FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis Informatics

Topic Classification for Clauses in Terms of Services with Machine Learning

Jan Robin Geibel

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelor's Thesis Informatics

Topic Classification for Clauses in Terms of Services with Machine Learning

Themenklassifizierung für Klauseln in Allgemeinen Geschäftsbedingungen mit maschinellem Lernen

Author:	Jan Robin Geibel
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Daniel Braun, M.Sc.
Date:	October 15, 2020

Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 17. September 2020 Jan Robin Geibel

Acknowledgments

I would like to thank Daniel Braun for his continued support and advice during this project.

Abstract

General terms and conditions are of significant importance in today's economy. They define to a large extent the circumstances under which transactions are conducted and services are offered. In light of the large number of terms of service agreements entered into today, as well as their complexity, it does not seem feasible for individual consumers to monitor their implications in detail. In fact, several studies indicate that few consumers actually read the many contracts they conclude online. It is, thus, of vital interest how Artificial Intelligence and software applications may be used to effectively communicate the content of terms of services or highlight their potentially controversial elements in order to protect consumers. Hence, the goal of this thesis is to investigate possibilities of using Machine Learning to automatically classify individual terms of such agreements according to their topic. A large part of the project is concerned with devising a suitable categorization as well as building and labeling an adequate corpus which Supervised Learning algorithms can be trained with. The theoretical bases of various Machine Learning algorithms are elaborated and their ability to accurately identify a clause's topic is experimentally investigated. The results obtained during the project and the algorithms' observed behavior as well as the potential causes are presented and discussed.

Contents

1. Introduction

Standard form contracts have become a significant part of today's economy, governing billions of transactions every year [3]. Their importance only seems to increase as the number of purchases made online grows larger. Yet, consumers appear to rarely read the numerous terms of service agreements they enter into today [27]. The cost of reading and assessing rather than finding those agreements seems to be the primary reason for consumers to simply accept them without actually knowing their contents [3].
Interest in the potential use of Artificial Intelligence and software for consumer protection applications, for instance detecting possibly unfair clauses [24], has grown as a consequence. The goal pursued by this thesis is hence to investigate the use of Machine Learning to classify individual clauses of terms of service agreements according to their topic. In particular, this thesis focuses on online consumer contracts being used by e-commerce platforms.

Machine Learning is a field of Computer Science concerned with the design of algorithms that are able to automatically learn a set of decision rules and patterns implicit in a given data set, rather than having to program those rules explicitly [31]. In supervised Learning, the branch of Machine Learning this thesis is mainly concerned with, the algorithm is presented with a large number of data points each of which is assigned a label describing a certain characteristic of interest. The algorithm hence attempts to learn the relationship between the training examples and their assigned labels implicit in the given data set, in order to infer a label for previously unknown data points [31]. Thus, one prerequisite for the deployment of supervised learning is the availability of an annotated data set that represents the characteristics of interest. A large part of the project is, therefore, concerned with building and labeling an adequate corpus accordingly.
A clause of a terms of service agreement can in some cases address a variety of topics at once. Rather than receiving a single label, each data point may, therefore, be assigned a set of labels. The problem at hand is, thus, not just characterized by multiple classes but also an instance of multi-label classification. Multi-label classification describes a classification problem in which multiple labels can be assigned to a data point simultaneously [7].

Machine Learning has found numerous applications in the context of legal contracts and proceedings. The first chapter of this paper briefly reviews prior related work and previous attempts to classifying clauses of online consumer contracts and privacy policies of online platforms.
Devising a suitable categorization of clauses of terms of service agreements and assembling a corpus accordingly has been a significant part of this project. Consequently, the second chapter is intended to give a general understanding of the origin and key characteristics of the data as well as the classes used during the process.
The subsequent chapter provides a brief overview of the theoretical bases of the data pre-

processing techniques as well as the various Machine Learning algorithms used.
Thereafter, the way key elements of the project were implemented is described and the concrete architectures of the deep learning models used throughout the project are portrayed.
Subsequently, the metrics used to evaluate the performance of individual classifiers are briefly explained and their expressiveness as well as their potential shortcomings are outlined. Furthermore, the project's experimental procedure and the obtained results are described in chronological order. Finally, possible explanations for the algorithms' observed behavior are discussed.
The last part of this thesis summarizes the project's results and makes some concluding remarks. Additionally, possible next steps that go beyond the current project's scope are elaborated.

2. Related Work

Machine Learning has been repeatedly applied in the context of legal contracts and proceedings. Examples include the prediction of court decisions [1], the extraction of arguments from legal documents [26] or the detection of claims in legal judgments [22]. The ones most relevant to this project, however, are attempts to classify the clauses of online consumer contracts according to their contents and their fairness as well as the use of classification algorithms for clauses in privacy policies of online platforms.

Lippi et al. attempt to use Machine Learning models to identify potentially unfair clauses of terms of service agreements [23]. The experimental study uses a comparably small corpus containing consumer contracts from 20 online platforms such *eBay* or *Amazon* and labels identifying each sentence's topic and estimating its fairness in form of a numeric value. Rather than classifying entire clauses, the authors focus on classifying individual sentences. The authors investigate the use of two SVM-based classifiers and a kernel machine which uses Tree Kernels.

In 2019 Lippi et al. introduce CLAUDETTE, a Machine Learning system to semi-automatically identify possible unfair clauses in online terms of service agreements [24]. A corpus consisting of 50 online consumer contracts is assembled and, similar to the one described above, contains for each sentence labels identifying its topic and indicating its potential unfairness in form of a numeric value. They train and evaluate several Machine Learning algorithms, including a variety of SVM's (see Section 4.3.1.) and different combinations thereof, a CNN (see Section 4.3.5.), a LSTM (see Section 4.3.6.) and finally use an ensemble algorithm which combines the predictions of the described models.

Lagioia et al. build on the results achieved by CLAUDETTE and explore the possibility of using memory-augmented deep learning (MANN) models (refer to [37] or [16] for more information) to identify unfair clauses of terms of service agreements [19]. The goal pursued by the authors was to investigate whether the use of a knowledge base of human expert rationales of determining a clause's fairness can improve the performance on the given classification task. They, too, focus on the classification of sentences but confine the scope to those unfairly limiting the particular platform's liability. The corpus used is made from 100 consumer contracts and the employed knowledge base consists of legal rationales developed by two experts. The experimental results show a significant improvement of performance in comparison to the thus far deployed deep learning models. The MANN also outperforms the SVM, though to a lesser degree.

The application of Machine Learning in the legal evaluation of privacy policies is, for example, investigated by Contissa et al. [9]. The authors use a web crawler to surveille and retrieve the privacy policies of a number of online services. The corpus assembled for the experimental study consists of 14 privacy policies. The sentences of those were decided to either provide sufficient or insufficient information in regard to the processing of data, contain unclear language or reference a potentially problematic data processing. In a first step, several algorithms for identifying problematic clauses are evaluated. In a second step, an SVM was trained to identify the class describing in what way the given clause is legally problematic.

3. The Corpus

The corpus was assembled in order to train and evaluate supervised Machine Learning algorithms to identify the topics of clauses of terms of service agreements used by e-commerce websites. It consists of 5020 clauses in German that were obtained from 142 e-commerce shops. The majority of these are located in Germany with the exception of one being headquartered in the United Kingdom, one in the Netherlands, one in Luxemburg and two in the Czech Republic.
Figure 3.1. shows the nature of the products being offered by the e-commerce shops the data was obtained from. The categorization is, however, not mutually exclusive resulting in a sum greater than 142. Furthermore, it is also not intended to be a completely exhaustive categorization that covers the entire e-commerce spectrum but merely to provide a generic understanding of the origin of the data. A *general store* refers to an e-commerce platform offering a large variety of product categories such as *amazon.com*.

For each clause, the corpus contains an identifier as well as information about the e-commerce website it was obtained from in addition to its title and actual text. It further includes the title of the paragraph the clause belongs to and if present the text describing the paragraph and thus applying to all clauses contained therein.

3.1. Description of Classes

The clauses obtained were divided into several classes according to their contents. The classification, however, only distinguishes clauses based on their topics and makes no assessment whether or not their legal implications are the same. A clause that, for example, addresses possible warranty claims and explicitly stipulates them receives the label *warranty:contractualClaims*, even if the claims mentioned may be identical to the ones established by the law.

The clauses and the associated information were manually collected from the websites of the above-mentioned e-commerce shops and stored in an Excel file. The same Excel file further includes the information about the e-commerce platforms the clauses were obtained from. The data was annotated within the same file and later exported to a CSV format in order to simplify their future use.

The classes used throughout the project were partly provided by the SEBIS Chair[1] which this project is supervised by and partly derived in an iterative manner by repeatedly grouping clauses according to their contents. In every iteration, a finer distinction between different topics was made. The classification follows a hierarchical approach and assigns

[1]Chair for Software Engineering for Business Information Systems of the Technical University of Munich

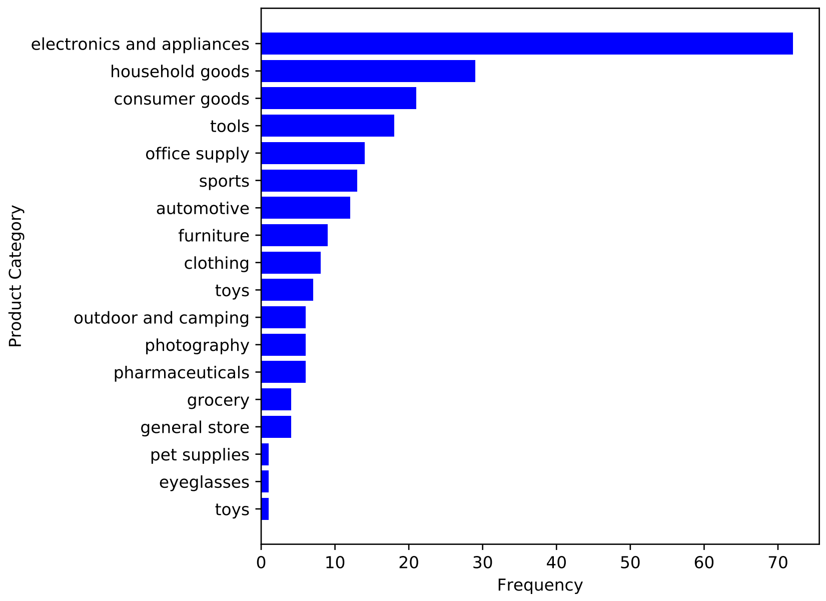

Figure 3.1.: Nature of products sold by e-commerce shops the data was obtained from

a rather broad label (level 1 hereafter) in a first step. These broad classes are in some cases then further subdivided into more granular classes (level 2 hereafter) according to the particular aspect of a topic they are addressing. Not every terms of service agreement collected, however, necessarily contains clauses belonging to every of the classes described below.

batteryElectronicsAndPackaging: Clauses involving the disposal of batteries, electronics or packaging or the associated environmental regulations.

choiceOfLaw: Clauses defining the law the contract is governed by.

codeOfConduct: Clauses concerning codes of conduct to which the particular e-commerce shop has subscribed.

contractChange: Clauses stipulating the e-commerce shop's right to unilaterally change its terms of service agreement.

contractLanguage: Clauses stipulating in which languages the contract can be concluded.

contractObject: Clauses describing aspects of the product or service which is the object of the contract.

contractStorage: Clauses specifying whether the contract text is saved after the contract is concluded.

contractTermination: Clauses concerning the termination of subscription agreements.

customerSupport: Clauses referring to an e-commerce shop's customer support and possibilities for customers to voice concerns, ask questions or make a complaint.

dataProtectionAndProcessing: Clauses addressing data protection, the processing of data or the disclosure of personal information to third parties.

definition: Clauses defining terms used throughout the terms of service agreement.

- **definition:consumer**: Definition of a consumer in terms of the particular agreement.
- **definition:customer**: Definition of a customer in terms of the particular agreement.
- **definition:entrepreneur**: Definition of an entrepreneur in terms of the particular agreement.
- **definition:workday**: Definition of a workday in terms of the particular agreement.
- **definition:other**: Other definitions used throughout the particular agreement. Examples include definitions of digital content or products for demonstrational purposes.

delivery: Clauses addressing any aspect of the delivery of products or services offered by the particular e-commerce shop.

- **delivery:acceptance**: Clauses regarding the acceptance of a delivery by the customer and the consequences should a delivery fail due to a reason the customer is responsible for.
- **delivery:collection**: Clauses concerning the customer's option to personally collect the contract object.
- **delivery:costs**: Clauses addressing the costs of the delivery.
- **delivery:destination**: Clauses regarding the delivery address, geographical limitations of the delivery or the delivery to packing stations.
- **delivery:inspectionAndDamages**: Clauses addressing the inspection of the delivered goods upon arrival as well as damages to the product that may occur during the delivery.
- **delivery:liability**: Clauses stating which contract party carries the liability for potential damages or the potential loss of products that are being delivered as well as the moment in time that liability passes from the vendor to the customer.
- **delivery:method**: Clauses regarding the method the contract object is being delivered by. This can, for instance, mean the delivery services or freight company that is commissioned to deliver the contract object, or the way digital content is transmitted electronically.

- **delivery:partialDeliveries**: Clauses regarding the possibility of the order being split into multiple deliveries.
- **delivery:productAvailability**: Clauses concerning the availability of products.
- **delivery:time**: Clauses specifying or addressing the time frame within which the contract object is expected to be delivered to the customer.

disputeResolution: Clauses concerning the resolution of disputes in front of a consumer arbitration board and the online platform for dispute settlement provided by the European Commission.

intellectualProperty: Clauses referring to the ownership of intellectual property, copyrights and rights of third parties.

liabilityScope: Clauses stipulating the scope of the liability assumed by the particular e-commerce shop.

orderRestrictions: Clauses concerning any restrictions of the order. This can be export controls or limitations placed by the particular e-commerce shop on the product quantities that can be ordered.

party: Clauses referring to any of the contract parties.

payment: Clauses concerning the payment of the products or services being purchased by the customer.

- **payment:costs**: Clauses addressing possible costs in regard to the payment of the order.
- **payment:creditRating**: Clauses regarding an assessment of the customer's credit-worthiness and the accuracy of his information.
- **payment:default**: Clauses specifying when a customer is defaulting on a payment and the resulting consequences.
- **payment:invoice**: Clauses concerning the invoice for the particular order.
- **payment:method**: Clauses regarding any aspect of the payment methods offered by the particular e-commerce shop.
- **payment:nettingAndWithholding**: Clauses addressing the customer's right to net receivables or withhold payments.
- **payment:time**: Clauses specifying when a payment is due.

placeOfFulfillment: Clauses referring to the contract's place of fulfillment.

placeOfJurisdiction: Clauses defining the contract's place of jurisdiction.

prices: Clauses concerning the prices quoted by the particular e-commerce shop.

realization: Clauses addressing the circumstances of the conclusion of the contract.

- **realization:offerAndAcceptance**: Clauses specifying what stipulates an offer to conclude a contract and when and how such an offer is accepted.
- **realization:orderProcess**: Clauses describing the technical details of the contract conclusion and the order process.

retentionOfTitle: Clauses stipulating the moment in time the ownership of the contract object passes to the customer.

rightToRefuse: Clauses defining the e-commerce shop's right to refuse to accept an order.

salvatorius: Salvatorian clauses.

scope: Clauses defining the scope of the particular terms of service agreement as well as the objection to any general terms and conditions the customer may have.

sellerWithdrawalRight: Clauses stating cases in which the particular e-commerce shop has the right to withdraw from the contract.

userAccount: Clauses referring to the customer's user account and the registration process.

vouchersDiscountsPromotionsGiftcards: Clauses concerning promotional campaigns, discounts or gift cards in any regard.

warranty: Clauses addressing the customer's warranty claims.

- **warranty:contractualClaims**: Clauses referring to guarantees given by the manufacturer of the products being offered or those explicitly stipulating any warranty policy.
- **warranty:exclusion**: Clauses explicitly excluding certain types of products or defects from the stated warranty claims.
- **warranty:lapse**: Clauses specifying the time period after which the stated warranty claims lapse.
- **warranty:legalClaims**: Clauses stipulating that the legal warranty claims apply.

withdrawal: Clauses addressing the customer's ability to withdraw from the contract after its conclusion. This includes both the legal withdrawal right as well as withdrawal options voluntarily offered by the particular e-commerce shop.

- **withdrawal:consequences**: Clauses describing the consequences of a withdrawal from the contract.
- **withdrawal:exclusion**: Clauses describing types of products or contracts for which no withdrawal right exists.
- **withdrawal:form**: The form which needs to be filled out by the customer to exercise his withdrawal right.
- **withdrawal:right**: Clauses informing the customer about his right to withdraw from the contract and the timeframe during which that right can be exercised.

3.2. Distribution of Clauses Among Classes

The distribution of clauses among level 1 classes is heavily skewed as is shown in Figure 3.2. The five most frequently occuring classes *delivery*, *payment*, *realization*, *warranty* and *withdrawal* make up about half of the labels given to clauses.

The distribution of clauses among level 2 classes (Figures 3.3.−3.8.) is similarly concentrated. For instance, 45.6% of the level 2 labels given to clauses belonging to the level 1 class *payment* were the label *payment:method*.

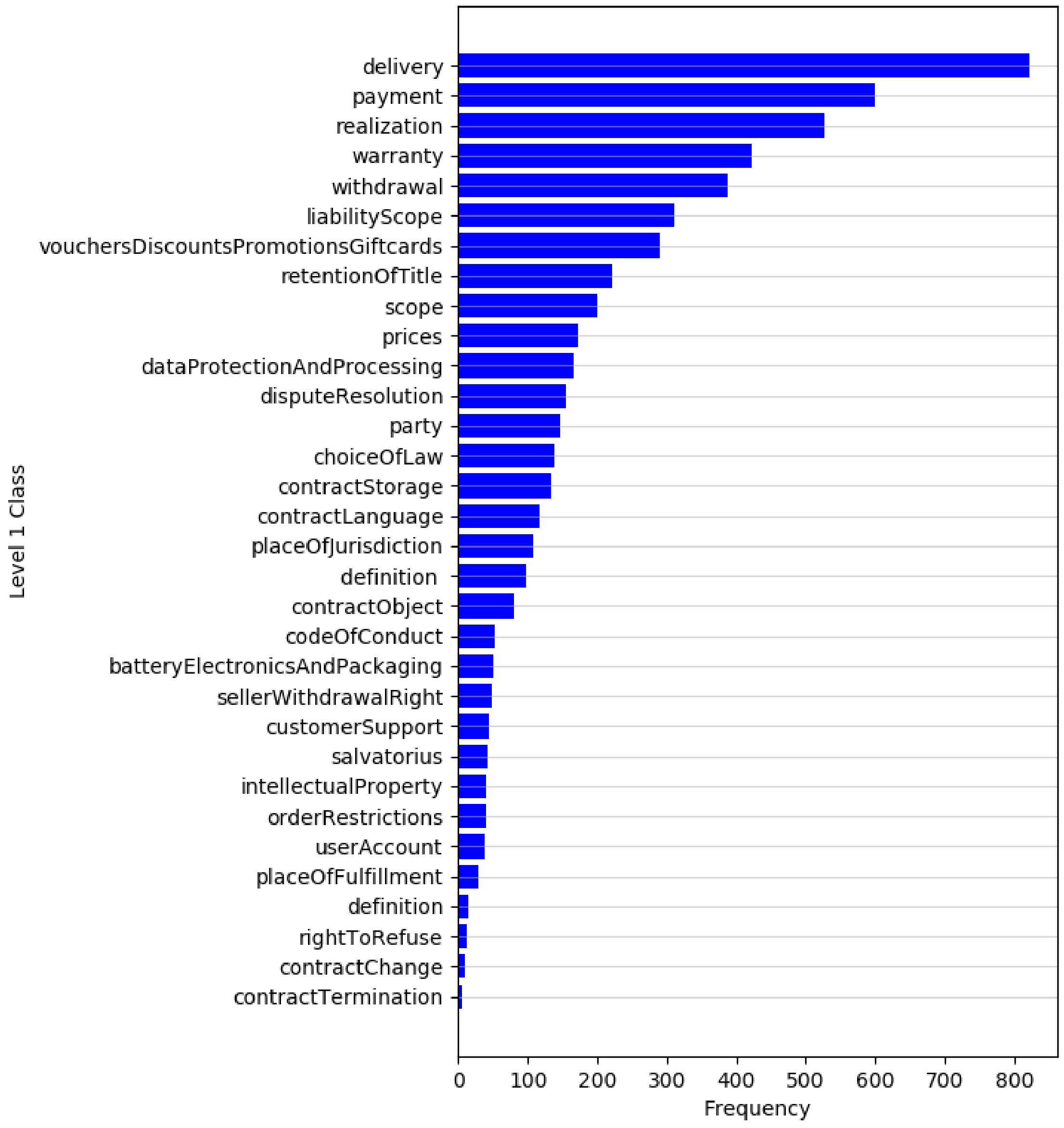

Figure 3.2.: Distribution of clauses among level 1 classes

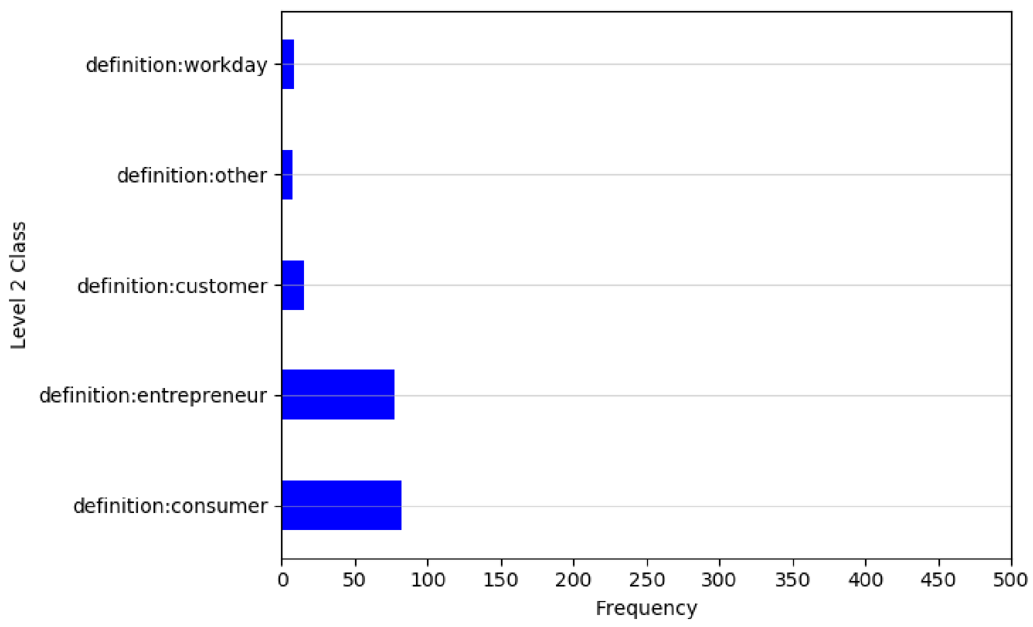

Figure 3.3.: Distribution of Level 2 classes within Level 1 class definition

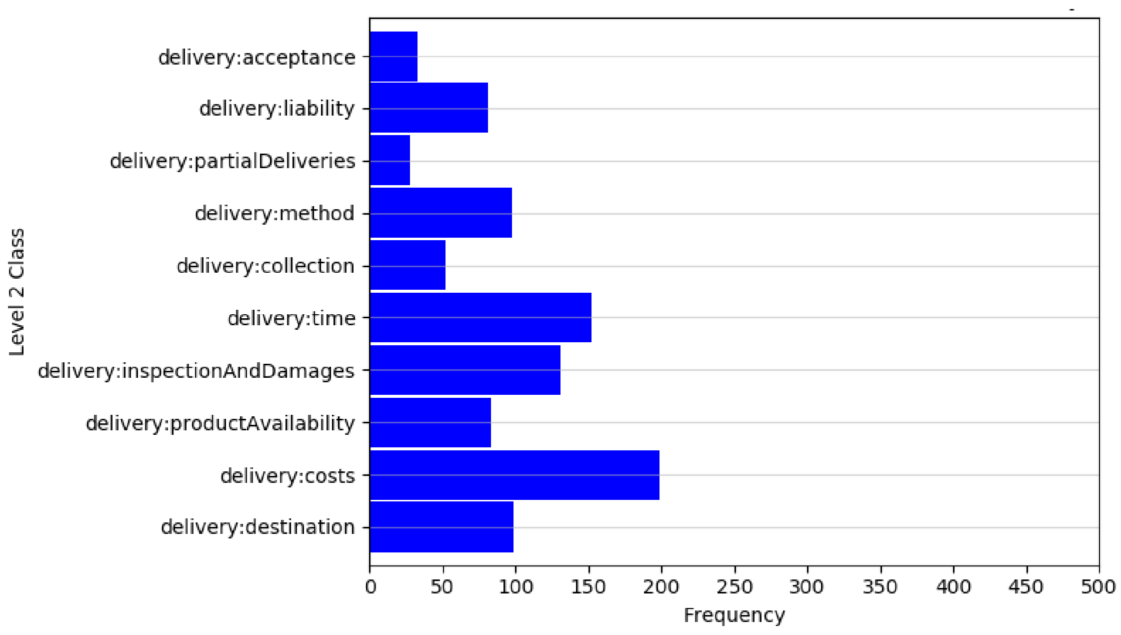

Figure 3.4.: Distribution of Level 2 classes within Level 1 class delivery

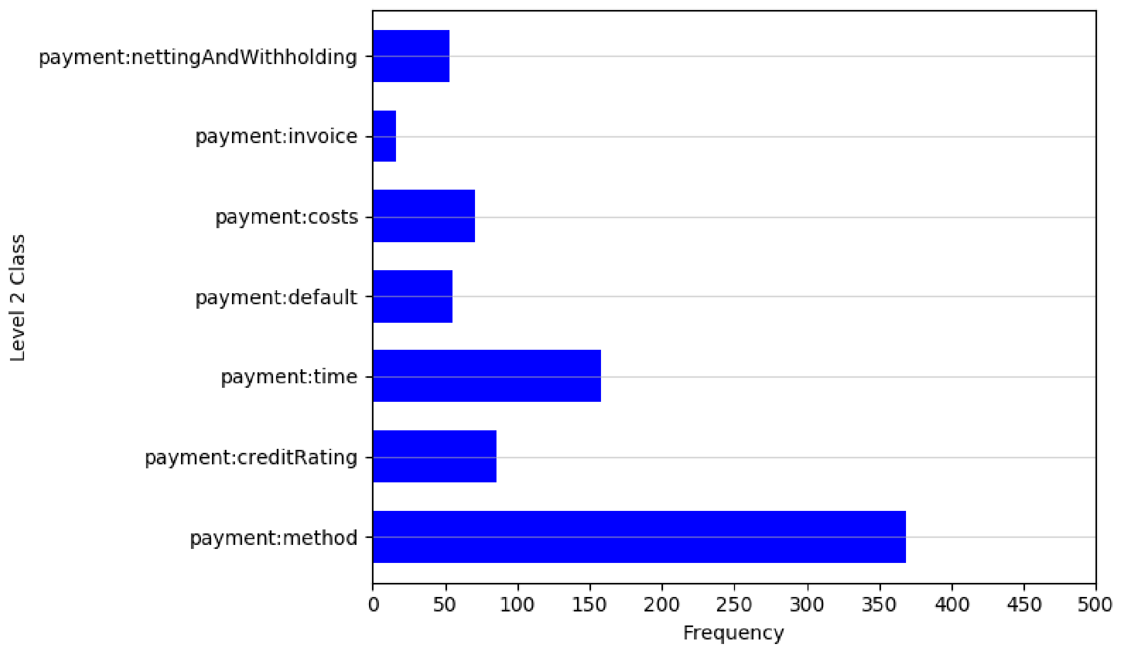

Figure 3.5.: Distribution of Level 2 classes within Level 1 class payment

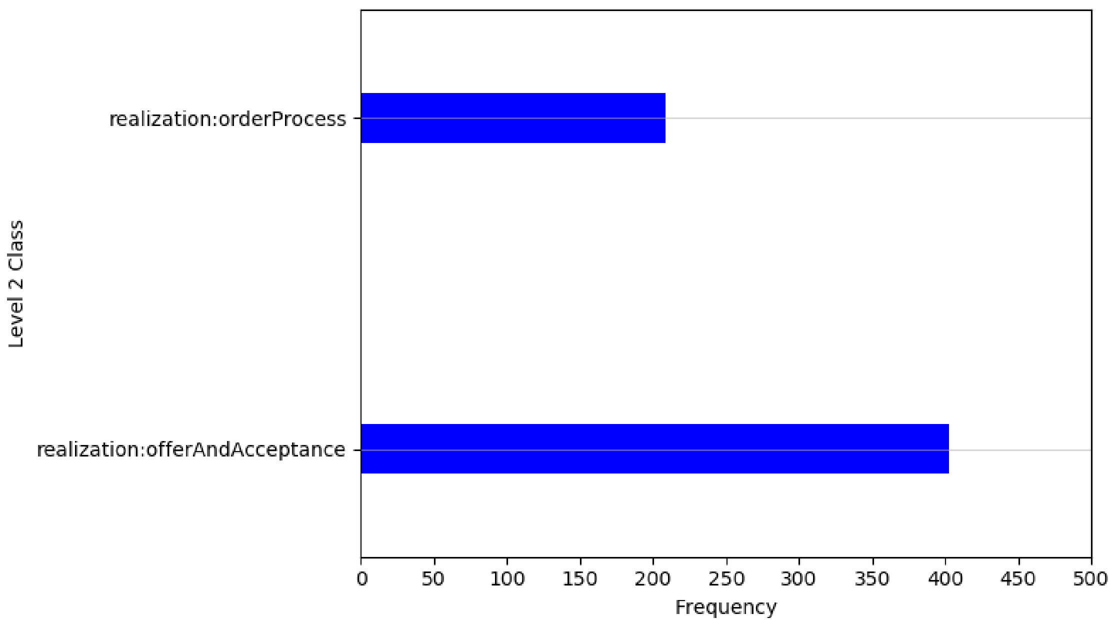

Figure 3.6.: Distribution of Level 2 classes within Level 1 class realization

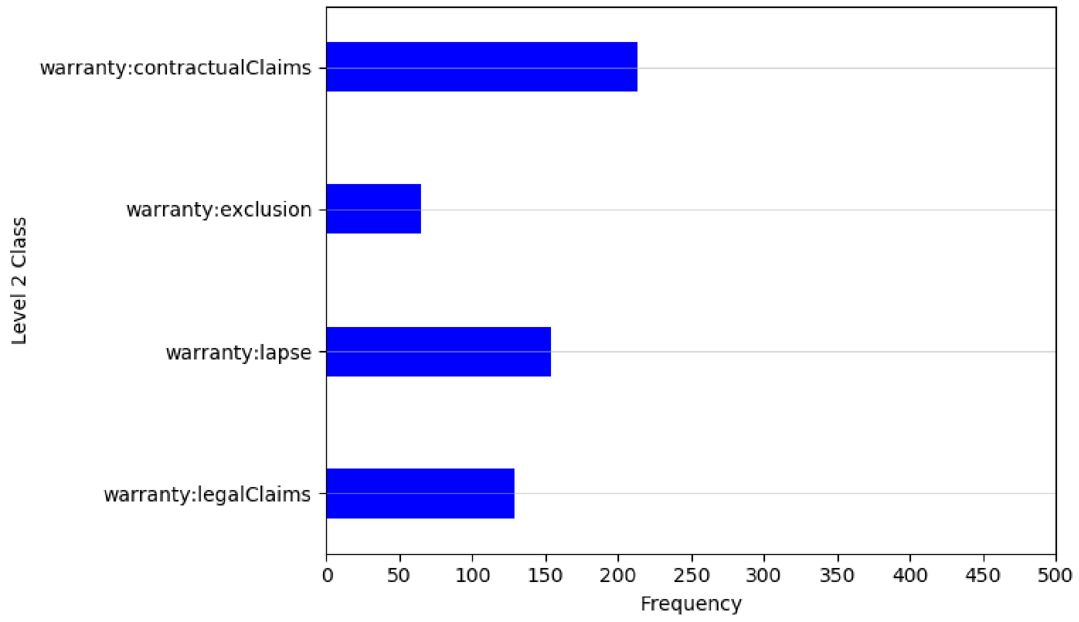

Figure 3.7.: Distribution of Level 2 classes within Level 1 class warranty

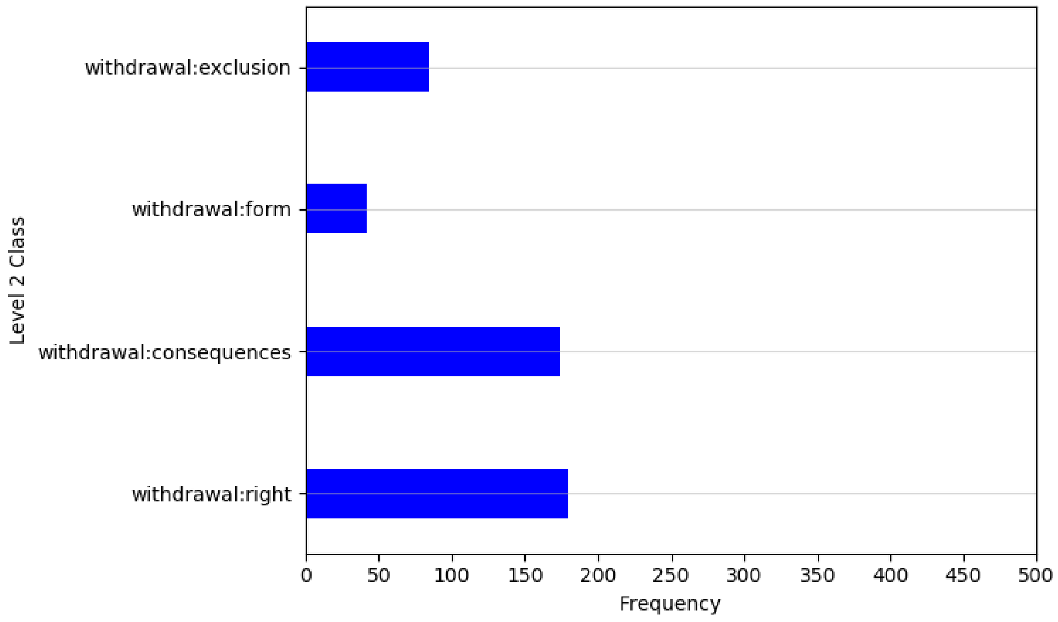

Figure 3.8.: Distribution of Level 2 classes within Level 1 class withdrawal

4. Background

4.1. Data Preprocessing

4.1.1. Tokenization and Stopwords

Tokenization describes the breaking up of text data into individual components, the so-called *tokens*, such as words or phrases. Stopwords are words that occur in the majority of the documents or texts to be classified and thus carry little information. Stopwords are, therefore, often removed from the data set [18].

4.1.2. Lemmatization and Stemming

In order to reduce the complexity of the data set while preserving the information it contains, different varieties of a single word which carry the same semantic meaning are consolidated. Two common strategies are stemming and lemmatization [18]. Stemming describes a method in which different varieties of a word are replaced by a common form [25]. Lemmatization is a way of normalizing a word by either stripping it of its suffix or appending one. Both strategies are inherently similar. However, lemmatization algorithms do not necessarily result in a word's stem and may thus produce different results in some cases [29]. Lemmatization has been found to outperform stemming with one possible reason being that it also takes a word's synonyms under consideration which stemming does not [4]. Hence, lemmatization is used throughout this project.

4.2. Feature Engineering

4.2.1. Term Frequency-Inverse Document Frequency

One method to extract features from the corpus that can be represented mathematically is the so-called *Term Frequency-Inverse Document Frequency* (TF-IDF). The inverse document frequency diminishes the siginificance given to words occuring very frequently throughout the entire corpus while increasing the weight assigned to those occuring rarely. This information is combined with the frequency with which a term occurs in a particular document [18].

The TF-IDF is thus calculated as follows:

$$TF - IDF = TF(d,t) * log(\frac{N}{df(t)}) \tag{4.1}$$

with

- $TF(d,t)$ being the frequency of term t in document d,
- N being the number of documents in the corpus,
- and $df(t)$ beeing the number of documents in which the term t occurs [18].

4.2.2. Word Embeddings

Word embeddings describe methods in which the vocabulary, the set of words making up the entire corpus, are represented by N-dimensional vectors [18]. A distributed semantic model is trained on a natural language corpus in order to represent each word by a numerical vector [2]. Word embeddings are then able to represent the lexical semantic relationship between words by the geometric proximity of their vectors [14].

Examples of commonly used word embedding frameworks are GloVe [28], Word2Vec [32] or FastText [6].

4.3. Machine Learning Algorithms

4.3.1. Support Vector Machines

The Support Vector Machine (SVM) was originally thought of in 1964 by Chervonekis and Vapnik [36] as a class of algorithms for pattern recognition tasks but has since been successfully applied in a number of different contexts for both regression as well as classification purposes [11].

An SVM classifier attempts to separate the data points by a hyperplane in a way that maximizes the distance or so called 'margin' between it and the data points of the two classes on either side of it [17]. If the data is separated linearly, the classification of a data point is determined by the equation

$$f(x) = sgn(w^T x + b) \tag{4.2}$$

$$\text{with } sgn(w^T x + b) \geq 1, \forall x \in P \tag{4.3}$$

$$\text{and } sgn(w^T x + b) \leq 1, \forall x \in N \text{ [17].} \tag{4.4}$$

The original design of the SVM used a linear separation which has since been further enhanced to a non-linear one [18]. In case of data that cannot be completely separated by any hyperplane, one solution is to essentially transform the input data into a higher dimensional space. The thus created *transformed feature space* is hence linearly separated by a hyperplane which is equivalent to a non-linear separation in the initial input space [17].

The higher dimensional space into which the data is mapped is created by a so-called *kernel function*. Kernel functions allow to make the necessary calculations during the training process directly in the feature space without actually having to transform the data. Novel data points are thereafter mapped into the higher dimensional feature space by using the kernel function and can then be classified by the previously obtained hyperplane [17]. Examples of commonly used kernel functions are the RBF kernel or the Gaussian kernel [17].

The training process results in a global optimum rather than a local one as may be the case in other learning algorithms [17].

The SVM was originally designed as a binary classifier [18]. One way to adapt the algorithm to a multi-class problem, which is used throughout this project, is the *one-vs-the-rest* or also called *All-vs-One* approach. The one-vs-the-rest approach is based on training multiple classifiers with each one being trained to distinguish instances of one class from the remaining ones [5].

4.3.2. Logistic Regression

Logistic Regression (LR) is a way to assign probabilities to a data point being in either of the classes rather than directly predicting the class it belongs to [18]. It is a linear classifier that was initially conceived by David Cox in as early as 1958 [18].

LR is a binary classifier with the probability of a data point x belonging to the class of interest given as

$$p(y = 1|x, \theta) = sigmoid(x\theta) \tag{4.5}$$

with the sigmoid function defined as:

$$sigmoid(x\theta) = \frac{1}{1 + e^{-(x\theta)}} \tag{4.6}$$

The parameter vector θ can either be determined by Maximum Likelihood estimation and thus the maximation of

$$\prod_{i=1}^{n} p(y_i|x_i, \theta)$$

or by use of a numerical optimization algorithm such as gradient descent [10].

As discussed above for SVMs, the one-vs-the-rest or also called All-vs-One approach is used to adapt the binary classifier to a multi-class problem.

4.3.3. k-Nearest Neighbors

The k-nearest neighbors classification algorithm (kNN) is a non-parametric algorithm [18] that assigns a label to a previously unknown data point according to the k data points of

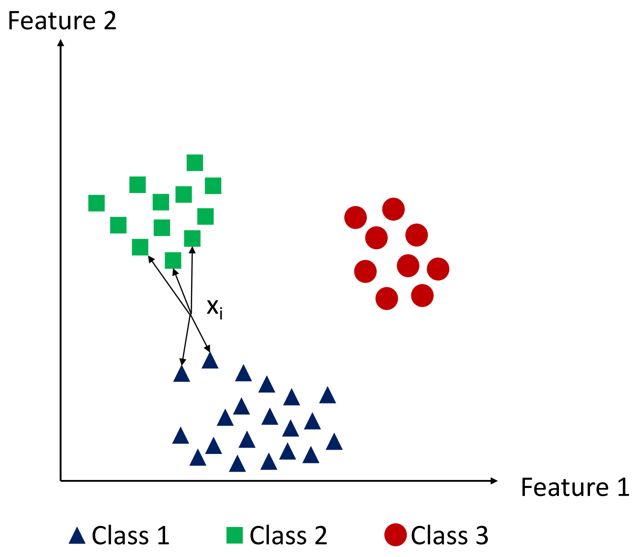

Figure 4.1.: k-Nearest Neighbors (based on [18])

the training set which are most similar to it, according to a predefined metric. The classification is then, for instance, performed by identifying the label that occurs most frequently among those k examples [17]. The performance of kNN, however, depends to a large degree on the choice of the parameter k which is determined beforehand. If the chosen k is too small the algorithm is more likely to produce less meaningful results since fewer examples are considered to infer the label of a new data point. If the chosen k is too large on the other hand, classes with fewer data points assigned to them will be dominated by those having a large number of representatives in the training set [21]. kNN is an example of an instance-based learning technique which requires more time to classify new data points but performs fewer computations during the training process [17].

The k nearest neighbors to a new data point are determined by calculating the relative distance to the instances of the training data set. If the data is characterized by n features, it is regarded as points in an n-dimensional space (see Figure 4.1.) [17]. Examples for distance metrics frequently used in the algorithm are the Euclidian or the Manhattan distance [17].

There are multiple functions by which a new data point x can be classified after the k nearest data points have been determined. Two commonly used decision functions are the following:

$$y(d_1) = arg\ max_k \sum_{x_j \in kNN} y(x_j, c_k) \tag{4.7}$$

$$y(d_1) = arg\ max_k \sum_{x_j \in kNN} Sim(d_i, x_j) y(x_j, c_k) \tag{4.8}$$

with

- d_i being the data point to be classified,
- x_j being a data point in the training set,
- $y(x_j, c_k) \in \{0, 1\}$ being 1 if x_j belongs to the class c_k
- and $Sim(d_j, x_j)$ being the function calculating the chosen similarity metric [21].

In the first case, x is being assigned the same label as the majority of its k nearest data points whereas in the second case, x is assigned the label for which the sum of the similarity measures is the largest [21]. The implementation chosen for this project uses Bayesian inference to select the assigned labels in order to accommodate the multi-label nature of the classification problem.

4.3.4. Multilayered Perceptron

The Multilayered Perceptron (MLP) was developed in order to classify data which cannot be separated linearly into different classes [17]. A Multilayered Perceptron, also termed Artificial Neural Network (ANN), is made of an input layer, an output layer and the so-called hidden layers located between the two. The individual units or neurons of each layer are connected to the ones of preceding and following layers, thus creating the architecture of the particular model (see Figure 4.2.) [17]. In a feed-forward ANN, the input only passes from one direction to the output layer by performing a number of calculations under the use of the weights assigned to the connections between the units as parameters [17].

In the first step of the classification, M linear combinations are formed out of the input variables x_i, weights w_{ij} and biases w_{j0} as is shown in equation 4.9 with $i \in [1, D]$ and $j \in [1, M]$.

$$a_j = \sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)} \tag{4.9}$$

In a second step, a non-linear activation function is applied to the result a_j, the so-called *activation*.

$$z_j = h(a_j) \tag{4.10}$$

The resulting values z_j are then again used to construct linear combinations in the manner shown in equation 3.11. The described steps are repeated for every hidden layer of the

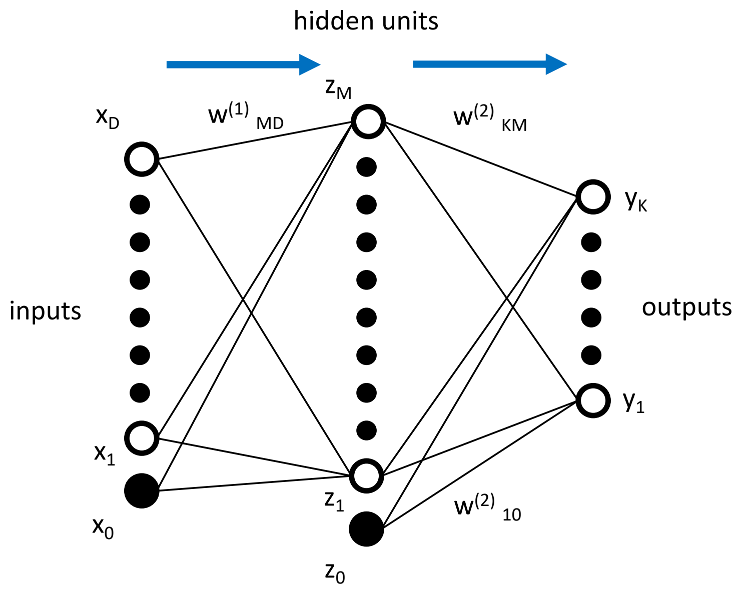

Figure 4.2.: Multilayered Perceptron (based on [5])

particular network until, lastly, the output activation units a_k, with $k \in [1, K]$ and K being the number of output values, are calculated.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{4.11}$$

An activation function like the sigmoid function (see equation 4.6) is applied to the output activation units in order to create the output values y_k.

$$y_k = sigmoid(a_k) \text{ [5]} \tag{4.12}$$

One commonly used algorithm to train the model and determine the weights assigned to the connections between the units is the Back Propagation (BP) algorithm [17]. During the training of the model, the ANN is continuously used to classify data points of the training set. In each iteration, the model's output for the particular training example is compared to its actual label. For every neuron, the weights are adjusted in the direction needed to produce the desired output. This local error, the difference between a neuron's actual and desired output, is then passed on to the neurons of previous layers in relation to the strength of their connection. The error is thus propagated back through the entire ANN and the weights of each layer are adjusted accordingly [17].

4.3.5. Convolutional Neural Network

Convolutional Neural Networks (CNN) are deep learning algorithms similar to the Multilayered Perceptron that were originally designed for Computer Vision and the processing

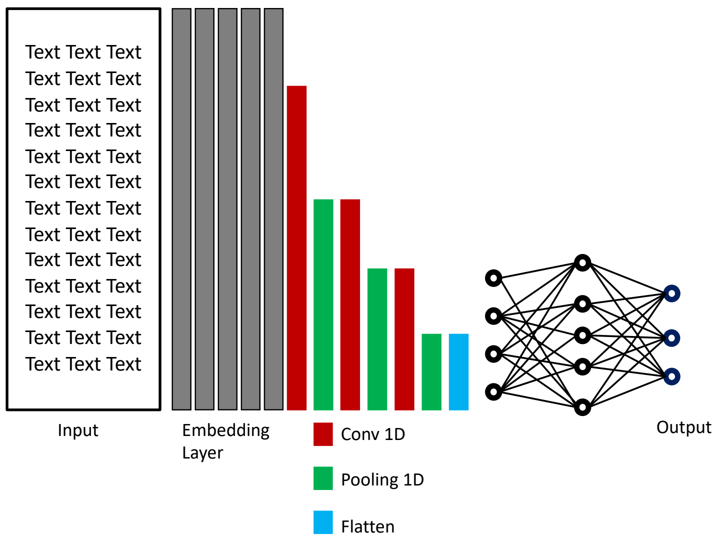

Figure 4.3.: CNN for text classification (based on [18])

of image data [18]. Figure 4.3. shows an exemplary architecture of a Convolutional Neural Network for the classification of text [5]. CNNs intent to leverage the fact that pixels of an image tent to be more strongly correlated the closer they are to each other. Different subregions of an image are, thus, individually processed and the resulting information merged at a later point of the algorithm [5].

In a convolutional layer, neurons are grouped into so-called *feature maps*, each of which processes a subset of the input data. Each feature map, therefore, provides a filter on the input data and is characterized by its individual set of weights and biases [5].

The outputs of the convolutional layers are pooled, e.g. by using max pooling and selecting the largest element of the particular window, in order to reduce the number of parameters passed on to the next layer and thereby reduce the computational complexity of the algorithm [18]. Additionally, a flatten layer is used to reduce the output of multiple stacked feature maps to one column in order to be processed by the following layer [18].

Commonly, the final layers in a CNN architecture are fully connected and result in a softmax output nonlinearity for multi-class problems [5].

4.3.6. Long Short-Term Memory

A Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) [18]. Figure 4.4. shows an example of a LSTM architecture. RNNs are neural networks used to process sequential data such as text and were designed to consider early data points of a series more than other architectures do [18]. The training of the original RNN, however,

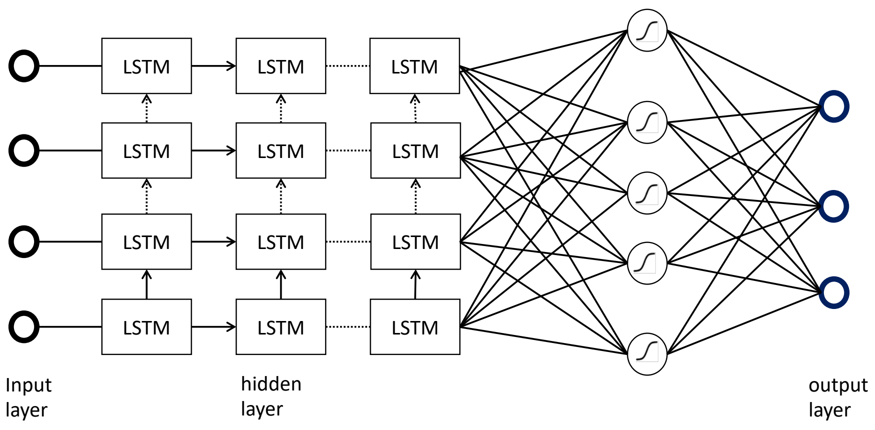

Figure 4.4.: Long Short-Term Memory Architecture (based on [18])

can be significantly impeded by a vanishing or exploding gradient [18]. LSTMs were, thus, introduced to address these issues by regulating the information that is passed on through their chain-like architecture by incorporating multiple gates in its units [18]. Those gates are called the input, output and forget gates [15].

One layer of the model consists of multiple LSTM cells [15] each of which performs the following calculations:

- The input gate is defined as

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \tag{4.13}$$

- The candid memory cell is defined as

$$\tilde{C_t} = tanh(W_c[x_t, h_{t-1} + b_c]) \tag{4.14}$$

- The forget-gate activation is calculated as

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{4.15}$$

- The new value of the memory cell is defined as

$$C_t = i_t * \tilde{C_t} + f_t C_{t-1} \tag{4.16}$$

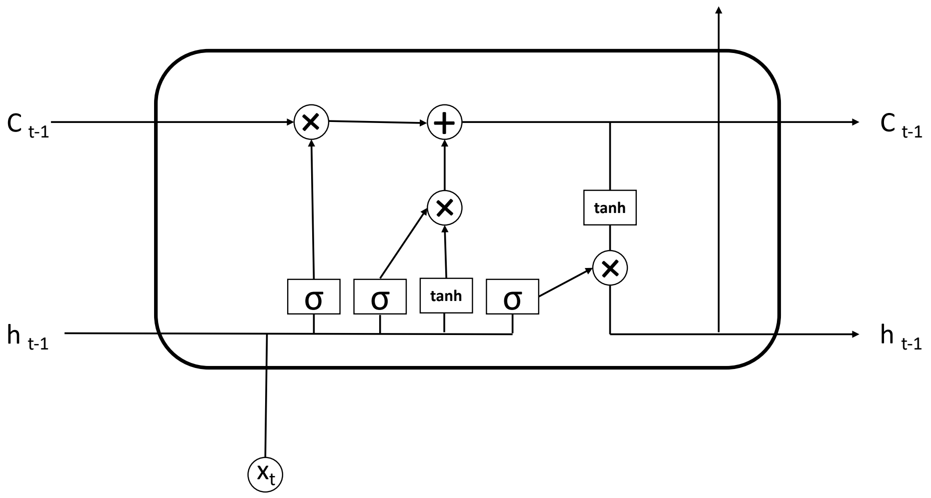

Figure 4.5.: Long Short-Term Memory Cell (based on [18])

- The resulting value of the output is derived by the following calculations

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o) \tag{4.17}$$

$$h_t = o_t \, tanh(C_t) \text{ [18]} \tag{4.18}$$

The described calculations are performed in every iteration and can be thought of as write, read and reset operations for the particular cell [15]. Figure 4.5. illustrates the design of an LSTM cell and the calculations described above.

5. Implementation

Since a clause of a terms of service agreement may address a variety of topics at once, the problem at hand is an instance of multi-label classification. Multi-label classification describes a classification problem in which multiple labels can be assigned to a data point simultaneously [7].

One possible way to address a multi-label classification problem is the so-called *Label Powerset* (LP) approach. The LP method transforms the multi-label problem into a multi-class problem by treating every combination of labels as a distinct class [33]. This would, however, result in a large number of classes with relatively few training examples being associated with each of them. Rather than transforming the problem into a multi-class classification, the approach chosen for this project, therefore, attempts to arrive at a ranking or probability estimate for each label being associated to a particular data point. The data point is then assigned every label whose score or probability estimate is above a predetermined threshold [30]. In a first step, the classifier is trained on the training set. In a second step, the classifier is evaluated on a validation set in combination with a range of different thresholds. The threshold for which the highest F_1-score (see Section 6.1.4.) on the validation set can be achieved is then selected.

The python libraries Keras[1] and scikit-learn[2] are used to implement the project.

5.1. Data Preprocessing

In a first step after loading the data, the clauses are lemmatized (see Section 4.1.2.) and stopwords (see Section 4.1.1.) are removed from the corpus. Additionally, special characters such as numbers and punctuation marks are removed from the words of the vocabulary. Certain characters such as § or € are, however, intentionally not removed inorder to retain potential references to a currency or legal code.

Moreover, a multi-label binarizer is used to encode the labels assigned to the clauses. Lastly, a test set (20% of the corpus) is split of the corpus while the remaining examples are either entirely used as the training set (the implementation of the multi-label kNN does not require a validation set) or divided again into a training (68% of the corpus) and a validation set (12% of the corpus).

[1]https://keras.io
[2]https://scikit-learn.org/stable/

5.2. Feature Engineering

The TF-IDF vectors (see Section 4.2.1.) of the corpus are created with scikit-learn's TfidfVectorizer [3] functionality. The TF-IDF scores are calculated on a word level and the number of features derived from the corpus is limited to the 10.000 most relevant ones. Additionally, 2-grams are also considered. Rather than only examining single words, the n-gram technique also takes sequences of n consecutive words into account [18]. The TfidfVectorizer also includes tokenization and preprocessing functionality. Since the data has already been tokenized and preprocessed, however, a dummy function that leaves the input unchanged is created and passed as a keyword argument.

```

def tf_idf_vector(X_train, X_val, X_test, val=True):

  tfidf_vectorizer = TfidfVectorizer(analyzer='word', tokenizer=dummy,
    preprocessor=dummy, token_pattern=None,max_features=10000, ngram_range=(1,2)
    )

  xtrain_vec = tfidf_vectorizer.fit_transform(X_train)

  if val:
    xval_vec = tfidf_vectorizer.transform(X_val)
  else:
    xval_vec = []

  xtest_vec = tfidf_vectorizer.transform(X_test)

  return xtrain_vec, xval_vec, xtest_vec
```

Listing 5.1.: TF-IDF Vectors

Word Embeddings are implemented by the use of embedding layers[4] of Keras' Sequential model[5].

5.3. Machine Learning Algorithms

5.3.1. Support Vector Machines

Scikit-learn's SVM[6] functionality is used to implement a Support Vector Classifier (SVC) for text classification. The default kernel function (see Section 4.3.1.) used by the SVC is an RBF-Kernel. SVM's are designed as decision functions and thus do not yield a probability distribution for a data point being in either of the classes [5]. By passing the keyword argument *probability*, however, scikit-learn's SVC implementation can be enabled to create posterior probability estimates. Although such estimates can be inconsistent with the decision that would have been otherwise derived at by the SVC, probability estimates are nonetheless used to account for the multi-label nature of the classification problem. As

[3]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[4]https://keras.io/api/layers/core_layers/embedding/
[5]https://keras.io/guides/sequential_model/
[6]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

described in Section 4.3.1., a one-vs-the-rest approach is used to adapt the binary classifier to multiple classes[7]. After training the model, the validation set is used to derive a suitable probability threshold.

```

# support vector classifier allowing for probability estimates
svc_classifier = SVC(probability=True)

# One vs Rest approach to extend binary classifier to multiple classes
svc_classifier = OneVsRestClassifier(svc_classifier)

# Model training
svc_classifier.fit(xtrain_vec,y_train)

# Evaluation of thresholds
print("Evaluation of thresholds")
threshold = find_th(svc_classifier,xval_vec,y_val,p=True)

# Evaluation of model performance
eval(svc_classifier, model_name, xtrain_vec, y_train,xtest_vec,y_test,xval_vec,
    y_val,threshold,p=False)
```

Listing 5.2.: Support Vector Classifier

Additionally, scikit-learn's pipeline[8] functionality is used to create an SVC that receives multiple inputs in order to classify the topic of a particular clause. A pipeline performs several sequential steps which must be so-called *transforms*, i.e. steps performing either *fit* or *transform* methods. The preprocessed data is passed to the pipeline which then creates the TF-IDF vectors (see Section 5.2.), uses a pre-trained SVC to derive an estimate of the level 1 labels assigned to a clause and determines the length of the clause. Scikit-learn's MinMaxScaler[9] is used to scale the lengths of the clauses to the range between zero and one. Those features are then passed to an SVC in order to predict the level 2 labels of a given clause.

```

# Pipeline handing the TF-IDF weights, level1 prediction
# of a trained SVC classifier, and the clause length to a SVC classifier

svc_classifier_minput = Pipeline([
  ('features', FeatureUnion([
    ('text', Pipeline([
      ('clean', FunctionTransformer(data_prep.clean, validate=False)),
      ('tfidf', TfidfVectorizer(analyzer='word', tokenizer=dummy, preprocessor=
    dummy, token_pattern=None,max_features=10000, ngram_range=(1,2))),
    ])),
    ('level1', Pipeline([
      ('clean', FunctionTransformer(data_prep.clean, validate=False)),
      ('tfidf', TfidfVectorizer(analyzer='word', tokenizer=dummy, preprocessor=
    dummy, token_pattern=None,max_features=10000, ngram_range=(1,2))),
```

[7]https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html
[8]https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
[9]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

```
      ('predict', FunctionTransformer(predict_level1, validate=False)),
    ])),
    ('length', Pipeline([
      ('count', FunctionTransformer(get_text_length, validate=False)),
      ('scaling', preprocessing.MinMaxScaler())
    ]))
  ])),
  ('clf', OneVsRestClassifier(SVC(probability=True)))])

# Training the model
svc_classifier_minput.fit(X_train, y_train)

# Evaluation of thresholds
print("Evaluation of thresholds")
threshold = find_th(svc_classifier_minput,X_val,y_val,p=True)

# Evaluation of model performance
eval(svc_classifier_minput, model_name, X_train, y_train,X_test,y_test,X_val,
    y_val,threshold,p=False)
```

Listing 5.3.: Multi-input Support Vector Classifier

5.3.2. Logistic Regression

LR is implemented under the use of scikit-learn's LogisticRegression[10] functionality and is adapted to multiple classes with a one-vs-the-rest approach[11] similar to the one used in the implementation of the SVC (Section 5.3.1.). LR naturally results in a posterior probability distribution for a clause being in either of the potential classes (Section 4.3.2.). To account for the multi-label classification problem considered here, each label whose probability estimate lies above a certain threshold is assigned to a given clause. The validation set is used to derive an appropriate probability threshold.

```

# logistic regression classifier
logistic_reg = LogisticRegression()

# One vs Rest approach to extend binary classifier to multiple classes
logistic_reg = OneVsRestClassifier(logistic_reg)

# Model training
logistic_reg.fit(xtrain_vec,y_train)

# Evaluation of treshold on validation set
print("Evaluation of thresholds")
threshold = find_th(logistic_reg,xval_vec,y_val)

# Evaluation of model performance
eval(logistic_reg, model_name, xtrain_vec, y_train,xtest_vec,y_test,xval_vec,
    y_val,threshold,p=False)
```

Listing 5.4.: Logistic Regression Classifier

[10]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[11]https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

5.3.3. k-Nearest Neighbors

The kNN algorithm is implemented by using scikit-learn's Multilabel kNN[12] functionality which is inherently designed for multi-label classification problems and uses Bayesian inference to assign a set of labels to a data point after the k nearest neighbors have been determined. Scikit-learn's GridSearchCV[13] is used to determine the values for the parameters k and s that maximize the F_1-score for the given data set. The parameter k defines the number of neighbors that need to be considered in classifying a data point. The parameter s is a smoothing parameter. After the best k and s have been determined, the model is initialized and trained with those values.

```

parameters = {'k': range(1,6), 's': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8,
    0.9, 1.0]}

#Multilabel knn using GridSearch to find the  best parameters for s and k
mlknn_clf = GridSearchCV(MLkNN(), parameters, scoring='f1_micro')

#Finding best parameters
mlknn_clf.fit(xtrain_vec, y_train)

#Best parameters
k = mlknn_clf.best_params_['k']
s = mlknn_clf.best_params_['s']

#Initializing classifier with best parameters
mlknn_classifier = MLkNN(k=k,s=s)

#Model training
mlknn_classifier.fit(xtrain_vec,y_train)
```

Listing 5.5.: Multi-label k-Nearest Neighbors

5.3.4. Multilayered Perceptron

Scikit-learn's MLPClassifier[14] functionality is used to implement a Multilayered Perceptron. The keyword argument *max_iter* defines the number of iterations to be performed by the solver. The process stops either when it converges, or the maximum number of iterations is reached. After training the model, the probability threshold that maximizes the F_1-score for predictions on the validation set is determined.

```

# Multilayered Perceptron
mlp_classifier = MLPClassifier(max_iter=400,random_state=1)

# Model training
mlp_classifier.fit(xtrain_vec,y_train)
```

[12]http://scikit.ml/api/skmultilearn.adapt.mlknn.html
[13]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[14]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

```

# Evaluation of treshold on validation set
print("Evaluation of thresholds")
threshold = find_th(mlp_classifier,xval_vec,y_val)

# Evaluation of model performance
eval(mlp_classifier,model_name,xtrain_vec, y_train,xtest_vec,y_test,xval_vec,
    y_val,threshold,p=False)
```

Listing 5.6.: Multilayered Perceptron

Apart from that, an MLP which receives as features both a clause's length as well as the prediction of a pretrained SVC for its level 1 label in addition to its TF-IDF scores is implemented. The implementation is performed analogue to the one of the multi-input SVC (Section 5.3.1.).

5.3.5. Convolutional Neural Network

Keras' Sequential model[15] is used to implement several different architectures of CNNs for text classification.

In a first step, a model which receives the TF-IDF vectors of training examples and consists of three one dimensional convolutional layers[16], one dropout layer[17], one flatten[18] and one dense layer[19] is implemented. The first convolutional layer has been implemented with 64 filters while 48 where chosen for the subsequent two. A kernel size of five was chosen for all three convolutional layers. Dropout is a technique that is used to prevent the model from overfitting to the training data. A subset of neurons and the connections between them is randomly chosen and excluded from training during each iteration [34]. The sigmoid function (equation 4.6) is chosen as the model's activation function. The model's architecture is visualized in Figure 5.1.

In a second step, the model described above has been extended by an embedding layer[20] and thus receives the preprocessed data rather than the clauses' TF-IDF vectors as input. For more information on word embeddings see Section 4.2.2. The model's architecture is visualized in Figure 5.2.

Lastly, the model displayed in Figure 5.1. has been extended to receive both a clause's TF-IDF vector as well as a pretrained SVC's estimate of its assigned level 1 labels as inputs. The model's architecture is visualized in Figure 5.3.

[15]https://keras.io/guides/sequential_model/
[16]https://keras.io/api/layers/convolution_layers/convolution1d/
[17]https://keras.io/api/layers/regularization_layers/dropout/
[18]https://keras.io/api/layers/reshaping_layers/flatten/
[19]https://keras.io/api/layers/core_layers/dense/
[20]https://keras.io/api/layers/core_layers/embedding/

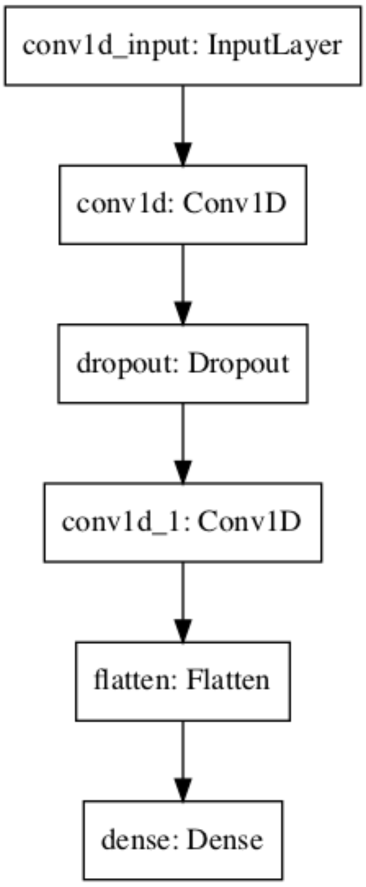

Figure 5.1.: CNN architecture

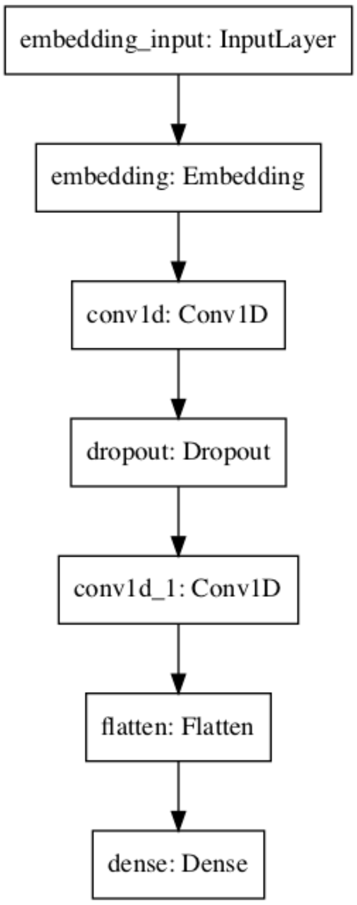

Figure 5.2.: CNN with embedding Layer architecture

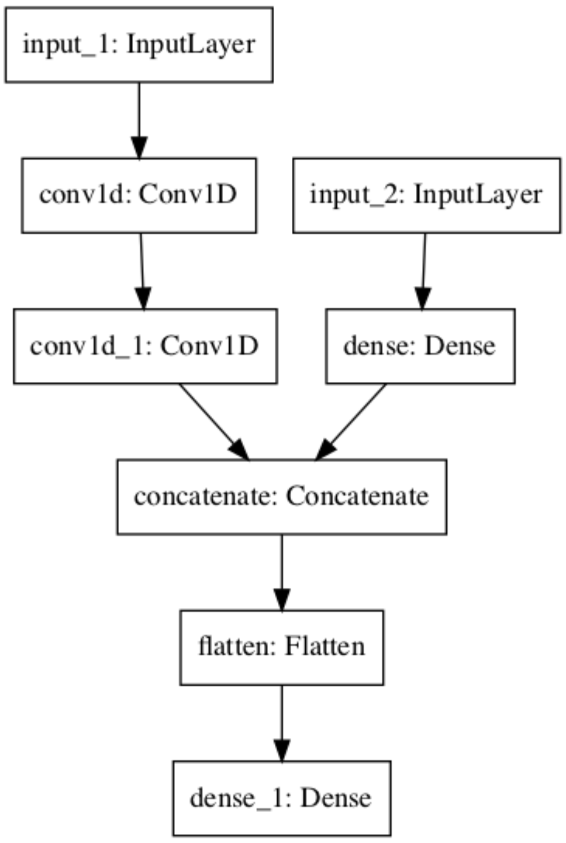

Figure 5.3.: Multi-input CNN architecture

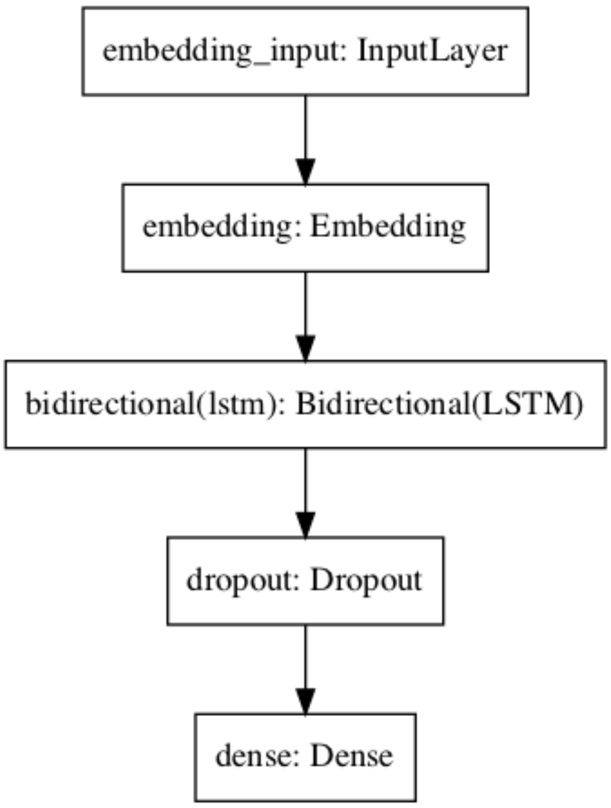
embedding_input: InputLayer
embedding: Embedding
bidirectional(lstm): Bidirectional(LSTM)
dropout: Dropout
dense: Dense

Figure 5.4.: LSTM with ambedding layer architecture

5.3.6. Long Short-Term Memory

A LSTM model is implemented with Keras' Sequential model[21]. The model consists of an embedding layer[22], a bidirectional LSTM layer[23], a dropout layer[24] and finally a dense layer[25]. A bidirectional RNN passes each data example to two separate RNNs. One of them receives the sequence as it is while the other receives the same information backwards. Both then pass the information on to the same output layer. The overarching architecture is thereby aware of information that lays before and after the current data point in a given sequence [15]. The model's architecture is visualized in Figure 5.4.

[21]https://keras.io/guides/sequential_model/
[22]https://keras.io/api/layers/core_layers/embedding/
[23]https://keras.io/api/layers/recurrent_layers/bidirectional/
[24]https://keras.io/api/layers/regularization_layers/dropout/
[25]https://keras.io/api/layers/core_layers/dense/

6. Results

6.1. Evaluation Metrics

In evaluating the results of different classification algorithms, it is vital to consider the various available metrics, their expressiveness and possible shortcomings. The evaluation metrics are calculated from

- true positives, i.e. the data points that were correctly assigned to a given class,
- true negatives, i.e. the data points that were correctly not assigned to a given class,
- false positives, i.e. the data points that were falsely assigned to a given class
- and false negatives, i.e. the data points that were not assigned to a given class they actually belong to [20].

Figure 6.1. illustrates true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) in a so-called *confusion matrix*. In order to account for the fact that the classification problem considered in this project is characterized by multiple classes, micro-averaging of the evaluation metrics is used. In contrast to macro-averaging which results in an average over classes, micro-averaging results in an average per data point by considering the decisions made for the particular clause over all classes [18].

6.1.1. Accuracy

The accuracy of a classification is a simple metric that considers the share of correctly classified data points in the entire data set (equation 6.1). Altough a classifier's accuracy is easy to interpret, its expressiveness is severely impaired in case of an unbalanced data set [20].

$$accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{6.1}$$

6.1.2. Recall

The recall of a classification is defined as the share of data points that were correctly assigned to a class of all data points in the particular class (equation 6.2) [18]. A classifier's recall can offer useful insights if the particular class has few representatives in the data set. A classifier simply assigning no data point to the class would achieve a high accuracy but might be of little use. Yet, assigning every data point to the given class would result in a high recall since the metric does not capture any information about TNs or FPs [20]. Additional information is therefore required.

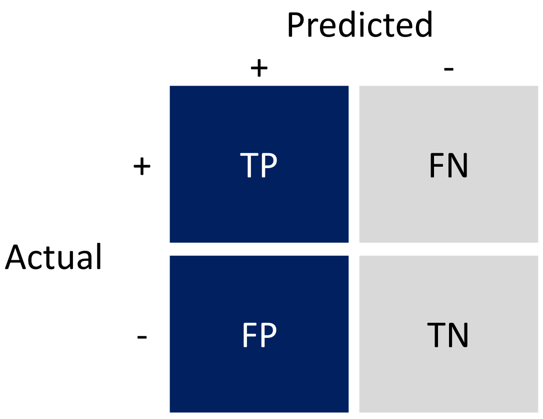

Figure 6.1.: Confusion Matrix
Source: own illustration based on [20]

$$recall = \frac{TP}{(TP + FN)} \tag{6.2}$$

6.1.3. Precision

Precision describes the share of data points that were correctly assigned to a class of all data points assigned to the given class (equation 6.3) [18]. The precision metric, however, is characterized by a shortcoming similar to the one of the recall metric. Since it does not consider TNs or FNs in any way, a classifier that assigns only few examples with high certainty to the particular class would also achieve a high precision [20].

$$precision = \frac{TP}{(TP + FP)} \tag{6.3}$$

6.1.4. F_1-Score

The F_1-score tries to make up for the shortcomings described above by taking both the recall as well as the precision of a classification algorithm into account and balancing the two metrics. It is defined by equation 6.4 [20].

$$F_1 - score = \frac{2TP}{(2TP + FP + FN)} \tag{6.4}$$

6.2. Discussion and Evaluation of Results

In order to establish a baseline, four classifiers, an SVC, an LR, an MLkNN and an MLP classifier, were trained on a first version of the corpus containing 3011 clauses (corpus version 1 hereafter) to predict their level 1 labels. Initially, the models only received a clause's information, i.e. its title and text, as input. The models were subsequently again trained with both a clause's information as well as the information of the paragraph to which it belongs. All four classifiers are performing remarkably well even when receiving only the clauses' information for training and prediction (results shown in Table 6.1.). However, the results can be severely improved by using the particular paragraph's information as additional input (results shown in Table 6.2.). While providing the paragraph's information has little impact on the SVC's precision, it significantly improves its recall which indicates that it more confidently assigns additional labels to a clause. Moreover, the larger input causes a rapid reduction in the MLkNN's false positives, evident in its increase in precision. The F_1-scores of all classifiers achieved on the test set improve by between 1.7 to 3.4 percentage points. One possible explanation might be that a proportion of the clauses implicitly refer to the ones coming before it in the particular paragraph. A clause may thus not mention terms that are key for identifying its topic explicitly. This can be remedied by providing the required context in form of the paragraph's information. Both a clause's as well as the corresponding paragraph's information are therefore used as input for the remainder of the project.

Classifier	F_1-Score	Accuracy	Precision	Recall
SVC	0.879	0.794	0.905	0.853
Logistic Regression	0.729	0.534	0.649	0.832
MLkNN	0.821	0.75	0.858	0.787
MLP	0.872	0.794	0.922	0.826

Table 6.1.: Classifiers trained on version 1 corpus using clause information as input to predict level 1 labels - results on test set

Classifier	F_1-Score	Accuracy	Precision	Recall
SVC	0.903	0.837	0.908	0.897
Logistic Regression	0.763	0.572	0.687	0.858
MLkNN	0.852	0.779	0.926	0.79
MLP	0.889	0.826	0.932	0.85

Table 6.2.: Classifiers trained on version 1 corpus using clause and paragraph information as input to predict level 1 labels - results on test set

The results obtained on the test set by the SVC are characterized by a high precision as well as recall resulting in an equally high F_1-score.
The LR classifier, however, performs poorly in terms of precision compared to its recall which indicates a larger number of false positives and thus misclassified clauses. The

result is a comparably low F_1-score. The classifier appears to correctly identify a large proportion of the classes a particular clause belongs to as is indicated by its recall. It may, however, assign too many labels to a given clause. The LR classifier, thus, correctly identifies the majority of labels but also assigns a large number of incorrect labels in the process. The MLkNN classifier's results show the opposite dynamic: precision scores that are considerably higher than its recall. The classifier, thus, appears to assign fewer labels resulting in fewer false positives but also fewer correctly identified classes.
Similar to the SVC, the MLP classifier performs equally well in terms of both precision and recall. Its results are, however, slightly less balanced as its precision is significantly higher than its recall resulting in a moderately lower F_1-score.

Classifier	F_1-Score	Accuracy	Precision	Recall
SVC	0.904	0.839	0.921	0.889
Logistic Regression	0.815	0.655	0.745	0.899
MLkNN	0.844	0.768	0.873	0.817
MLP	0.895	0.82	0.91	0.881
CNN	0.867	0.773	0.908	0.83
CNN Embedding Layer	0.568	0.468	0.583	0.553
LSTM	0.861	0.791	0.882	0.84

Table 6.3.: Classifiers trained on version 2 corpus using clause and paragraph information as input to predict level 1 labels - results on test set

After the labeling process has been completed, the above-mentioned classifiers as well as a CNN, a CNN with an embedding layer and a LSTM (see Sections 5.3.5. and 5.3.6. for details on the models' architectures) were trained and examined on the final corpus containing 5020 clauses (version 2 corpus hereafter) and their level 1 labels. The results (shown in Table 6.3.) display the same characteristics as the ones described above.
The performance of the MLP classifier, however, appears to be more balanced after presenting it the additional data. Despite its slightly decreasing precision, its F_1-score is consequently improving.
Although the MLkNN's recall slightly improves, its precision severely decreases despite it being trained on a significantly larger data set, resulting in a lower F_1-score.
While the results of the LR classifier show a significant improvement through the additional data, the ones of the remaining three change only marginally. This may indicate that the majority of the clauses within a class is rather homogeneous in regard to their wording. Firstly, since a terms of service agreement regulates the general conditions of a sales contract, the majority of its paragraphs explicitly mention certain words indicating their topic, where against their semantic meaning can be ambiguous. A clause stipulating a customer's right to withdraw from the contract, for instance, most likely contains the word *withdrawal*. Secondly, the TF-IDF technique used to create the features the models are trained with disregards semantic meaning and only considers the set of words occurring in the particular clause. A comparably small amount of data points may thus be enough to sufficiently train the described algorithms to identify a clause's topic.
Both the CNN and the LSTM perform rather well as is indicated by their similarly high

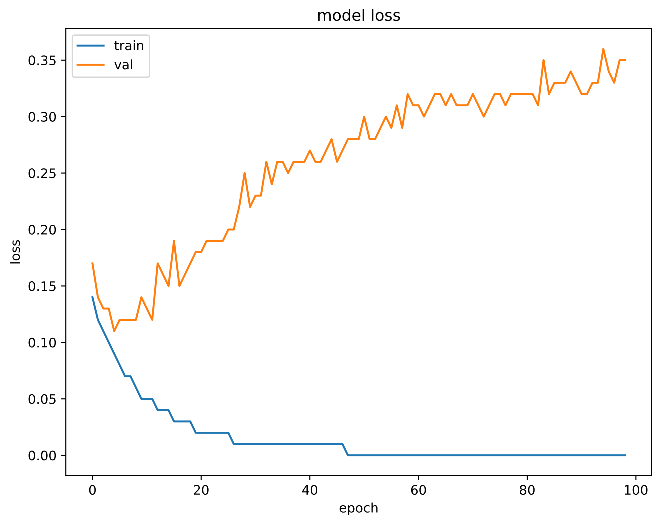

Figure 6.2.: CNN embedding layer trained on corpus version 2 to predict level 1 labels - loss during training process

F_1-scores. The CNN that receives the preprocessed clauses as inputs and then passes them through an embedding layer, however, produces extremely poor results despite being able to accurately classify the greater share of the training data (0.993 F_1-score). This suggests that the model overfits to the training data and fails to generalize its results to previously unknown clauses, in spite of the use of a substantial dropout. Figure 6.2. illustrates the model's loss on the training as well as the validation data during the training process. The loss on the training data is steadily declining while the one received for the validation set is quickly increasing again which is as another indicator for a model overfitting to the training data [12].

The models were subsequently trained on the version 2 corpus in order to predict their level 2 labels (results shown in Table 6.4.). The LR classifier's performance for level 2 predictions appears to be significantly more balanced than the one for level 1 predictions. While its recall is severely lower, it exhibits a slightly higher precision. As was the case described above, the MLkNN classifier's performance is characterized by a considerably higher precision than recall. Moreover, it shows comparably little ability to correctly classify the clauses in the training set (0.856 F_1-score). One explanation may lie in the fact that kNN is generally sensitive to noise in the data and thus prone to overestimate irrelevant features [17]. The SVC performs well in predicting a clause's level 2 label which is indicated by its F_1-score. Its results are, however, severely more unbalanced than was the case for the level 1 prediction. The same is true for the MLP as its precision is significantly higher than its recall.
Additionally, both the CNN as well as the LSTM also show a severe discrepancy betweeen their precision and recall. The LSTM offers the starkest contrast with a precision that is 16.6 percentage points higher than its recall. Table 6.7. shows the LSTM's average results per

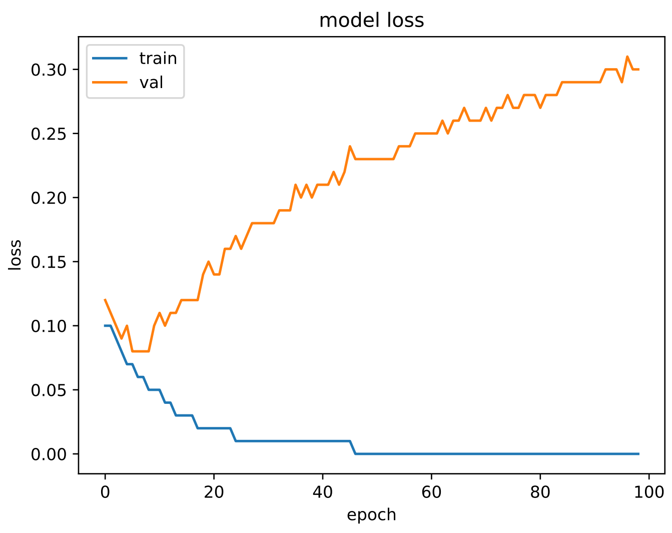

Figure 6.3.: CNN embedding layer trained on corpus version 2 to predict level 2 labels - loss during training process

level 2 class. The sharp difference between its precision and recall is even more apparent in its results for individual classes. Its precision for classes such as *delivery:acceptance*, *payment:nettingAndWithholding*, *userAccount*, *withdrawal:exclusion* and *withdrawal:form* is 1.0, meaning that there have been no false positives for any of those classes. The recall achieved for them, however, was between 37.5 and 72.7 percentage points lower. Apart from that, the LSTM appears to perform rather poorly in terms of both precision as well as recall for some frequently occurring classes such as *delivery:costs* or *dataProtectionAndProcessing*. LSTMs were found to work less well in classification tasks where the decision hinges on the recognition of some key phrases [38]. This may explain the results shown here since the correct classification of a clause potentially relies less on the structure of a sentence but rather the occurrence of some key terms. Yet, the LSTM was able to correctly classify the vast majority of training examples (0.967 F_1-score) which may also be an indicator of overfitting. For the model's loss during training see Appendix A.2. As was the case in the prediction of level 1 labels, the CNN whose architecture includes an embedding layer seems to yield a rather mediocre performance despite being able to correctly classify the examples of the training set to a large degree (0.982 F_1-score). It appears that it, too, is overfitting to the training data which is also indicated by its loss during the training process (Figure 6.3.).

Furthermore, to take advantage of the hierarchy of the classes the data was divided into, an SVC, an MLP and a CNN also received a previously trained SVC's prediction for a clause's level 1 label as input. For the multi-input SVC as well as the multi-input MLP, the clause's length was also used as a feature. A level 1 estimation would provide helpful information if a model were to regularly place clauses in another level 1 class instead of merely selecting the wrong level 2 label within the correct level 1 class. A classifier might, for instance, mistake a clause labeled *payment:costs* for one belonging to *delivery:costs* rather

Classifier	F_1-Score	Accuracy	Precision	Recall
SVC	0.834	0.706	0.805	0.866
Multi-input SVC	0.842	0.727	0.865	0.82
Logistic Regression	0.783	0.601	0.769	0.798
MLkNN	0.775	0.652	0.83	0.727
MLP	0.827	0.704	0.861	0.794
Multi-input MLP	0.837	0.708	0.863	0.812
CNN	0.791	0.643	0.854	0.736
CNN Embedding Layer	0.47	0.352	0.635	0.373
Multi-input CNN	0.82	0.695	0.878	0.769
LSTM	0.768	0.642	0.86	0.694

Table 6.4.: Classifiers trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

than say one belonging to *payment:method*. Moreover, a clause's length may help to distinguish between classes but also indicate when a clause received multiple labels.
Training sperate classifiers at each node of the classification hierarchy is another commonly used method for hierarchical text classification. A chain of models classifies a document in a top-down manner. So-called *subtree classifiers* form a decision about the document at every node of the category tree until a so-called *local classifier* at one of the tree's leaves performs the final classification [35]. Due to the limited size of the corpus as well as the multi-label nature of the given classification problem, this approach has not been implemented in this project.
While providing multiple inputs had little effect on the MLP classifier's precision and accuracy, it did slightly improve its recall. This may indicate that providing a clause's length led the classifier to assign more labels and thus correctly identify more class representatives. Contradicting that argument, however, is the opposite development shown by the multi-input SVC. Its precision is considerably higher than the one of the SVC receiving only a clause's TF-IDF scores. Its recall on the other hand is lower by almost the same magnitude. Both the SVC's as well as the MLP's F_1-score slightly improves by providing multiple inputs. It is, nevertheless, not possible to clearly ascertain which additional feature is responsible or why. Comparing the classifiers' result per level 2 class (shown in Tables 6.5. and 6.6.) does not allow for a clear conclusion either. While the results for a vast number of classes only change marginally, some show a significant alteration through the provision of multiple inputs. The results for some classes, for instance *payment:costs*, improve drastically. The ones for others, however, are significantly worse. The F_1-scores for the classes *payment:creditRating* and *userAccount* for example, fell from 0.786 and 0.588 to 0.692 and 0.308 respectively.
The multi-input CNN which receives a prediction for a clause's level 1 label apart from a clause's TF-IDF scores is somewhat better performing than the CNN only trained on a clause's TF-IDF scores. For a detailed listing of the remaining classifier's performances on each level 2 class see Appendix A.1.

As a concluding remark, it can be noted that the SVC and the MLP and its variations consistently show the best results of the evaluated algorithms. Neural Networks and SVMs generally appear to perform especially well when the input-output relationship of the given problem is a non-linear one and the input features are highly correlated [17]. Both the CNN as well as the LSTM show promising results in both level 1 as well as level 2 predictions but perform considerably worse than the SVC and the MLP classifiers do. The CNN containing an embedding layer and also the LSTM, though to a much lesser degree, appear to overfit to the training data. The performance of such deep learning models may, thus, considerably improve if further measures to prevent overfitting are taken. Additionally, optimizing their architecture and hyperparameters to the given task could also significantly enhance their ability to identify a clause's topic [38].

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	1.0	1.0	1.0	9.0
choiceOfLaw	0.96	0.923	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.286	0.2	0.5	2.0
contractLanguage	0.93	0.87	1.0	20.0
contractObject	0.783	1.0	0.643	14.0
contractStorage	0.977	1.0	0.955	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.556	0.5	0.625	8.0
dataProtectionAndProcessing	0.745	0.679	0.826	23.0
definition:consumer	0.97	0.941	1.0	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.727	1.0	0.571	7.0
delivery:collection	0.88	0.786	1.0	11.0
delivery:costs	0.769	0.667	0.909	44.0
delivery:destination	0.809	0.731	0.905	21.0
delivery:inspectionAndDamages	0.889	0.96	0.828	29.0
delivery:liability	0.941	0.941	0.941	17.0
delivery:method	0.629	0.917	0.478	23.0
delivery:partialDeliveries	0.615	0.5	0.8	5.0
delivery:productAvailability	0.732	0.6	0.938	16.0
delivery:time	0.758	0.735	0.781	32.0
disputeResolution	0.971	0.971	0.971	35.0
intellectualProperty	0.778	0.636	1.0	7.0
liabilityScope	0.84	0.778	0.913	46.0
orderRestrictions	0.4	0.4	0.4	5.0
party	0.769	0.714	0.833	36.0
payment:costs	0.533	0.8	0.4	10.0
payment:creditRating	0.786	0.688	0.917	12.0
payment:default	0.75	0.706	0.8	15.0

payment:invoice	0.222	0.125	1.0	1.0
payment:method	0.821	0.744	0.914	70.0
payment:nettingAndWithholding	0.941	1.0	0.889	9.0
payment:time	0.737	0.724	0.75	28.0
placeOfFulfillment	0.75	0.75	0.75	4.0
placeOfJurisdiction	0.941	0.889	1.0	16.0
prices	0.878	0.837	0.923	39.0
realization:offerAndAcceptance	0.916	0.874	0.962	79.0
realization:orderProcess	0.875	0.921	0.833	42.0
retentionOfTitle	0.979	0.979	0.979	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.933	0.933	0.933	45.0
sellerWithdrawalRight	0.696	0.615	0.8	10.0
userAccount	0.588	0.833	0.455	11.0
vouchersDiscountsPromotionsGiftcards	0.939	0.982	0.9	60.0
warranty:contractualClaims	0.693	0.614	0.795	44.0
warranty:exclusion	0.533	0.667	0.444	18.0
warranty:lapse	0.868	0.846	0.892	37.0
warranty:legalClaims	0.754	0.657	0.885	26.0
withdrawal:consequences	0.8	0.711	0.914	35.0
withdrawal:exclusion	0.788	0.929	0.684	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.889	0.889	0.889	45.0

Table 6.5.: SVC trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	1.0	1.0	1.0	9.0
choiceOfLaw	0.98	0.96	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.5	0.5	0.5	2.0
contractLanguage	0.976	0.952	1.0	20.0
contractObject	0.727	1.0	0.571	14.0
contractStorage	0.952	1.0	0.909	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.615	0.8	0.5	8.0
dataProtectionAndProcessing	0.783	0.783	0.783	23.0
definition:consumer	1.0	1.0	1.0	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0

definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.727	1.0	0.571	7.0
delivery:collection	0.88	0.786	1.0	11.0
delivery:costs	0.784	0.717	0.864	44.0
delivery:destination	0.762	0.762	0.762	21.0
delivery:inspectionAndDamages	0.889	0.96	0.828	29.0
delivery:liability	0.941	0.941	0.941	17.0
delivery:method	0.588	0.909	0.435	23.0
delivery:partialDeliveries	0.727	0.667	0.8	5.0
delivery:productAvailability	0.75	0.75	0.75	16.0
delivery:time	0.721	0.759	0.688	32.0
disputeResolution	0.957	0.971	0.943	35.0
intellectualProperty	0.8	0.75	0.857	7.0
liabilityScope	0.899	0.93	0.87	46.0
orderRestrictions	0.5	0.667	0.4	5.0
party	0.781	0.893	0.694	36.0
payment:costs	0.706	0.857	0.6	10.0
payment:creditRating	0.692	0.643	0.75	12.0
payment:default	0.741	0.833	0.667	15.0
payment:invoice	0.333	0.2	1.0	1.0
payment:method	0.841	0.813	0.871	70.0
payment:nettingAndWithholding	0.875	1.0	0.778	9.0
payment:time	0.717	0.76	0.679	28.0
placeOfFulfillment	0.857	1.0	0.75	4.0
placeOfJurisdiction	0.941	0.889	1.0	16.0
prices	0.892	0.943	0.846	39.0
realization:offerAndAcceptance	0.942	0.961	0.924	79.0
realization:orderProcess	0.861	0.919	0.81	42.0
retentionOfTitle	0.945	0.977	0.915	47.0
rightToRefuse	0.667	0.5	1.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.943	0.976	0.911	45.0
sellerWithdrawalRight	0.588	0.714	0.5	10.0
userAccount	0.308	1.0	0.182	11.0
vouchersDiscountsPromotionsGiftcards	0.947	1.0	0.9	60.0
warranty:contractualClaims	0.703	0.681	0.727	44.0
warranty:exclusion	0.533	0.667	0.444	18.0
warranty:lapse	0.853	0.842	0.865	37.0
warranty:legalClaims	0.712	0.636	0.808	26.0
withdrawal:consequences	0.789	0.732	0.857	35.0
withdrawal:exclusion	0.765	0.867	0.684	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.92	0.952	0.889	45.0

Table 6.6.: Multi-input SVC trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	0.941	1.0	0.889	9.0
choiceOfLaw	0.941	0.889	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.0	0.0	0.0	2.0
contractLanguage	0.923	0.947	0.9	20.0
contractObject	0.522	0.667	0.429	14.0
contractStorage	0.909	0.909	0.909	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.769	1.0	0.625	8.0
dataProtectionAndProcessing	0.667	0.64	0.696	23.0
definition:consumer	0.968	1.0	0.938	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.828	0.923	0.75	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.6	1.0	0.429	7.0
delivery:collection	0.952	1.0	0.909	11.0
delivery:costs	0.605	0.619	0.591	44.0
delivery:destination	0.634	0.65	0.619	21.0
delivery:inspectionAndDamages	0.863	1.0	0.759	29.0
delivery:liability	0.903	1.0	0.824	17.0
delivery:method	0.537	0.611	0.478	23.0
delivery:partialDeliveries	0.333	1.0	0.2	5.0
delivery:productAvailability	0.522	0.857	0.375	16.0
delivery:time	0.596	0.933	0.438	32.0
disputeResolution	0.941	0.97	0.914	35.0
intellectualProperty	0.333	0.4	0.286	7.0
liabilityScope	0.894	0.974	0.826	46.0
orderRestrictions	0.333	1.0	0.2	5.0
party	0.667	0.905	0.528	36.0
payment:costs	0.462	1.0	0.3	10.0
payment:creditRating	0.692	0.643	0.75	12.0
payment:default	0.235	1.0	0.133	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.809	0.833	0.786	70.0
payment:nettingAndWithholding	0.714	1.0	0.556	9.0
payment:time	0.667	0.8	0.571	28.0
placeOfFulfillment	0.4	1.0	0.25	4.0

placeOfJurisdiction	0.839	0.867	0.812	16.0
prices	0.845	0.938	0.769	39.0
realization:offerAndAcceptance	0.826	0.966	0.722	79.0
realization:orderProcess	0.714	0.714	0.714	42.0
retentionOfTitle	0.92	1.0	0.851	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	1.0	1.0	1.0	9.0
scope	0.907	0.951	0.867	45.0
sellerWithdrawalRight	0.4	0.6	0.3	10.0
userAccount	0.429	1.0	0.273	11.0
vouchersDiscountsPromotionsGiftcards	0.957	1.0	0.917	60.0
warranty:contractualClaims	0.675	0.75	0.614	44.0
warranty:exclusion	0.357	0.5	0.278	18.0
warranty:lapse	0.677	0.84	0.568	37.0
warranty:legalClaims	0.679	0.667	0.692	26.0
withdrawal:consequences	0.704	0.694	0.714	35.0
withdrawal:exclusion	0.538	1.0	0.368	19.0
withdrawal:form	0.769	1.0	0.625	8.0
withdrawal:right	0.907	0.951	0.867	45.0

Table 6.7.: LSTM trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

7. Conclusion

The SVC as well as the MLP perform remarkably well in identifying a clause's level 1 class, even when trained on comparably little data and receiving only the clause's title and text as input. The results can, however, still be improved by additionally providing the title and text of the paragraph a particular clause belongs to. This indicates that the paragraph's information can in some cases provide additional context and may contain key terms not already present in the clause itself.
Increasing the corpus from 3011 to 5020 clauses, nevertheless, had little impact on the performance of the majority of classifiers. This may suggest that the greater proportion of clauses within a class are fairly homogeneous in terms of their terminology. Both the CNN as well as the LSTM are to a large degree able to correctly identify the level 1 labels of the test examples. Their performance in predicting level 2 labels, however, is significantly less balanced. Their precision is much higher than their recall resulting in a lower F_1-score. The CNN containing an embedding layer appears to overfit to the training data for the level 1 and level 2 classifications.
Additionally, providing a clause's length as well as an estimate of its level 1 labels leads to a slight improvement of the SVC's and the MLP's overall results. But it remains unclear which of the supplementary inputs is responsible and for which reasons. Comparing the SVC's and the multi-input SVC's results per class does not allow for a clear conclusion either. While its performance for some classes improves, the one for others appears to worsen through the provision of the additional inputs. The majority of the results, however, remain unchanged. Although the CNN's ability to predict a clause's level 2 labels also improves by using an estimate of its level 1 label as a supplementary feature, the SVC and MLP perform significantly better throughout the entire project.

Possible next steps beyond the scope of the current project that could be taken include improving the model's performance on the text classification, adapting the categorization to enhance its potential usefulness and making use of the topic classification in other applications.
One way to potentially improve the classification's results is to use a pretrained word embedding model during the feature engineering. Fasttext for instance, a commonly used word embedding framework developed by Facebook's AI Research lab [6], includes models trained on Wikipedia[1] and Common Crawl[2] for 157 different languages[3]. Since the CNN which uses an embedding layer to extract features from the corpus appears to overfit to the training data in both level 1 as well as level 2 classifications, further measures to prevent overfitting may be necessary if a pretrained word embedding were to be used. Apart from that, Deep Neural Networks tend to be sensitive to the choice of hidden sizes

[1]https://www.wikipedia.org
[2]https://commoncrawl.org
[3]https://fasttext.cc/docs/en/crawl-vectors.html

as well as batch sizes [38]. Further optimizing the CNN's and the LSTM's hyperparameters and architectures might, therefore, severely increase their performance.
Additionally, very deep model architectures similar to the ones used in Computer Vision, processing the input on a character rather than word level, seem to also perform extremely well in text classification applications [8]. Increasing the models' complexity, however, bears the risk of overfitting with a comparably small corpus such as the one being used in this project [12]. It might, thus, also be beneficial to further enlarge the corpus which may also enable the use of more elaborate deep learning model architectures.
Another way to possibly improve performance is to address the potentially negative effects of the severely unbalanced corpus of the classification problem at hand, e.g. by oversampling classes with fewer representatives. This may in turn, however, also increase the model's propensity to overfit [13].
Moreover, further approaches to hierarchical text classification could be investigated to make full use of the classes' hierarchical structure.
Apart from improving the model's performance, the corpus could also be adapted to make an even more granular differentiation between clauses depending on the particular use case. The scope of the classification could also be enhanced by including terms of service agreements in languages other than German in the corpus.
Lastly, the topic classification's results obtained during the project could be used in more advanced applications, possibly after the steps described above have been investigated, for instance to make some sort of further qualitative assessment of a clause that goes beyond its topic.

A. Appendix

A.1. Detailed Results Level 2 Predictions

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	0.941	1.0	0.889	9.0
choiceOfLaw	0.98	0.96	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.0	0.0	0.0	2.0
contractLanguage	0.976	0.952	1.0	20.0
contractObject	0.696	0.889	0.571	14.0
contractStorage	0.977	1.0	0.955	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.667	1.0	0.5	8.0
dataProtectionAndProcessing	0.681	0.667	0.696	23.0
definition:consumer	0.968	1.0	0.938	16.0
definition:customer	0.4	0.5	0.333	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.667	0.8	0.571	7.0
delivery:collection	0.909	0.909	0.909	11.0
delivery:costs	0.755	0.685	0.841	44.0
delivery:destination	0.737	0.824	0.667	21.0
delivery:inspectionAndDamages	0.868	0.958	0.793	29.0
delivery:liability	0.857	0.833	0.882	17.0
delivery:method	0.615	0.75	0.522	23.0
delivery:partialDeliveries	0.571	1.0	0.4	5.0
delivery:productAvailability	0.765	0.722	0.812	16.0
delivery:time	0.741	0.909	0.625	32.0
disputeResolution	0.957	0.971	0.943	35.0
intellectualProperty	0.714	0.714	0.714	7.0
liabilityScope	0.891	0.891	0.891	46.0
orderRestrictions	0.222	0.25	0.2	5.0
party	0.769	0.862	0.694	36.0
payment:costs	0.421	0.444	0.4	10.0
payment:creditRating	0.8	0.769	0.833	12.0
payment:default	0.609	0.875	0.467	15.0
payment:invoice	0.0	0.0	0.0	1.0

payment:method	0.811	0.795	0.829	70.0
payment:nettingAndWithholding	0.875	1.0	0.778	9.0
payment:time	0.654	0.708	0.607	28.0
placeOfFulfillment	0.75	0.75	0.75	4.0
placeOfJurisdiction	0.909	0.882	0.938	16.0
prices	0.861	0.939	0.795	39.0
realization:offerAndAcceptance	0.942	0.961	0.924	79.0
realization:orderProcess	0.819	0.829	0.81	42.0
retentionOfTitle	0.968	0.978	0.957	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.918	0.975	0.867	45.0
sellerWithdrawalRight	0.667	0.636	0.7	10.0
userAccount	0.706	1.0	0.545	11.0
vouchersDiscountsPromotionsGiftcards	0.931	0.964	0.9	60.0
warranty:contractualClaims	0.659	0.711	0.614	44.0
warranty:exclusion	0.545	0.6	0.5	18.0
warranty:lapse	0.824	0.903	0.757	37.0
warranty:legalClaims	0.731	0.731	0.731	26.0
withdrawal:consequences	0.845	0.833	0.857	35.0
withdrawal:exclusion	0.765	0.867	0.684	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.871	0.925	0.822	45.0

Table A.1.: MLP trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	1.0	1.0	1.0	9.0
choiceOfLaw	0.98	0.96	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.667	1.0	0.5	2.0
contractLanguage	0.976	0.952	1.0	20.0
contractObject	0.667	0.8	0.571	14.0
contractStorage	0.952	1.0	0.909	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.615	0.8	0.5	8.0
dataProtectionAndProcessing	0.756	0.773	0.739	23.0
definition:consumer	0.968	1.0	0.938	16.0
definition:customer	0.4	0.5	0.333	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0

delivery:acceptance	0.769	0.833	0.714	7.0
delivery:collection	0.909	0.909	0.909	11.0
delivery:costs	0.78	0.696	0.886	44.0
delivery:destination	0.75	0.789	0.714	21.0
delivery:inspectionAndDamages	0.868	0.958	0.793	29.0
delivery:liability	0.857	0.833	0.882	17.0
delivery:method	0.667	0.812	0.565	23.0
delivery:partialDeliveries	0.75	1.0	0.6	5.0
delivery:productAvailability	0.812	0.812	0.812	16.0
delivery:time	0.731	0.95	0.594	32.0
disputeResolution	0.971	0.971	0.971	35.0
intellectualProperty	0.714	0.714	0.714	7.0
liabilityScope	0.882	0.872	0.891	46.0
orderRestrictions	0.364	0.333	0.4	5.0
party	0.769	0.862	0.694	36.0
payment:costs	0.556	0.625	0.5	10.0
payment:creditRating	0.769	0.714	0.833	12.0
payment:default	0.636	1.0	0.467	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.803	0.792	0.814	70.0
payment:nettingAndWithholding	0.875	1.0	0.778	9.0
payment:time	0.679	0.72	0.643	28.0
placeOfFulfillment	0.857	1.0	0.75	4.0
placeOfJurisdiction	0.941	0.889	1.0	16.0
prices	0.892	0.943	0.846	39.0
realization:offerAndAcceptance	0.962	0.974	0.949	79.0
realization:orderProcess	0.81	0.81	0.81	42.0
retentionOfTitle	0.957	0.978	0.936	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.943	0.976	0.911	45.0
sellerWithdrawalRight	0.625	0.833	0.5	10.0
userAccount	0.706	1.0	0.545	11.0
vouchersDiscountsPromotionsGiftcards	0.939	0.982	0.9	60.0
warranty:contractualClaims	0.636	0.636	0.636	44.0
warranty:exclusion	0.606	0.667	0.556	18.0
warranty:lapse	0.845	0.882	0.811	37.0
warranty:legalClaims	0.746	0.667	0.846	26.0
withdrawal:consequences	0.822	0.789	0.857	35.0
withdrawal:exclusion	0.765	0.867	0.684	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.884	0.927	0.844	45.0

Table A.2.: Multi-input MLP trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	0.941	1.0	0.889	9.0
choiceOfLaw	0.936	0.957	0.917	24.0
codeOfConduct	0.9	1.0	0.818	11.0
contractChange	0.0	0.0	0.0	2.0
contractLanguage	0.952	0.909	1.0	20.0
contractObject	0.769	0.833	0.714	14.0
contractStorage	0.875	0.808	0.955	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.667	1.0	0.5	8.0
dataProtectionAndProcessing	0.727	0.625	0.87	23.0
definition:consumer	1.0	1.0	1.0	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.833	1.0	0.714	7.0
delivery:collection	0.87	0.833	0.909	11.0
delivery:costs	0.605	0.719	0.523	44.0
delivery:destination	0.562	0.818	0.429	21.0
delivery:inspectionAndDamages	0.857	0.889	0.828	29.0
delivery:liability	0.848	0.875	0.824	17.0
delivery:method	0.65	0.765	0.565	23.0
delivery:partialDeliveries	0.889	1.0	0.8	5.0
delivery:productAvailability	0.69	0.769	0.625	16.0
delivery:time	0.586	0.654	0.531	32.0
disputeResolution	0.971	0.971	0.971	35.0
intellectualProperty	0.625	0.556	0.714	7.0
liabilityScope	0.804	0.804	0.804	46.0
orderRestrictions	0.286	0.5	0.2	5.0
party	0.725	0.758	0.694	36.0
payment:costs	0.375	0.5	0.3	10.0
payment:creditRating	0.583	0.583	0.583	12.0
payment:default	0.522	0.75	0.4	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.763	0.82	0.714	70.0
payment:nettingAndWithholding	0.824	0.875	0.778	9.0
payment:time	0.706	0.783	0.643	28.0
placeOfFulfillment	0.571	0.667	0.5	4.0

placeOfJurisdiction	0.914	0.842	1.0	16.0
prices	0.775	0.756	0.795	39.0
realization:offerAndAcceptance	0.882	0.918	0.848	79.0
realization:orderProcess	0.778	0.933	0.667	42.0
retentionOfTitle	0.92	1.0	0.851	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	1.0	1.0	1.0	9.0
scope	0.86	0.902	0.822	45.0
sellerWithdrawalRight	0.444	0.5	0.4	10.0
userAccount	0.667	0.857	0.545	11.0
vouchersDiscountsPromotionsGiftcards	0.947	1.0	0.9	60.0
warranty:contractualClaims	0.569	0.5	0.659	44.0
warranty:exclusion	0.444	0.667	0.333	18.0
warranty:lapse	0.644	0.864	0.514	37.0
warranty:legalClaims	0.65	0.929	0.5	26.0
withdrawal:consequences	0.806	0.844	0.771	35.0
withdrawal:exclusion	0.571	0.889	0.421	19.0
withdrawal:form	0.857	1.0	0.75	8.0
withdrawal:right	0.833	0.897	0.778	45.0

Table A.3.: MLkNN trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	0.8	1.0	0.667	9.0
choiceOfLaw	0.939	0.92	0.958	24.0
codeOfConduct	0.9	1.0	0.818	11.0
contractChange	0.0	0.0	0.0	2.0
contractLanguage	0.976	0.952	1.0	20.0
contractObject	0.64	0.727	0.571	14.0
contractStorage	0.952	1.0	0.909	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.545	1.0	0.375	8.0
dataProtectionAndProcessing	0.8	0.741	0.87	23.0
definition:consumer	0.941	0.889	1.0	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.727	1.0	0.571	7.0
delivery:collection	0.9	1.0	0.818	11.0
delivery:costs	0.757	0.627	0.955	44.0
delivery:destination	0.684	0.765	0.619	21.0

delivery:inspectionAndDamages	0.889	0.96	0.828	29.0
delivery:liability	0.895	0.81	1.0	17.0
delivery:method	0.629	0.917	0.478	23.0
delivery:partialDeliveries	0.571	1.0	0.4	5.0
delivery:productAvailability	0.815	1.0	0.688	16.0
delivery:time	0.618	0.739	0.531	32.0
disputeResolution	0.971	0.971	0.971	35.0
intellectualProperty	0.25	1.0	0.143	7.0
liabilityScope	0.845	0.804	0.891	46.0
orderRestrictions	0.333	1.0	0.2	5.0
party	0.698	0.815	0.611	36.0
payment:costs	0.5	0.667	0.4	10.0
payment:creditRating	0.69	0.588	0.833	12.0
payment:default	0.125	1.0	0.067	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.793	0.677	0.957	70.0
payment:nettingAndWithholding	0.875	1.0	0.778	9.0
payment:time	0.645	0.588	0.714	28.0
placeOfFulfillment	0.4	1.0	0.25	4.0
placeOfJurisdiction	0.882	0.833	0.938	16.0
prices	0.829	0.791	0.872	39.0
realization:offerAndAcceptance	0.837	0.733	0.975	79.0
realization:orderProcess	0.795	0.761	0.833	42.0
retentionOfTitle	0.968	0.978	0.957	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.891	0.872	0.911	45.0
sellerWithdrawalRight	0.533	0.8	0.4	10.0
userAccount	0.0	0.0	0.0	11.0
vouchersDiscountsPromotionsGiftcards	0.929	1.0	0.867	60.0
warranty:contractualClaims	0.569	0.458	0.75	44.0
warranty:exclusion	0.467	0.583	0.389	18.0
warranty:lapse	0.771	0.696	0.865	37.0
warranty:legalClaims	0.625	0.526	0.769	26.0
withdrawal:consequences	0.741	0.652	0.857	35.0
withdrawal:exclusion	0.7	0.667	0.737	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.787	0.755	0.822	45.0

Table A.4.: Logistic Regression trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support

batteryElectronicsAndPackaging	0.714	1.0	0.556	9.0
choiceOfLaw	0.829	1.0	0.708	24.0
codeOfConduct	0.957	0.917	1.0	11.0
contractChange	1.0	1.0	1.0	2.0
contractLanguage	0.952	0.909	1.0	20.0
contractObject	0.133	1.0	0.071	14.0
contractStorage	0.706	1.0	0.545	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.545	1.0	0.375	8.0
dataProtectionAndProcessing	0.76	0.704	0.826	23.0
definition:consumer	0.968	1.0	0.938	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.833	1.0	0.714	7.0
delivery:collection	0.909	0.909	0.909	11.0
delivery:costs	0.727	0.655	0.818	44.0
delivery:destination	0.174	1.0	0.095	21.0
delivery:inspectionAndDamages	0.929	0.963	0.897	29.0
delivery:liability	0.909	0.938	0.882	17.0
delivery:method	0.545	0.9	0.391	23.0
delivery:partialDeliveries	0.571	1.0	0.4	5.0
delivery:productAvailability	0.839	0.867	0.812	16.0
delivery:time	0.745	1.0	0.594	32.0
disputeResolution	0.957	0.971	0.943	35.0
intellectualProperty	0.778	0.636	1.0	7.0
liabilityScope	0.835	1.0	0.717	46.0
orderRestrictions	0.286	0.5	0.2	5.0
party	0.458	0.917	0.306	36.0
payment:costs	0.526	0.556	0.5	10.0
payment:creditRating	0.588	1.0	0.417	12.0
payment:default	0.636	1.0	0.467	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.785	0.85	0.729	70.0
payment:nettingAndWithholding	0.875	1.0	0.778	9.0
payment:time	0.739	0.944	0.607	28.0
placeOfFulfillment	0.667	1.0	0.5	4.0
placeOfJurisdiction	0.839	0.867	0.812	16.0
prices	0.847	0.783	0.923	39.0
realization:offerAndAcceptance	0.926	0.986	0.873	79.0
realization:orderProcess	0.864	0.826	0.905	42.0
retentionOfTitle	0.967	1.0	0.936	47.0

rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.864	0.972	0.778	45.0
sellerWithdrawalRight	0.333	1.0	0.2	10.0
userAccount	0.308	1.0	0.182	11.0
vouchersDiscountsPromotionsGiftcards	0.932	0.948	0.917	60.0
warranty:contractualClaims	0.714	0.962	0.568	44.0
warranty:exclusion	0.384	0.255	0.778	18.0
warranty:lapse	0.8	0.744	0.865	37.0
warranty:legalClaims	0.607	0.567	0.654	26.0
withdrawal:consequences	0.825	0.929	0.743	35.0
withdrawal:exclusion	0.813	1.0	0.684	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.911	0.911	0.911	45.0

Table A.5.: CNN trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	0.2	1.0	0.111	9.0
choiceOfLaw	0.619	0.722	0.542	24.0
codeOfConduct	0.7	0.778	0.636	11.0
contractChange	0.0	0.0	0.0	2.0
contractLanguage	0.78	0.762	0.8	20.0
contractObject	0.571	0.857	0.429	14.0
contractStorage	0.5	0.556	0.455	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.182	0.333	0.125	8.0
dataProtectionAndProcessing	0.05	0.059	0.043	23.0
definition:consumer	0.72	1.0	0.562	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.64	0.889	0.5	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.6	1.0	0.429	7.0
delivery:collection	0.667	0.857	0.545	11.0
delivery:costs	0.405	0.5	0.341	44.0
delivery:destination	0.0	0.0	0.0	21.0
delivery:inspectionAndDamages	0.565	0.765	0.448	29.0
delivery:liability	0.625	0.667	0.588	17.0
delivery:method	0.276	0.667	0.174	23.0
delivery:partialDeliveries	0.333	1.0	0.2	5.0
delivery:productAvailability	0.182	0.333	0.125	16.0

delivery:time	0.213	0.333	0.156	32.0
disputeResolution	0.727	0.774	0.686	35.0
intellectualProperty	0.0	0.0	0.0	7.0
liabilityScope	0.5	0.588	0.435	46.0
orderRestrictions	0.0	0.0	0.0	5.0
party	0.17	0.364	0.111	36.0
payment:costs	0.375	0.5	0.3	10.0
payment:creditRating	0.25	0.5	0.167	12.0
payment:default	0.0	0.0	0.0	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.505	0.683	0.4	70.0
payment:nettingAndWithholding	0.364	1.0	0.222	9.0
payment:time	0.308	0.545	0.214	28.0
placeOfFulfillment	0.5	0.5	0.5	4.0
placeOfJurisdiction	0.417	0.625	0.312	16.0
prices	0.296	0.533	0.205	39.0
realization:offerAndAcceptance	0.571	0.656	0.506	79.0
realization:orderProcess	0.694	0.833	0.595	42.0
retentionOfTitle	0.658	0.923	0.511	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.462	0.75	0.333	9.0
scope	0.525	0.6	0.467	45.0
sellerWithdrawalRight	0.267	0.4	0.2	10.0
userAccount	0.154	0.5	0.091	11.0
vouchersDiscountsPromotionsGiftcards	0.796	0.896	0.717	60.0
warranty:contractualClaims	0.31	0.407	0.25	44.0
warranty:exclusion	0.222	0.333	0.167	18.0
warranty:lapse	0.4	0.611	0.297	37.0
warranty:legalClaims	0.465	0.588	0.385	26.0
withdrawal:consequences	0.377	0.556	0.286	35.0
withdrawal:exclusion	0.214	0.333	0.158	19.0
withdrawal:form	0.5	0.75	0.375	8.0
withdrawal:right	0.559	0.826	0.422	45.0

Table A.6.: CNN with embedding layer trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

Class	F_1-Score	Precision	Recall	Support
batteryElectronicsAndPackaging	1.0	1.0	1.0	9.0
choiceOfLaw	0.96	0.923	1.0	24.0
codeOfConduct	1.0	1.0	1.0	11.0
contractChange	0.5	0.5	0.5	2.0
contractLanguage	0.976	0.952	1.0	20.0

contractObject	0.727	1.0	0.571	14.0
contractStorage	0.952	1.0	0.909	22.0
contractTermination	0.0	0.0	0.0	0.0
customerSupport	0.615	0.8	0.5	8.0
dataProtectionAndProcessing	0.809	0.792	0.826	23.0
definition:consumer	0.968	1.0	0.938	16.0
definition:customer	0.0	0.0	0.0	3.0
definition:entrepreneur	0.97	0.941	1.0	16.0
definition:other	0.0	0.0	0.0	3.0
definition:workday	0.0	0.0	0.0	0.0
delivery:acceptance	0.0	0.0	0.0	7.0
delivery:collection	0.952	1.0	0.909	11.0
delivery:costs	0.675	0.788	0.591	44.0
delivery:destination	0.444	1.0	0.286	21.0
delivery:inspectionAndDamages	0.462	0.9	0.31	29.0
delivery:liability	0.919	0.85	1.0	17.0
delivery:method	0.545	0.9	0.391	23.0
delivery:partialDeliveries	0.8	0.8	0.8	5.0
delivery:productAvailability	0.812	0.812	0.812	16.0
delivery:time	0.69	0.769	0.625	32.0
disputeResolution	0.971	0.971	0.971	35.0
intellectualProperty	0.8	0.75	0.857	7.0
liabilityScope	0.901	0.911	0.891	46.0
orderRestrictions	0.5	0.667	0.4	5.0
party	0.781	0.893	0.694	36.0
payment:costs	0.588	0.714	0.5	10.0
payment:creditRating	0.783	0.818	0.75	12.0
payment:default	0.571	1.0	0.4	15.0
payment:invoice	0.0	0.0	0.0	1.0
payment:method	0.829	0.768	0.9	70.0
payment:nettingAndWithholding	0.615	1.0	0.444	9.0
payment:time	0.766	0.947	0.643	28.0
placeOfFulfillment	0.857	1.0	0.75	4.0
placeOfJurisdiction	0.941	0.889	1.0	16.0
prices	0.892	0.943	0.846	39.0
realization:offerAndAcceptance	0.936	0.948	0.924	79.0
realization:orderProcess	0.881	0.881	0.881	42.0
retentionOfTitle	0.945	0.977	0.915	47.0
rightToRefuse	0.0	0.0	0.0	1.0
salvatorius	0.947	0.9	1.0	9.0
scope	0.943	0.976	0.911	45.0
sellerWithdrawalRight	0.588	0.714	0.5	10.0
userAccount	0.308	1.0	0.182	11.0

vouchersDiscountsPromotionsGiftcards	0.947	1.0	0.9	60.0
warranty:contractualClaims	0.659	0.638	0.682	44.0
warranty:exclusion	0.5	0.7	0.389	18.0
warranty:lapse	0.754	0.812	0.703	37.0
warranty:legalClaims	0.7	0.618	0.808	26.0
withdrawal:consequences	0.833	0.811	0.857	35.0
withdrawal:exclusion	0.733	1.0	0.579	19.0
withdrawal:form	1.0	1.0	1.0	8.0
withdrawal:right	0.875	1.0	0.778	45.0

Table A.7.: Multi-input CNN trained on version 2 corpus using clause and paragraph information as input to predict level 2 labels - results on test set

A.2. Training Process

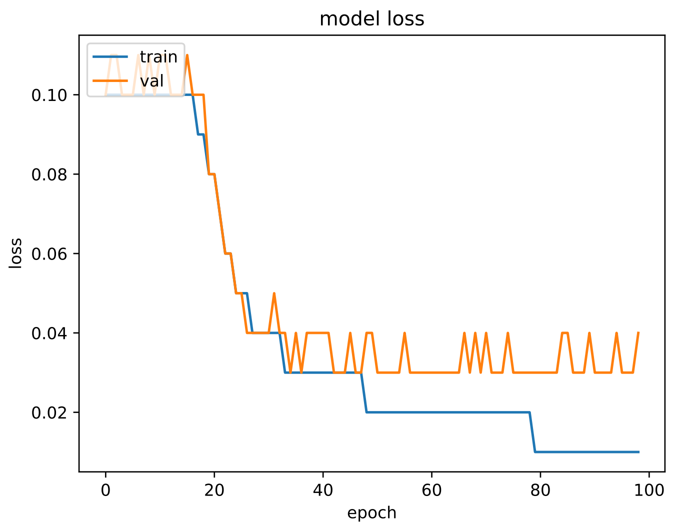

Figure A.1.: CNN trained on corpus version 2 to predict level 1 labels - loss during training process

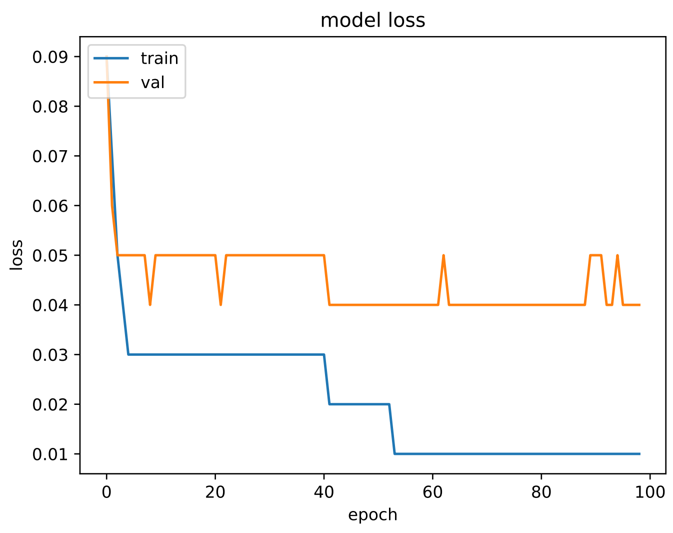

Figure A.2.: Multi-input CNN trained on corpus version 2 to predict level 1 labels - loss during training process

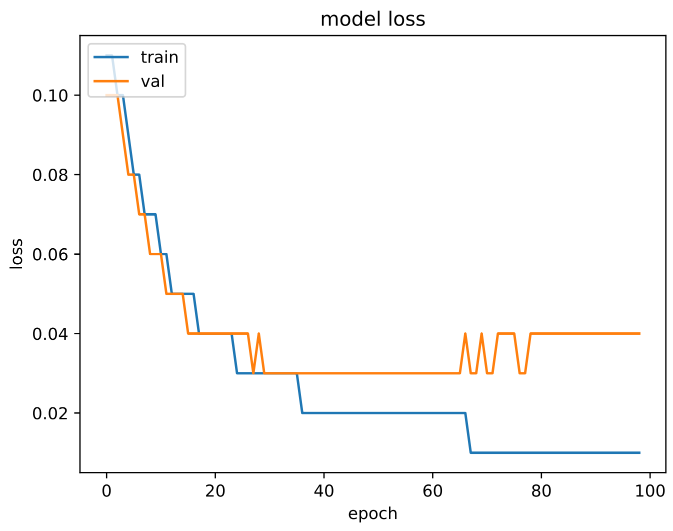

Figure A.3.: LSTM trained on corpus version 2 to predict level 1 labels - loss during training process

Bibliography

[1] N. Aletras, D. Tsarapatsanis, D. Preoţiuc-Pietro, and V. Lampos. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science*, 2:e93, 2016.

[2] A. Bakarov. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*, 2018.

[3] Y. Bakos, F. Marotta-Wurgler, and D. R. Trossen. Does anyone read the fine print? consumer attention to standard-form contracts. *The Journal of Legal Studies*, 43(1):1–35, 2014.

[4] V. Balakrishnan and E. Lloyd-Yemoh. Stemming and lemmatization: a comparison of retrieval performances. *Lecture Notes on Software Engineering*, 2(3):262–267, 2014.

[5] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

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

[7] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[8] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[9] G. Contissa, K. Docter, F. Lagioia, M. Lippi, H.-W. Micklitz, P. Pałka, G. Sartor, and P. Torroni. Claudette meets gdpr: Automating the evaluation of privacy policies using artificial intelligence. *Available at SSRN 3208596*, 2018.

[10] S. Dreiseitl and L. Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, 35:352–359, 2002.

[11] T. Evgeniou and M. Pontil. Support vector machines: Theory and applications. In *Advanced Course on Artificial Intelligence*, pages 249–257. Springer, 1999.

[12] M. Fenner. *Machine Learning with Python for Everyone*. Addison-Wesley Professional, 2019.

[13] V. Ganganwar. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47, 2012.

[14] N. Garg, L. Schiebinger, D. Jurafsky, and J. Zou. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644, 2018.

[15] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[16] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[17] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(4):249–268, 2007.

[18] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown. Text classification algorithms: A survey. *Information*, 10:150, 2019.

[19] F. Lagioia, F. Ruggeri, K. Drazewski, M. Lippi, H.-W. Micklitz, P. Torroni, and G. Sartor. Deep learning for detecting and explaining unfairness in consumer contracts. In *JURIX*, pages 43–52, 2019.

[20] J. Lever, M. Krzywinski, and N. Altman. Classification evaluation. *Nature Methods*, 13:603–604, 2016.

[21] B. Li, S. Yu, and Q. Lu. An improved k-nearest neighbor algorithm for text categorization. *arXiv preprint cs/0306099*, 2003.

[22] M. Lippi, F. Lagioia, G. Contissa, G. Sartor, and P. Torroni. Claim detection in judgments of the eu court of justice. In *AI Approaches to the Complexity of Legal Systems*, pages 513–527. Springer, 2015.

[23] M. Lippi, P. Palka, G. Contissa, F. Lagioia, H.-W. Micklitz, Y. Panagis, G. Sartor, and P. Torroni. Automated detection of unfair clauses in online consumer contracts. In *JURIX*, pages 145–154, 2017.

[24] M. Lippi, P. Pałka, G. Contissa, F. Lagioia, H.-W. Micklitz, G. Sartor, and P. Torroni. Claudette: an automated detector of potentially unfair clauses in online terms of service. *Artificial Intelligence and Law*, 27(2):117–139, 2019.

[25] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11, 1968.

[26] M.-F. Moens, E. Boiy, R. M. Palau, and C. Reed. Automatic detection of arguments in legal texts. In *Proceedings of the 11th international conference on Artificial intelligence and law*, pages 225–230, 2007.

[27] J. A. Obar and A. Oeldorf-Hirsch. The biggest lie on the internet: Ignoring the privacy policies and terms of service policies of social networking services. *Information, Communication & Society*, 23(1):128–147, 2020.

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

[29] J. Plisson, N. Lavrac, and D. Mladenic. A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86, 2004.

[30] J. R. Quevedo, O. Luaces, and A. Bahamonde. Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognition*, 45(2):876–883, 2012.

[31] G. Rebala, A. Ravi, and S. Churiwala. *An Introduction to Machine Learning*. Springer International Publishing, 2019.

[32] X. Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[33] N. SpolaôR, E. A. Cherman, M. C. Monard, and H. D. Lee. A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151, 2013.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[35] A. Sun, E. P. Lim, W. K. Ng, and J. Srivastava. Blocking reduction strategies in hierarchical text classification. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1305–1308, 2004.

[36] V.N. Vapnik and A.Y. Chervonenkis. A class of algorithms for pattern recognition learning. *Avtomat. i Telemekh.*, 25:937 – 945, 1964.

[37] J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[38] W. Yin, K. Kann, M. Yu, and H. Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.