences. All results are expressed in percent.

8. Discussion

8.1. Blurbs as an Information Source

Classification based on blurbs showed promising results. The capsule network classifies into only one of the seven root classes with an accuracy of 84.9 percent. This is significantly higher than previous approaches based on cover page and title that showed an accuracy of 45.4 percent on five classes (Chiang et al., 2015). Even the classification of 46 classes on the second level of the hierarchy was achieved with higher accuracy (51.5 percent). Although the accuracies are not directly comparable to each other, our results still definitely highlight the benefit of using blurbs. We expect that the addition of traditional features, e.g. book titles or date of publication further improve the results. This needs to be investigated in further research.

Additionally, the exploration of the inherent hierarchy of genres improved the results of the classifier significantly in comparison to a flat classifier. The large difference might be due to many intermediate nodes of the tree being the most special label for at least one book. Filtering all but the most special classes for each blurb results to a blurb averaging less than two labels compared to the original 3.01. However, in the flat classification task only seven classes dropped out, so there is a substantial information loss. This highlights again the benefit and necessity of using hierarchical structured genres for certain domains. A comparison regarding the performance of a hierarchical writing genre classification task is not available as all attempts to classify writing genres have been made on flat and few classes.

Furthermore, we tried to understand some causes of classification errors when using blurbs by analyzing error patterns that are conspicuous. For some samples that were assigned to *Fiction, Literary Fiction* instead of *Fiction* by the capsule network, we notice that even for humans it is extremely difficult to separate these label sets. Take for example the blurb

> "Richard Hughes's celebrated short novel is a masterpiece of concentrated narrative. Its dreamlike action begins among the decayed plantation houses and overwhelming natural abundance of late nineteenth-century Jamaica, before moving out onto the high seas, as Hughes tells the story of a group of children thrown upon the mercy of a crew of down-at-the-heel pirates. A tale of seduction and betrayal, of accommodation and manipulation, of weird humor and unforeseen violence, this classic of twentieth-century literature is above all an extraordinary reckoning with the secret reasons and otherworldly realities of childhood."[12]

Literary fiction is mainly defined by having literary merit, which is often correlated to a greater focus on social or political criticism and on in-depth character studies

[12] Penguin Random House, `https://www.penguinrandomhouse.com/books/84118/a-high-wind-in-jamaica-by-richard-hughes/9780940322158/`, accessed on 15.10.18

(Coles, 2008). Characteristics of literary fiction are also the style and complexity of the writing, which is described as being more lyrical and elegant than non-literary books (Saricks, 2009). However, since blurbs normally do not try to imitate the language that is being used inside the book (Section 2.4), it is increasingly difficult to distinguish the genres.

Interestingly, the blurb above is taken from the book *A High Wind in Jamaica* by Richard Hughes and is known for the provocative insight into the psychology of children and thus established itself as a classic in English literature. Therefore, the annotation itself could be slightly incomplete as *Literary Fiction* appears to be a correct genre. This is not an isolated case, as we could repeatedly find cases as the above one for more specialized genres. So it seems possible that the annotation itself could have contributed to worse scores for higher levels.

We also noticed multiple times, that blurbs can be plainly misleading. Take for example the following blurb:

> "A BOSTON GLOBE BEST MYSTERY OF THE YEAR Lisa and Joe Stone, married for twenty years and partners in their small Virginia law firm, handle routine, run-of-the-mill cases, including never-ending complaints from their cantankerous client Lettie VanSandt. When Lettie dies in a suspicious accident and unexpectedly leaves her entire estate to Joe, the Stones find themselves entangled in a corporate conspiracy that will require all their legal skills—not to mention some difficult ethical choices—for them to survive. Complicating matters, Lisa is desperately trying to shield Joe from a dreadful secret, a mistake that she would give anything to erase. With a cast of perfectly drawn imperfect characters, an intricate tour of the legal system, and a remarkably entertaining plot alongside a no-holds-barred portrait of a marriage, The Jezebel Remedy is a legal drama in a class of its own."[13]

It seems obvious that based on the blurb this book belongs to *Fiction, Mystery & Suspense*. That is also exactly what the classifier predicted. However, the actual labels of the book are *Fiction, Literary Fiction, Women's Fiction*. Although blurbs appear to offer highly valuable information about a books genre, it can sometimes be misleading, due to the factors explained in Section 2.4.

Further insights may be achieved by evaluating the annotation quality of human annotators, who are exclusively using blurbs as as source of information. This, however, is out of scope for this work.

8.2. Capsule Networks for HMC Tasks

The first employment of capsule networks on a high-scale hierarchical classification task indicates that the beneficial properties of capsules can be transfered to the do-

[13] Penguin Random House, `https://www.penguinrandomhouse.com/books/240037/the-jezebel-remedy-by-martin-clark/9780804172905/`, accessed on 15.10.18

main of hierarchical text classification. Particularly notable are the increased results of the capsule network on higher hierarchical levels as well as the improved results on low-frequency genres. These improvements are possible due to the routing algorithm which appears to enable better classification results on rare label combinations. This observation aligns with Zhao et al. (2018). Both Sabour et al. (2017) and Zhao et al. (2018) compare the effect of the dynamic routing algorithm with parallel attention mechanisms. A detailed comparison between both approaches might lead to interesting insights.

However, the subset accuracy and prediction consistency of the capsule network is not on par with the LSTM's. The underlying convolutional architecture of the capsule network could contribute to that since the low scores on consistency are achieved for both the capsules and CNN. That conclusion is made based on the significant improvement of the respective scores by using label correction methods, even passing the subset accuracy of the LSTM. To explore more complex parent child relationships with the capsule network the architecture possibly needs to consist of more than just one layer. However, the use of a deeper capsule network with the proposed architecture does currently not seem to be feasible based on the computational cost to train and optimize a capsule network.

Moreover during hyperparameter optimization we noticed several interesting properties. First of all, using primary capsules directly after the embedding layer required far larger window sizes than for CNN's. Since capsules encapsulate information in a vector the richness of the output is far greater than for a single neuron thus using more input information appears reasonable. Secondly, the proposed capsule network is susceptible to overfitting. Sabour et al. (2017) uses a reconstruction loss as a regularization during training. However, this approach does not appear to be applicable in the domain of TC because of multiple reasons like that the reconstruction of the input blurb would result in far more degrees of freedom as every word is represented by a 300 dimensional vector. That is why we used an additional learning decay to function as a regularization method, which increased the results but does not seem optimal as the training for the capsules converged very early with 4 epochs for the English dataset, as seen in Figure 13.

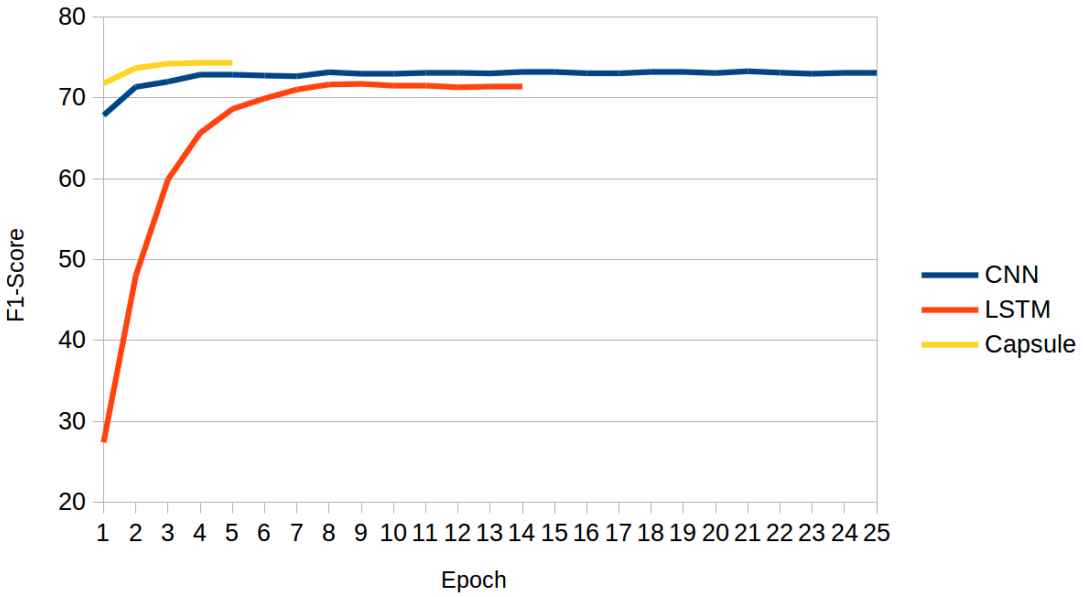


Figure 13: Learning progress for different models for the English dataset. F1-Score is displayed in percent. The Capsule network converges at 4 epochs while CNN and LSTM were run for 30 and 15 respectively.

The results of the German and English dataset show different tendencies. There are several reasons that could explain that. First of all, the lower results on the German dataset may be due to the reduced number of capsules that we had to select in order to run the model (as the number of classes is higher in the German dataset, more capsules in the dense layer were needed). During hyperparameter tuning the performance consistently increased with the number of capsules. However, we were limited in memory and also limited in time as we reduced the search space for hyperparameter tuning and only did the hyperparameter tuning on the English task and minimally adjusted them for the German one. These factors indicate that the performance of the proposed capsule network could possibly be still improved by a substantial amount.

The capsule achieved the best results on both datasets regarding untrained low frequency label combinations. However, the difference was not as substantial as the one shown by Sabour et al. (2017) and Zhao et al. (2018). That could be due to the experiment not testing the transitional properties as cleanly as previous work as many rare combinations are defined by one missing genre. Therefore many genres are still detectable without the presumably useful properties of the capsules.

It is important to discuss the key element of the capsule network, which is the routing algorithm. It was not thoroughly analyzed in the work by Sabour et al. (2017). The behavior of the routing algorithm with an increasing number of capsules has not been studied in detail. It is imaginable that with an increasing number of capsules the routing algorithm decreases in effectiveness which would further explain the

difference in the results between both datasets. However, this hypothesis needs to be explored in a different work. The capsule network does not appear to currently scale as well with an increasing amount of classes. Larger hierarchies than the ones introduced in this work would allow even less primary capsules and thus possibly perform worse. Moreover, most training time of the capsule network is spent with executing the routing. Especially for the hierarchical classification task, the number of capsules in the final layer and therefore the computational cost is significantly higher than for previous applications of capsule networks. This also makes it very difficult to expand the architecture of the capsule network. In contrast, CNNs and LSTMs are easily expandable. The combination CNN and LSTM is found repeatably in literature — that scalability is difficult to achieve with the proposed capsule network. The usage of a capsule network in an ensemble learner, however, might be feasible.

An enhanced version that improves the performance or increases the speed would highly benefit our architecture. Recently, several improvements to the routing algorithm were proposed. Wang and Liu (2018) introduces an alternative routing algorithm based on a clustering-like objective function. Hinton et al. (2018) uses the Expectation-Maximization algorithm so that one cluster per capsule is calculated. No statements about the space and time complexity were made, however, both approaches claim to increase the performance of the capsule network. This seems to be promising as the employment of an enhanced routing algorithm on a small problem is most likely going to improve the results when applied on a much bigger network that heavily depends on it.

As seen in Figure 11, all classifiers seemingly suffer from the genre imbalance in the dataset. There are multiple ways to deal with genre imbalances that were not explored in this work. A common method is oversampling and undersampling. In the study by Buda et al. (2017) oversampling was recommended for CNNs.

Finally, the initialization generally led to improvements of the recall and accuracy. The capsule network and CNN were both only slightly affected by the initialization. There are several reasons that could cause the discrepancy. First and foremost, since only a total of 1342 and 484 co-occurrences were observed, many neurons were not initialized as the CNN has a total of 4500 neurons on the preliminary layer. The capsule has even more with 18360 initializeable units. Furthermore, as shown in Table 1 the Jaccard similarity is almost twice as high on the English dataset. It seems possible that the higher co-occurrence between labels makes it easier for the networks to learn the relationships without initialization hence less benefit attained for the English dataset. We further noticed that training without initialization converges faster, which appears reasonable as less weights can effectively be used to adjust the model when initialization is applied.

9. Conclusion

The use of blurbs as a source of information to classify books into their respective writing genres showed to be highly effective, hence confirming our initial hypothesis. This work introduced two datasets consisting of blurbs for the English and German language. Both datasets do not have optimal properties for a classification task because of the existing hierarchical and label imbalance which, however, accurately represents hierarchical structured information in a subsection of the web. Therefore, both datasets possibly offer valuable information for further research.

We further demonstrated the effectiveness of neural network architectures to the domain of HMC tasks, beating the SVM baseline by a substantial margin. Additional improvements to results of the networks were achieved by the usage of label co-occurrence leveraging and label correction methods. Most notably, capsules indicate promising properties for HMC tasks. Further employment of capsules on different hierarchical datasets could lead to additional insights. Based on this work's observations, the use of capsule network appears reasonable on medium sized hierarchies. Many traditional hierarchical datasets consist of less labels than the introduced datasets. Under these conditions the exploration of the proposed capsule network is encouraged. The potential of capsules in the domain of TC heavily depends on the routing algorithm. As there are currently many improvements coming up, the performance with future and advanced routing algorithms may be investigated.

Due to the convincing results of presented approaches, a real world application for an automatic writing genre classifier could exist, for example in adding new books to an existent library with minimal effort and high accuracy. The effect of using a combination of different information sources like title, blurb and publication data should be investigated in additional research.

Reference List

Afshar, P., Mohammadi, A., and Plataniotis, K. N. (2018). Brain tumor type classification via capsule networks. arXiv:1802.10200.

Baker, S. and Korhonen, A. (2017). Initializing neural networks for hierarchical multi-label text classification. In *Proceedings of the 16th Biomedical Natural Language Processing Workshop*, pages 307–315, Vancouver, Canada.

Baker, S., Silins, I., Guo, Y., Ali, I., Högberg, J., Stenius, U., and Korhonen, A. (2015). Automatic semantic classification of scientific literature according to the hallmarks of cancer. *Bioinformatics*, 32(3):432–440.

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In *Advances in neural information processing systems*, pages 2814–2822, Stateline, NV, USA.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bhatia, V. K. (2014). *Analysing genre: Language use in professional settings*. Routledge.

Bhattacharya, G. and Ranganathan, S. (1971). From knowledge classification to library classification. In *Proceedings of the Ottawa Conference on the Conceptual Basis of the Classification of Knowledge*, pages 119–143, Ottawa, Canada.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(1):135–146.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buda, M., Maki, A., and Mazurowski, M. (2017). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106.

Cerri, R., Barros, R. C., and de Carvalho, A. C. (2014). Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39 – 56.

Chiang, H., Ge, Y., and Wu, C. (2015). Classification of book genres by cover and title. Technical report. Stanford University, CA, USA.

Coles, W. (2008). *Literary Story As an Art Form: A Text for Writers*. AuthorHouse.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Franzoni, V., Poggioni, V., and Zollo, F. (2017). Emotional book classification from book blurbs. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 931–938, Leipzig, Germany.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pages 148–156, Bari, Italy.

Fürnkranz, J., Hüllermeier, E., Mencía, E. L., and Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems 2016*, pages 1019–1027, Barcelona, Spain.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, Sardinia, Italy.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Hanahan, D. and Weinberg, R. (2011). Hallmarks of cancer: The next generation. *Cell*, 144(5):646 – 674.

Hettinger, L., Becker, M., Reger, I., Jannidis, F., and Hotho, A. (2015). Genre classification on german novels. In *Database and Expert Systems Applications*, pages 249–253, Valencia, Spain.

Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51, Espoo, Finland.

Hinton, G. E., Sabour, S., and Frosst, N. (2018). Matrix capsules with em routing. In *International Conference on Learning Representations*, Vancouver, Canada.

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

Holanda, A. J., Matias, M., Ferreira, S. M. S. P., Benevides, G. M. L., and Kinouchi, O. (2017). Character networks and book genre classification. arXiv:1704.08197.

Jordan, E. (2014). Automated genre classifcation in literature. Master's thesis, Kansas State University, Manhattan, KS, USA.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP*, pages 1746–1751, Doha, Qatar.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, Banff, Canada.

Kiritchenko, S., Matwin, S., and Famili, A. F. (2005). Functional annotation of genes using hierarchical text categorization. In *BioLINK SIG: Linking Literature, Information and Knowledge for Biology*, Detroit, MI, USA.

Kowsari, K., Brown, D. E., Heidarysafa, M., Meimandi, K. J., Gerber, M. S., and Barnes, L. E. (2017). Hdltex: Hierarchical deep learning for text classification. In *IEEE International Conference on Machine Learning and Applications*, pages 364–371, Cancun, Mexico.

Kurata, G., Xiang, B., and Zhou, B. (2016). Improved neural network-based multilabel classification with better initialization leveraging label co-occurrence. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 521–526, New Orleans, LA, USA.

Larsson, K., Baker, S., Silins, I., Guo, Y., Stenius, U., Korhonen, A., and Berglund, M. (2017). Text mining for improved exposure assessment. *PloS one*, 12(3).

Le, H. T., Cerisara, C., and Denis, A. (2018). Do convolutional networks need to be deep for text classification? In *AAAI Conference on Artifiical Intelligence*, pages 29 – 36, New Orleans, LA, USA.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, Stateline, NV, USA.

Nair, P. Q., Doshi, R., and Keselj, S. (2018). Pushing the limits of capsule networks. Technical note, `https://pdfs.semanticscholar.org/de7f/27677ae04bf1f09c223f68cea71af90be3d4.pdf?_ga=2.36557536.177846097.1539472311-674172649.1532568691`, Last access: 15.10.18.

Nam, J., Kim, J., Mencía, E. L., Gurevych, I., and Fürnkranz, J. (2014). Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452, Nancy, France.

Nam, J., Loza Mencía, E., Kim, H. J., and Fürnkranz, J. (2017). Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in Neural Information Processing Systems 30*, pages 5413–5423, Long Beach, CA, USA.

Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., Song, Y., and Yang, Q. (2018). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072, Lyon, France.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*. Morgan Kaufmann.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv:1609.04747.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson Education Limited,.

Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30*, pages 3856–3866, Long Beach, CA, USA.

Santini, M. (2007). Automatic genre identification: towards a flexible classification scheme. In *BCS IRSG Symposium: Future Directions in Information Access 2007*, pages 5 –10, Glasgow, United Kingdom.

Saricks, J. G. (2009). *The readers' advisory guide to genre fiction*. American Library Association.

Schölkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

Semeniuta, S., Severyn, A., and Barth, E. (2016). Recurrent dropout without memory loss. In *Conference on Computational Linguistics*, pages 1757 – 1766, Osaka, Japan.

Silla, C. N. and Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations 2015*, San Diego, CA, USA.

Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. *Oregon State University*. Corvallis, OR, USA.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. (2015). Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*, San Diego, CA, USA.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, pages 1929–1958.

Tsoumakas, G. and Katakis, I. (2006). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3).

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., and Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12(Jul):2411–2414.

Valor, M. L. G. (2005). Advertising books: A linguistic analysis of blurbs. In *Ibérica: Revista de la Asociación Europea de Lenguas para Fines Específicos (AELFE)*, pages 41–62, Badajoz, Extremadura, Spain.

Van der Wees, M., Bisazza, A., Weerkamp, W., and Monz, C. (2015). What's in a domain? analyzing genre and topic differences in statistical machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 560–566, Beijing, China.

Wang, D. and Liu, Q. (2018). An optimization view on dynamic routing between capsules. In *International Conference on Learning Representations*, Vancouver, Canada.

Wang, Q., Qiu, J., Zhou, Y., Ruan, T., Gao, D., and Gao, J. (2018). Recurrent capsule network for relations extraction: A practical application to the severity classification of coronary artery disease. arXiv:1807.06718.

Wang, Y., Sun, A., Han, J., Liu, Y., and Zhu, X. (2018). Sentiment analysis by capsules. In *Proceedings of the 2018 World Wide Web Conference*, pages 1165–1174, Lyon, France.

Wu, F., , J., and Honavar, V. (2005). Learning classifiers using hierarchically structured class taxonomies. In *International Symposium on Abstraction, Reformulation, and Approximation*, pages 313–320, Airth Castle, United Kingdom.

Xi, E., Bing, S., and Jin, Y. (2017). Capsule network performance on complex data. arXiv:1712.03480.

Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing. *ArXiv e-prints*. arXiv:1702:01923.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *International Conference on Neural Information Processing Systems*, pages 649–657, Montreal, Canada.

Zhao, W., Ye, J., Yang, M., Lei, Z., Zhang, S., and Zhao, Z. (2018). Investigating capsule networks with dynamic routing for text classification. In *Conference on Empirical Methods in Natural Language*, Brussels, Belgium.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

A. Dataset hierarchy

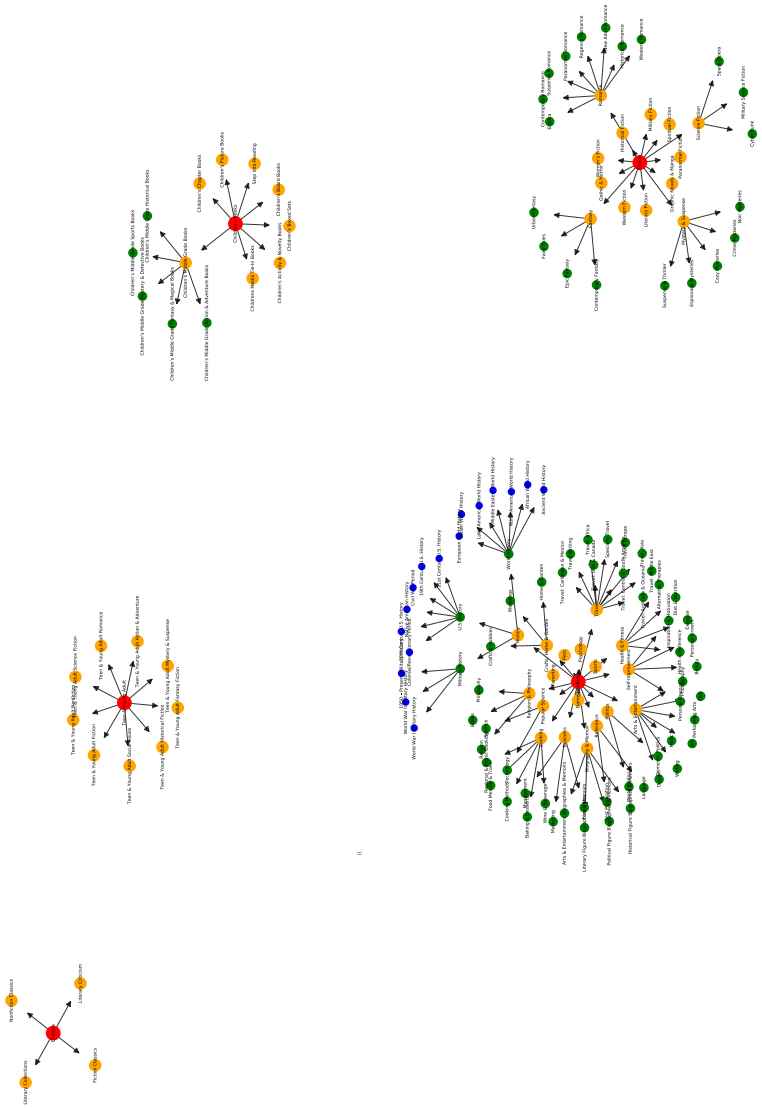


Figure A1: Visualization of the English genre hierarchy. Every node on the same level is assigned the same color. Red nodes are root nodes.

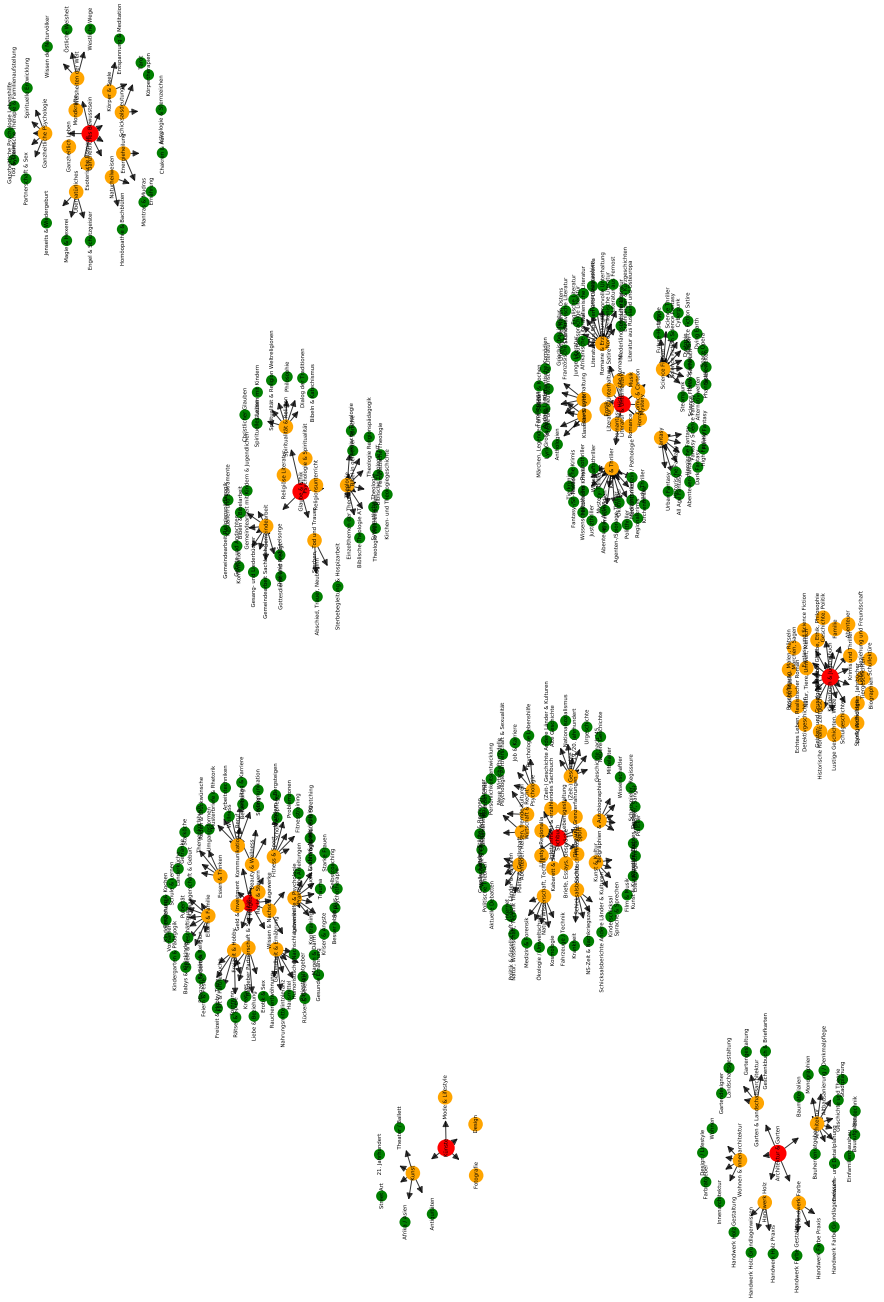


Figure A2: Visualization of the German genre hierarchy. Every node on the same level is assigned the same color. Red nodes are root nodes.

B. Hyperparameters

Parameter	*EN*	*DE*	Tested values
Window sizes	{3,4,5}	{3,4,5}	—
Dropout probability	0.5	0.5	—
Minibatch size	32	32	—
Max. feature length	100	100	—
Epochs	30	30	—
l_2 constraint	3	3	—
Optimizer	Adam	Adam	—
Embedding	non-static	non-static	—
Number of Filters	1500	1500	{800, 1000, 1500}
Learning rate	0.0005	0.0005	{0.001, 0.0005, 0.0001}
Learning decay	0.9	1.0	{1, 0.9}

Table B1: Hyperparameter CNN, sorted in the order in which parameters were optimized.

Parameter	*EN*	*DE*	Tested values
Recurr. Dropout probability	0.5	0.5	—
Minibatch size	32	32	—
Max. feature length	100	100	—
Epochs	15	25	—
Optimizer	Adam	RMSprop	—
Embedding	non-static	non-static	—
Number of units	1500	1500	{700, 1000, 1500}
Learning rate	0.0005	0.001	{ 0.001, 0.0005, 0.0001}
Learning decay	1.0	1.0	{1, 0.9}

Table B2: Hyperparameter LSTM, sorted in the order in which parameters were optimized.

Parameter	*EN*	*DE*	Tested values
Primary Capsule dim	8	8	—
Dense Capsule dim	16	16	—
Minibatch size	32	32	—
Max. feature length	100	100	—
Optimizer	Adam	Adam	—
Embedding	non-static	non-static	—
Number of Capsules	55	45	{16, 32, 55}
Window sizes	{50}	{50}	{3, 5, 30, 50}
Learning rate	0.001	0.001	{0.001, 0.0005}
Learning decay	0.4	0.92	{0.2, 0,4, 0.6, 1}

Table B3: Hyperparameter capsule network, sorted in the order in which parameters were optimized.

Parameter	*DE*	*EN*
Embedding vocabulary size	2,275,233	2,519,370
Dataset vocabulary size	41,140	105,360
Present in Embedding	38,527 (93.6%)	77,187 (73.3%)

Table B4: Fasttext embedding properties for the English and German models without usage of subwords

C. Additional results

Model	CNN	CNN_{init}	LSTM	$LSTM_{init}$	Capsule	$Capsule_{init}$
English F_1 Macro	46.31	46.31	42.63	50.04	46.82	46.38
German F_1 Macro	21.34	20.56	23.67	24.61	18.66	19.34

Table C1: F_1 macro scores of most recent trained neural network models. Results are expressed in percent.

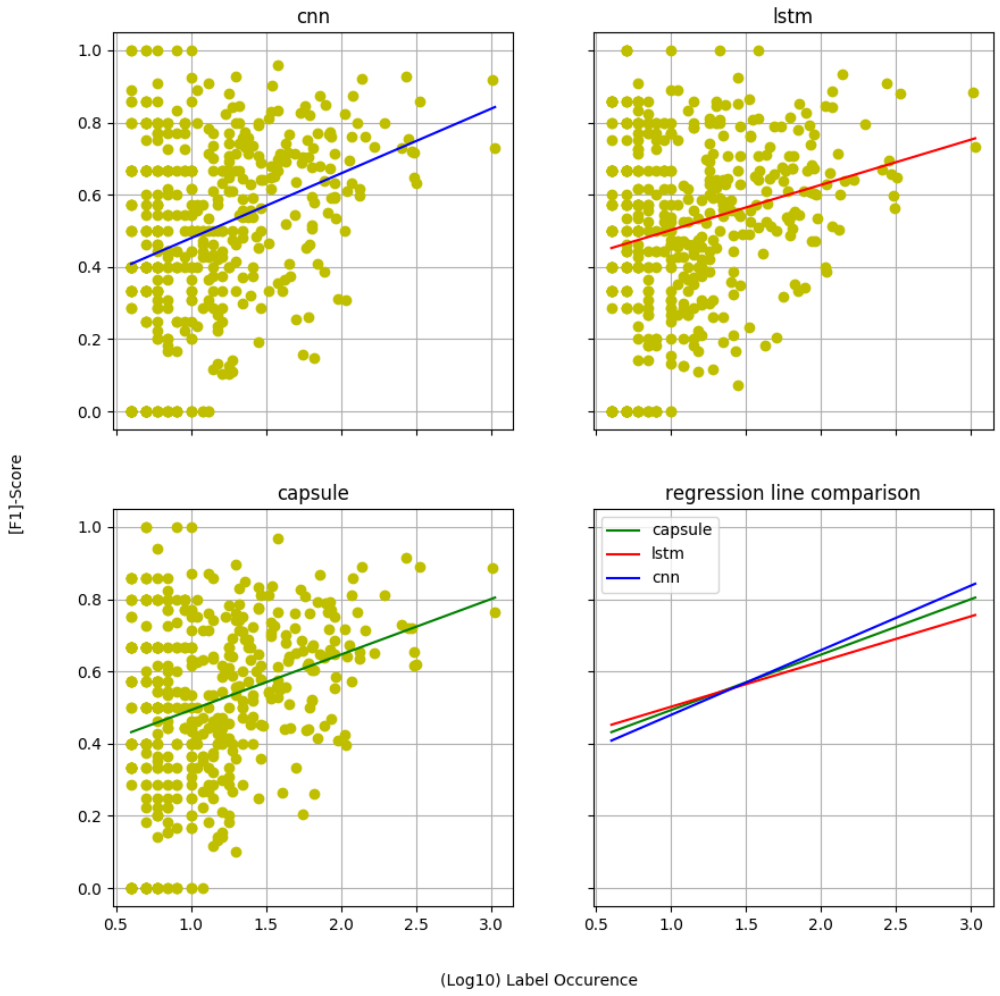


Figure C1: Test set F_1-score of classifiers on the German dataset. X-axis is sort by frequency of genre scaled to $\log_{10}$ scale while the y-axis shows the respective result. The results are fitted to a line in order to emphasize the trend.

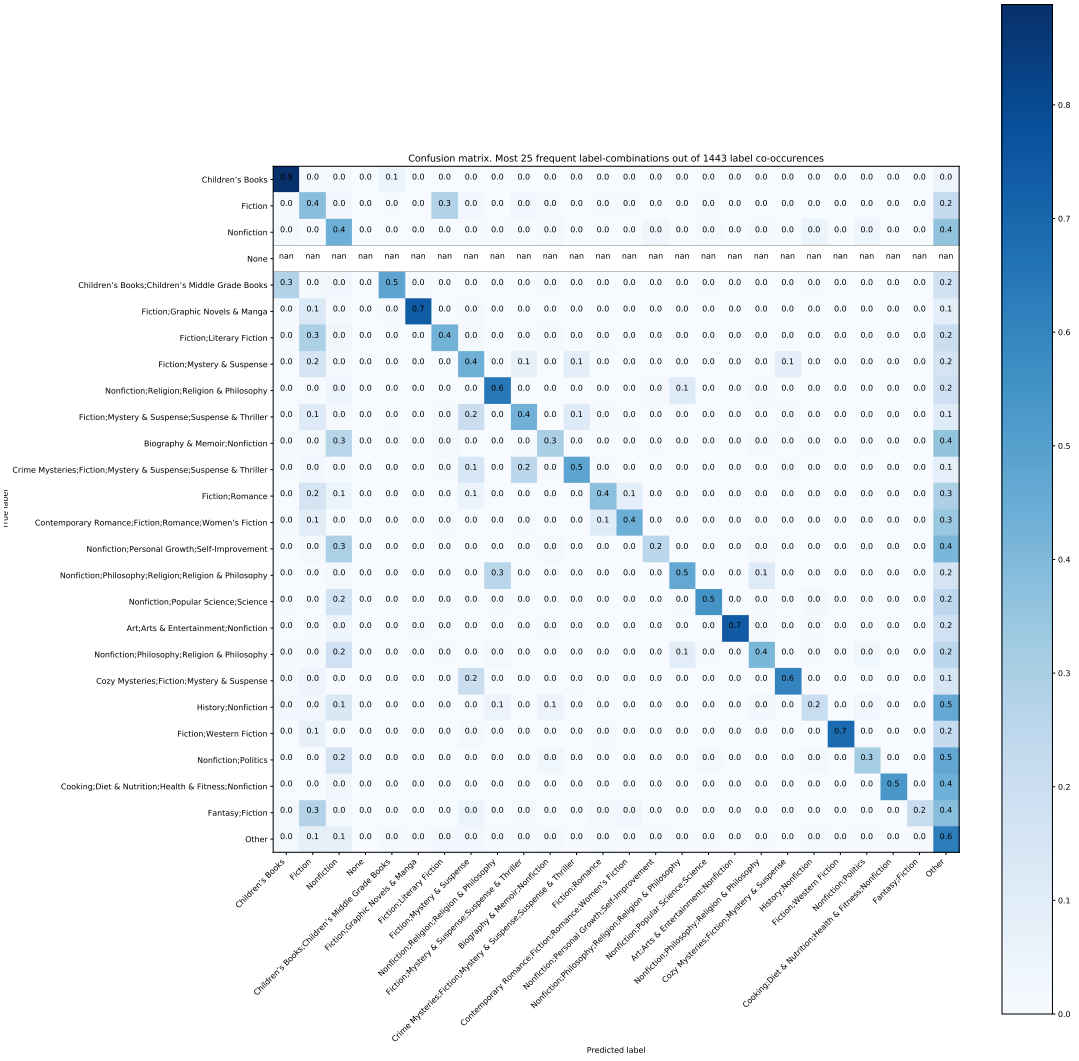


Figure C2: Confusion matrix on the 25 most frequent genre combinations of CNN's predictions. On the y-axis are the true labels. The results are normalized. Genre combinations that are not in the top 25 are labeled as *other*. Empty predictions are represented by *None*. Each genre of a combination is separated by a *;*.

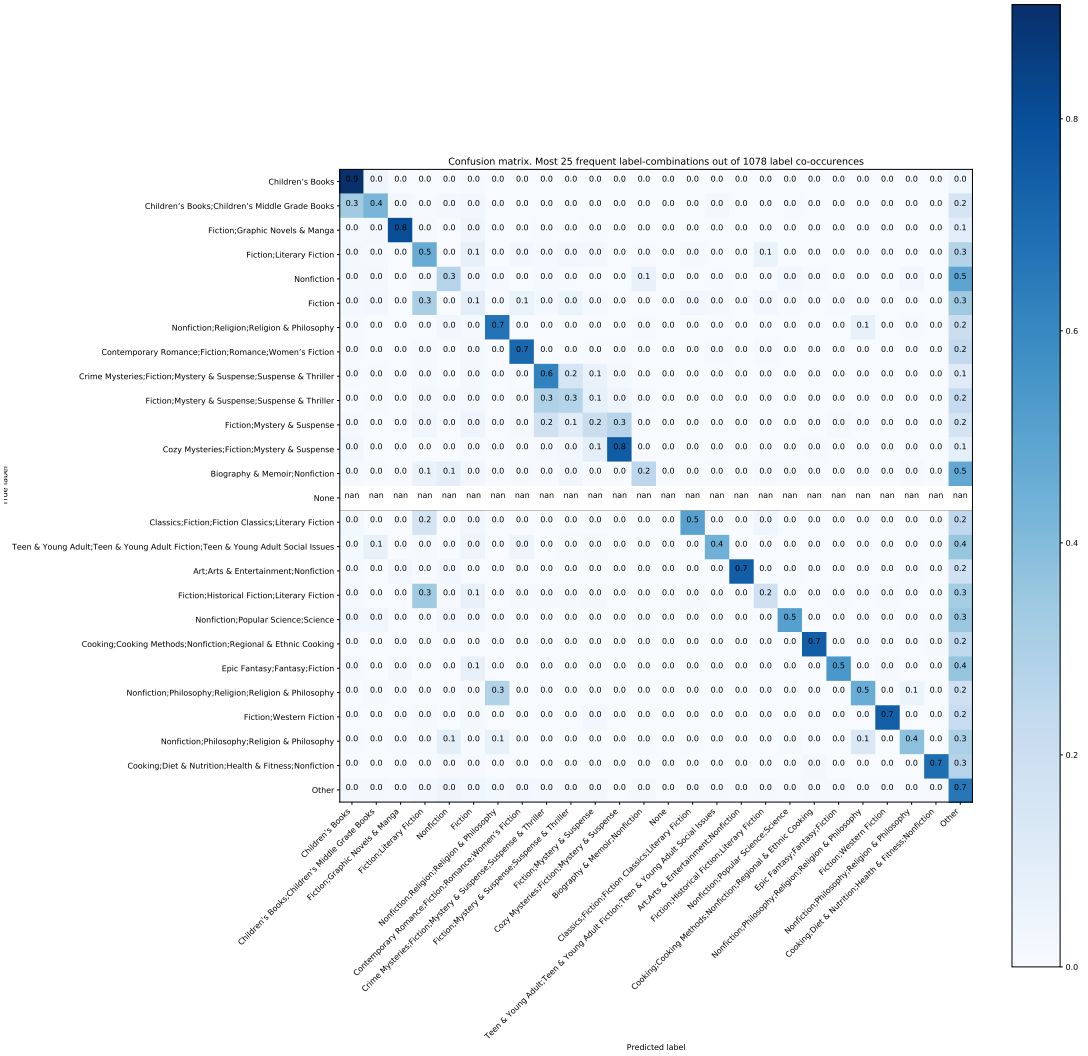


Figure C3: Confusion matrix on the 25 most frequent genre combinations of LSTM's predictions. On the y-axis are the true labels. The results are normalized. Genre combinations that are not in the top 25 are labeled as *other*. Empty predictions are represented by *None*. Each genre of a combination is separated by a *;*.

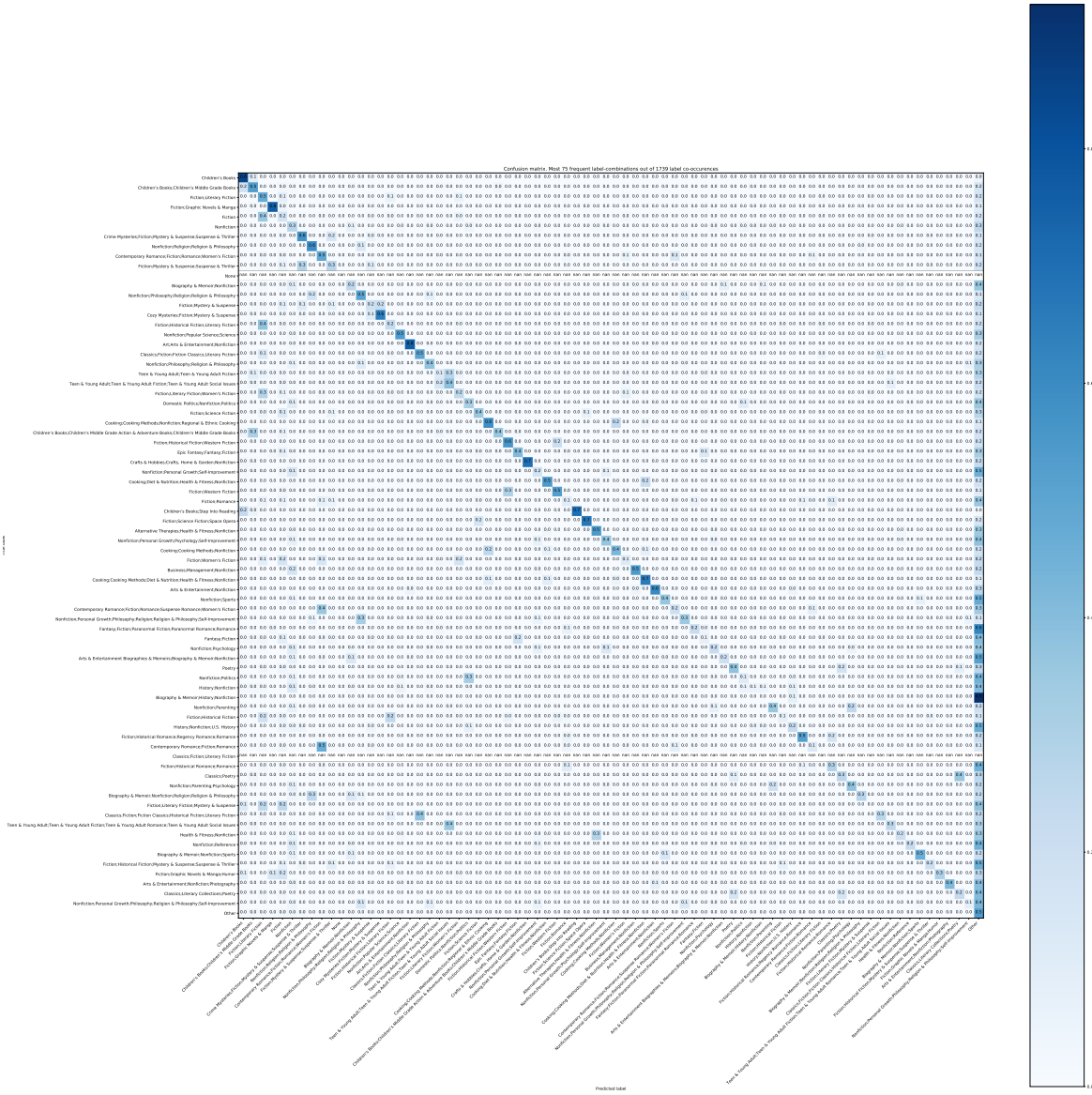


Figure C4: Confusion matrix on the 75 most frequent genre combinations of capsule networks predictions. On the y-axis are the true labels. Results are normalized. Genre combinations that are not in the top 25 are labeled as *outside*. Empty predictions are represented by an empty string. Each genre of a combination is separated by a *;*.

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen - benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

$Unterschrift:$ $Ort, Datum:$

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Arbeit in die Bibliothek des Fachbereichs Informatik zu.

$Unterschrift:$ $Ort, Datum:$