INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine
Universität
Düsseldorf

Mining Argument Components in Public Participation Processes

Suzan Padjman

Masterarbeit

Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

[redacted]

[redacted]

Abstract

Evaluating public participation processes requires a lot of resources such as time, money and expertise. Therefore, tools are desired that help to automate and accelerate the evaluation process in order to support municipalities in decision-making. There are different methods from the area of machine learning that may help to support the evaluation of public participation processes. In this thesis, the focus lies on the automated classification of argument component types. In particular, the goal is to identify *major positions* and *premises* among argumentative sentences in the contributions, following up on the work of Romberg and Conrad (2021). However, in this work it is additionally considered that sentences may contain both argument types, i.e., the task is viewed as a *multi-label classification problem* where samples can be associated with more than just one label.

For the classification of argument component types, the following machine learning methods were applied and compared to another: BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2019), SVM (Cortes and Vapnik, 1995) and XGBoost (T. Chen and Guestrin, 2016). Since SVM and XGBoost do not support multi-label classification directly, different problem transformation methods were considered, which transform the multi-classification problem into one or multiple single-label classification problems. Among different problem transformation methods, *label power-set*, *binary relevance* and *classifier chains* were chosen. In order to train and test the models for the multi-label classification task, five datasets provided by Romberg et al. (2022) were used, which stem from German public participation processes in the domain of urban planning and sustainable mobility.

First, the proposed models were evaluated in an dataset-internal evaluation, in which BERT and DistilBERT clearly outperformed SVM and XGBoost. BERT achieved the best results, attaining an average macro F_1 score of 0.92 on the datasets. Nevertheless, the significantly smaller and more efficient DistilBERT model achieved similar good results with an average macro F_1 score of 0.91. Among the applied problem transformation methods for SVM and XGBoost, binary relevance achieved slightly better results.

Next, the performances of BERT, DistilBERT, XGBoost and SVM were investigated in terms of generalizability. For that, the models were evaluated in a cross-dataset evaluation, i.e., the models were trained on one or more datasets and evaluated on the remaining, unseen datasets. The results have shown that BERT and DistilBERT again outperformed SVM and XGBoost by producing better and more stable results. Using the largest dataset for training, BERT achieved an average macro F_1 score of 0.90 on the remaining datasets and DistilBERT an average macro F_1 score of 0.89, achieving highly generalizable results. In contrast, SVM and XGBoost showed weaknesses in generalizing on some of the datasets.

Overall, the achieved results for the multi-label classification of major positions and premises were comparable to the results by Romberg and Conrad (2021) for the single-label classification. Using BERT, they attained an average macro F_1 score of 0.91 for the dataset-internal evaluation and an average macro F_1 score of 0.90 for the cross-dataset evaluation, which lie very close to our results.

Contents

1 Introduction

Municipalities are urban political units of a local government, i.e., they are usually formed by cities, boroughs, villages or towns. Their tasks include providing public services such as waste disposal, police and fire protection, water supply, health services and making decisions on local political issues and services (Encyclopaedia Britannica, 1998). Since these decisions may affect the daily lives of citizens, it is desirable to make decisions that are in the best interest of the community. In this regard, the involvement of the citizens concerned plays an important role in municipal decision-making processes.
There exist multiple ways, in which citizens can engage in municipal decision-making processes, e.g., by expressing their suggestions through e-mails or letters, on online platforms or on social media. For the term *public participation*, many definitions can be found in the literature. Creighton (2005, p. 7) defined it as "the process by which public concerns, needs, and values are incorporated into governmental and corporate decision making. It is two-way communication and interaction, with the overall goal of better decisions that are supported by the public". Similarly, Rowe and Frewer (2004, p. 512) called it "the practice of consulting and involving members of the public in the agenda-setting, decision-making, and policy-forming activities of organizations or institutions responsible for policy development". According to Arnstein (1969, p. 216), *citizen participation*, another commonly used term for public participation, stands for "citizen power", which means that it enables citizens that have no governmental authority to be included in politics and economic processes. However, most of the definitions have in common that public participation applies to administrative decisions, includes an interaction between the organization making the decision and the participants, is an organized process for involving the public and participants have some level of impact on the decision-making process (Creighton, 2005).
In the past decades, especially electronic forms of public participation, so-called *e-participation*, have increasingly become a common instrument used by governments all across the world (Le Blanc, 2020). Moreover, the involvement of the public in municipal decision-making is viewed as an effective way of improving the quality and legitimacy of decisions. Studies like Halvorsen (2003) have shown that public participation can affect the citizens' belief in desirable ways, e.g., citizens are more likely to believe that the respective municipality strives to be responsive to public concerns. Also, according to the study the public tends to tolerate more disagreement over taken actions. Consequently, demands for public participation have expanded in such a way that it has become a cornerstone of democracy (Roberts, 2004).
In order to consider the citizens' suggestions in municipal decision-making processes, their contributions have to be analyzed first. However, the growth of public participation processes leads to a high number of generated contributions. Hence, evaluating thousands of citizen comments by hand can very challenging and time-consuming (Teufl et al., 2009). Furthermore, an adequate evaluation of the contributions often requires experts with sufficient in-depth domain knowledge. K. Chen and Aitamurto (2019) have found that despite the strong encouragement of policy-makers for high numbers of participation, the overload of user comments exceeds the government's resources for analyzing all of them effectively. If municipalities do not have enough resources such as time, money and expertise to cover the requirements for analyzing all contributions, in extreme

cases only a small portion of the contributions can be taken into account (Arana-Catania et al., 2021). This may affect the legitimacy of the evaluation negatively. For instance, filtering out contributions could lead to an under-representation of some interest groups or opinions. In order to surpass these obstacles of evaluating public participation processes, tools are desired that help to automate and accelerate the evaluation process. In particular, algorithms from the area of machine learning have proven to be promising for automating content analysis (Scharkow, 2011).

There have been several attempts to automate the evaluation of public participation processes with the help of methods from the area of machine learning. One approach to tackle the problem of information overload in e-participation platforms is clustering and categorizing comments by topics or grouping citizens by interest. For example, Kwon et al. (2006) developed techniques to help rule-makers analyze large numbers of public comments, inter alia, by categorizing the comments into pre-defined subtopics. Also, Teufl et al. (2009) proposed a framework that employs different machine learning methods to automatically cluster, pre-screen and pre-evaluate public opinions and contributions from an Austrian e-participation platform. The framework is designed to gain a better picture of texts dealing with similar topics and to provide a real time data analysis for e-participation platforms. Similarly, Arana-Catania et al. (2021) developed a tool that summarizes user comments posted on a Spanish Consul platform[1] and groups citizens by interest.

Besides thematic structuring, *argument mining* is also a commonly used approach to automate public participation analysis. Argument mining is a field of research that focuses on the analysis of arguments, e.g., by identifying argumentative text passages and categorizing them. The automated identification and extraction of specified argument components can help an analyst to take the participants' opinions and suggestions into consideration. Most related studies in the field of argument mining, e.g., Park and Cardie (2014), Liebeck et al. (2016) and Romberg and Conrad (2021), share the common task of a two-step classification: First, they aim to identify argumentative and non-argumentative sentences. Second, they classify argumentative sentences into argument components according to a specified argument model. More precisely, Park and Cardie (2014) developed a framework to recognize propositions in user content collected from an eRulemaking platform and assign each proposition with respect to its verifiability to pre-defined categories. Liebeck et al. (2016) analyzed user content from the German online participation project *Tempelhofer Feld*, regarding a former airport in Berlin and its possible future use. They classified argumentative sentences from user content as *major positions* (conclusions), *claims* or *premises*. Similarly, Romberg and Conrad (2021) applied argument mining in order to analyze German public participation processes that address urban planning. They specified an argument model based on Liebeck et al. (2016) and categorized sentences as non-argumentative, major positions or premises. Also, they compared a new approach based on BERT (Devlin et al., 2019) with previously published approaches. Apart from extracting argument component types, there are also studies that focus on analyzing the argument structure and the relation between arguments, e.g, by Kwon et al. (2006) and Cocarascu et al. (2020).

All in all, Romberg and Conrad (2021) produced encouraging results for identifying argumentative sentences and classifying major positions or premises among these, however,

[1] www.consulproject.org

some edge cases have not been taken into account. The authors viewed the task of identifying major positions and premises as a traditional *single-label classification* problem, i.e., the possibility of sentences containing both major positions and premises was not considered. Nevertheless, in real-life applications sentences may contain both major positions and premises (e.g., "The road is damaged and should be repaired.").

Since analyzing arguments from public participation user content can play an important role for municipal decision-making processes, this thesis will thus deal with the task of analyzing arguments with the help of argument mining techniques. This work aims at the classification of argumentative sentences into different classes of argument component types and follows up on the work of Romberg and Conrad (2021). Given argumentative sentences from citizen contributions that originate from German participation processes in the domain of urban planning and sustainable mobility, machine learning methods are applied in order to identify major positions and premises among these sentences. The focus lies on the case of sentences that contain both major positions and premises, i.e., the classification task will be viewed as a *multi-label classification* problem, in which one instance can have more than one label. For that, the public participation datasets provided by Romberg et al. (2022) will be used. The goal is to further improve the practical use of the earlier mentioned methods in order to provide a helpful tool for supporting the evaluation of public processes in the domain of urban planning and beyond.

This thesis is organized as follows: First, in Section 2 the work of Romberg and Conrad (2021) and the provided datasets by Romberg et al. (2022) are introduced. Moreover, the research goal of this thesis is addressed. Next, problem transformation techniques for solving the multi-label classification task are discussed and selected in Section 3. In Section 4, machine learning algorithms for the classification problem are proposed, followed up by the explanation of considered features for the algorithms in Section 5. Then, in Section 6 the experimental setup for the evaluation of the models is explained and in Section 7, the proposed methods are evaluated. Finally, in Section 8 a conclusion is drawn and an outlook on future-work is given.

2 Previous Work

This thesis follows up on the work of Romberg and Conrad (2021) and aims at the classification of argumentative sentences into different classes of argument component types, specifically, into *major positions* and *premises*. In this section, the datasets that were used by the authors are described in detail along with the associated annotation scheme. Furthermore, the approaches and results are presented and shortcomings are discussed. Lastly, the research goal of this thesis is addressed.

2.1 Description of the Dataset

Romberg and Conrad (2021) rely on the five datasets provided by Romberg et al. (2022), which originate from German public participation processes concerning urban planning and mobility transition. While four of the datasets focus on cycling, the latter concerns a more general mobility concept. In the following, the different datasets are described in more detail.

Cycling dialogues
Three of the five datasets are taken from e-participation processes concerning the improvement of cycling infrastructure in the German cities Bonn[2], Moers[3] and the Cologne city district Ehrenfeld[4]. As part of the cycling dialogues, citizens were encouraged to make suggestions and propose ideas on how to make cycling more attractive in those cities, so that future measures can be planned. The cycling dialogues took place during a five-week period in 2017. Citizens were able to make propositions, discuss or rate propositions and mark concerned places on a map-based online platform. In the following, the datasets that stem from the cycling dialogues of the cities of Bonn, Moers and the Cologne district Ehrenfeld will be denoted as CD_B, CD_M and CD_C respectively. CD_B contains 10,442 sentences from 2,163 contributions, CD_M contains 2,193 sentences from 459 contributions and CD_C contains 1,704 sentences from 366 contributions.

Citizen questionnaire on cycling
In addition to the cycling dialogue for the city of Bonn, a dataset originating from a postal survey is provided. For the postal survey, participants were chosen randomly among the city's population. In the questionnaire, among others, the participants were asked to make proposals for improving cycling infrastructure. Participants had the option to fill out the questionnaire by hand or online. The dataset containing contributions from the postal survey for the city of Bonn, denoted as CQ_B, comprises 1,505 sentences from 1,386 contributions.

Mobility concept
The German city of Krefeld has conducted multiple e-participation processes concerning the city's general mobility concept since 2019. The fifth dataset, which will be denoted

[2] https://www.raddialog.bonn.de
[3] https://www.raddialog.moers.de
[4] http://www.raddialog-ehrenfeld.koeln

as MC_K, contains citizen comments from two connected e-participation processes for the city of Krefeld. In the first process, citizens were asked to express their opinions regarding some citywide action plans concerning urban development, regional cooperation and various forms of traffic and transport. The second process allowed citizens to make concrete proposals regarding formulated action plans in specified city districts. In total, MC_K includes 2,008 sentences from 337 contributions.

The five datasets comprise 17,852 annotated sentences in total, of which 15,517 sentences are argumentative. Each dataset is given as a tsv-file that contains one entry per sentence with the following information: `id` specifies a unique sentence id for each sentence, `document_id` is the id of the document that contains the sentence and `sentence_nr` is the sentence number of the sentence in the document. Entries in the column `content` contain the textual content of the sentences and entries in the column `code` refer to the annotation label of the sentences ("non-arg", "mpos", "premise" or "mpos+premise"). For more information about the labels, see the described argumentation scheme in Section 2.2. Furthermore, `Title/text` denotes whether the sentence is taken from the title or the body text of a document and `curated/single` denotes whether the sentence was curated or coded by a single person. The column `dataset` contains the name of the dataset (e.g., "CD_B") and `url` contains the original url of the document.

2.2 Argumentation Model

Since all datasets originate from urban planning processes, the proposed improvements in the contributions are mostly based on infrastructure problems or planning deficits. Romberg and Conrad (2021) follow the argumentation scheme and terminology introduced by Liebeck et al. (2016), who developed an argumentation model for public participation processes.

Liebeck et al.'s argumentation model is based on three different types of argument components: *major positions*, *claims* and *premises*. They define major positions as "options for actions or decisions that occur in the discussion", e.g., "We should build a playground with a sandbox.". In other words, major positions are suggestions from citizens for improvements. A claim is defined as a "pro or contra stance towards a major position", in which citizens express their opinion regarding a suggestion, e.g., "Yes, we should definitely do that!". Furthermore, a premise is defined as a "reason that attacks or supports a major position, a claim or another premise", e.g., "This would allow us to save money.". In contrast to Liebeck et al., Romberg and Conrad (2021) consider only major positions and premises in their argumentation model. Claims were left out since the considered datasets do not include feedback comments on the suggestions made by other users, i.e., their focus lies on the classification of major positions and premises only.

Each sentence in the provided datasets by Romberg et al. (2022) is annotated with one of the following labels:

- *non-arg* marks sentences that are non-argumentative and contain no major position or premise. However, in this work non-argumentative sentences will not be considered.

id	document_id	content	title/text	code
3	B1592	"Könnte man aus mehreren Straßen der Innenstadt (Am Hof, Rathausgasse, Wesselstraße, Am Neutor) nicht beidseitig geöffnete Fahrradstraßen machen?" (engl. "Couldn't several streets in the city center (Am Hof, Rathausgasse, Wesselstraße, Am Neutor) be made into bicycle lanes open on both sides?")	text	mpos
4	B1593	"Radweg trifft auf Straße" (engl. "Bicycle lane meets road")	title	premise
5	B1593	"Hier wäre eine breite Einmündung ohne Bordsteinkante und Poller von Vorteil, eine Vorfahrt für Radfahrer*innen sinvoll, da dies eine stark befahrene Pendlerstrecke ist." (engl. "Here, a wide junction without curb and bollard would be advantageous, and a right of way for cyclists would make sense, as this is a busy commuter route.")	text	mpos+premise
10442	Boff10004_1	"Geradlinige Wegführung!" (engl. "Straight path!")	text	non-arg
10443	Boff10004_1	"An Ampelübergängen kreuzt sich der Rad- & Fußgängerweg." (engl. "At traffic light crossings, the bike & pedestrian path crosses.")	text	premise
10444	Boff10004_1	"Das gibt oft gefährliche Situationen" (engl. "This often leads to dangerous situations")	text	premise

Table 1: Example sentences taken from the cycling dialogue dataset CD_B and the citizen questionnaire dataset CQ_B. For a better overview, irrelevant columns of the datasets have been removed. The sentences with id 3, 4 and 5 originate from CD_B and the sentences with id 10442, 10443 and 10444 from CQ_B. The sentences have been translated into English (in parentheses).

- *mpos* denotes that the sentence contains a major position, i.e., an option for action or decision.
- *premise* labels sentences that contain a premise, i.e., a reason that attacks or supports a major position or another premise.
- *mpos+premise* indicates that the sentence contains both argument component types.

For a better understanding of the data and corresponding labels, some example sentences from the cycling dialogue dataset CD_B and the citizen questionnaire dataset CQ_B are shown in Table 1.

Table 2 shows the distribution of sentences among the different argument component

	CD_B		CD_C		CD_M		CQ_B		MC_K		all
total	10,442		1,704		2,193		1,505		2,008		17,852
non-arg	1,153	(11.0%)	197	(11.6%)	382	(17.4%)	172	(11.4%)	431	(21.5%)	2,335
mpos	2,589	(24.8%)	556	(32.6%)	359	(16.4%)	960	(63.8%)	892	(44.4%)	5,356
premise	6,438	(61.7%)	904	(53.1%)	1,407	(64.2%)	250	(16.6%)	616	(30.7%)	9,615
mpos+premise	262	(2.5%)	47	(2.8%)	45	(2.1%)	123	(8.2%)	69	(3.4%)	546

Table 2: Distribution of sentences among the different argument component categories for each dataset.

types for each dataset. As earlier mentioned, non-argumentative sentences will not be considered in this work. The multi-label *mpos+premise* is highly under-represented compared to the single labels *mpos* and *premise*. The percentage of sentences that are labeled with *mpos+premise* is the highest in the dataset CQ_B (8.2%) and the lowest for CD_M (2.1%). Moreover, in the datasets CD_B, CD_C and CD_M, *premise* forms the majority class, whereas in CQ_B and MC_K, *mpos* occurs most frequently. This disparity may be due to the different types of public participation processes the datasets stem from. CD_B, CD_C and CD_M originate from cycling dialogues, while CQ_B originates from a postal survey and MC_K concerns a more general mobility concept. Since all five datasets are noticeably imbalanced, this may affect the ability of the applied classification algorithms to recognize the minority class negatively.

In the annotation process, each sentence of the datasets was annotated by three coders and was subsequently reviewed by two supervisors in a curation step. The inter-coder agreement was measured using Fleiss' κ (Fleiss, 1971), which is defined as follows:

$$\kappa = \frac{\overline{P} - \overline{P_e}}{1 - \overline{P_e}} \tag{1}$$

Here, $\overline{P_e}$ denotes the expected agreement by chance, while $\overline{P}$ is the overall observed agreement between the coders. To summarize, Fleiss' κ expresses to which degree the amount of agreement exceeds the expected amount if all coders make random ratings and takes values between -1 and 1. High κ values indicate a high amount of agreement. For example, if the coders are in complete agreement, then $\kappa = 1$ holds. Negative values, on the other hand, indicate an agreement that is even smaller than the expected agreement by chance, e.g., in the case of systematical disagreement. If there is complete disagreement between the coders, then $\kappa = -1$ holds.
Table 3 shows the inter-coder agreement that was estimated on about 10% of the sentences of each dataset. It should be noted again that non-argumentative sentences are not considered in this work. The high average κ values for the labels *mpos* and *premise* (0.81 and 0.84) indicate a high agreement among the codings. In contrast, the label *mpos+premise* shows a lower agreement with an average κ of 0.44 for all datasets. This may affect the ability of the classification algorithms to identify sentences that belong to both classes of argument component types negatively.

		CD_B	CD_C	CD_M	CQ_B	MC_K	all
	sentences	1,251	191	230	188	376	2,236
Fleiss' κ	non-arg	0.58	0.68	0.67	0.59	0.69	0.63
	only mpos	0.82	0.81	0.81	0.73	0.78	0.81
	only premise	0.82	0.87	0.83	0.83	0.77	0.84
	mpos+premise	0.37	0.35	0.40	0.69	0.42	0.44
	overall	0.76	0.80	0.77	0.72	0.73	0.77

Table 3: Number of sentences under consideration and calculated Fleiss' κ agreement for argument component annotation. Values for *mpos* and *premise* exclude example sentences annotated with *mpos+premise* and values for *mpos+premise* exclude example sentences that are annotated with *mpos* or *premise* as a single label. The Fleiss' κ values were calculated based on the original codings of the datasets provided by Romberg et al. (2022). Since in this work examples labeled with *mpos+premise* were excluded when calculating the Fleiss' κ values for *mpos* and *premise*, the scores differ from the tables shown in the related works.

2.3 Previous Results

The objective of Romberg and Conrad (2021) was to help automating the analysis of public participation processes by applying argument mining techniques and identifying major positions and premises in the contributions. For that, their argument mining task was divided into two subtasks. First, they identified argumentative sentences from the contributions. Next, the authors classified argumentative sentences according to their specified argumentation scheme, differentiating between the two classes major position and premise. In this work, the focus lies on the second task of identifying argument component types, thus the first task will not be further discussed.

In order to classify major positions and premises among argumentative sentences, the authors applied SVM (Cortes and Vapnik, 1995), fastText (Joulin et al., 2017), the transformer-based model BERT (Devlin et al., 2019) and the ensemble method ECGA (Giannakopoulos et al., 2019). SVM, fastText and ECGA had already been used in previous approaches for analyzing argument components in the domain of public participation processes, e.g., by Kwon et al. (2006), Liebeck et al. (2016) and Giannakopoulos et al. (2019). In contrast, BERT had only been used by Cocarascu et al. (2020) on public participation data to identify relations between argumentative sentences. However, BERT has shown promising results for argument mining tasks in other domains, e.g., in Chakrabarty et al. (2019).

First, the authors evaluated the classification algorithms' performances on each of the five datasets individually. Table 4 shows the results for each individual dataset. BERT performed best among all methods, achieving F_1 macro scores between 0.86 and 0.93. The second best classifier was SVM, achieving F_1 macro scores between 0.79 and 0.88. For fastText and ECGA, two model variants were evaluated. One model was trained on the original, imbalanced dataset and the other one on a modified version of the dataset, in which the majority class was undersampled by picking random samples. In the table, the results of the better variants are listed. Compared to BERT and SVM, fastText and ECGA performed poorly and produced less stable results. Applying undersampling only slightly increased the performance of ECGA. Still, ECGA showed poor results for

		SVM	*fastText*	*ECGA*	*BERT*
CD_B	mpos	0.82 (0.01)	0.79 (0.01)	0.78 (0.03)	**0.90** (0.01)
	prem	0.93 (0.01)	0.93 (0.00)	0.92 (0.01)	**0.96** (0.00)
	macro	0.88 (0.01)	0.86 (0.01)	0.85 (0.02)	**0.93** (0.01)
CD_C	mpos	0.77 (0.02)	0.74 (0.02)	0.76 (0.03)*	**0.89** (0.02)
	prem	0.85 (0.01)	0.86 (0.01)	0.84 (0.01)*	**0.93** (0.01)
	macro	0.81 (0.02)	0.80 (0.02)	0.80 (0.02)*	**0.91** (0.02)
CD_M	mpos	0.67 (0.03)	0.58 (0.03)	0.52 (0.08)*	**0.84** (0.06)
	prem	0.92 (0.01)	0.92 (0.00)	0.86 (0.05)*	**0.91** (0.04)
	macro	0.80 (0.03)	0.75 (0.02)	0.69 (0.06)*	**0.90** (0.03)
MC_K	mpos	0.83 (0.03)	0.83 (0.03)	0.84 (0.03)*	**0.88** (0.02)
	prem	0.75 (0.03)	0.74 (0.04)	0.74 (0.05)*	**0.84** (0.03)
	macro	0.79 (0.03)	0.78 (0.04)	0.79 (0.05)*	**0.86** (0.03)
CQ_B	mpos	0.93 (0.02)	0.92 (0.01)	0.89 (0.03)*	**0.97** (0.01)
	prem	0.70 (0.08)	0.58 (0.06)	0.55 (0.10)*	**0.88** (0.03)
	macro	0.81 (0.05)	0.75 (0.04)	0.72 (0.06)*	**0.93** (0.02)

Table 4: Intra-dataset evaluation for the classification of major positions (mpos) and premises (prem) among argumentative sentences. Scores are mean F_1 values of the five test sets obtained from a 5-fold cross-validation and standard deviation is given in parentheses. Model variants using undersampling are marked with an asterisk. (Romberg and Conrad, 2021)

datasets that do not have sufficient samples in the minority class. In the case of fastText, undersampling did not improve the performance at all.

Furthermore, Romberg and Conrad compared the behavior of each method in a cross-dataset evaluation (cf. Figure 1) to investigate the models' generalizability. Each classification algorithm was trained on the largest dataset CD_B and then tested on the remaining datasets. Additionally, all classifiers were evaluated with undersampling the majority class. BERT achieved the best results (F_1 macro scores above 0.88) and generalized very well. Both BERT variants (with and without undersampling) performed similarly well, while the undersampling variant produced slightly more stable results. The remaining methods, SVM, fastText and ECGA, showed weaknesses in generalizing on the datasets MC_K and CQ_B. Applying undersampling lead to more stable results, nevertheless, their performances were clearly outperformed by BERT.

All in all, the authors have shown that BERT outperforms previous approaches and provides comparable results among different datasets. SVM achieved the second best results, while fastText and ECGA performed poorly compared to BERT and SVM.

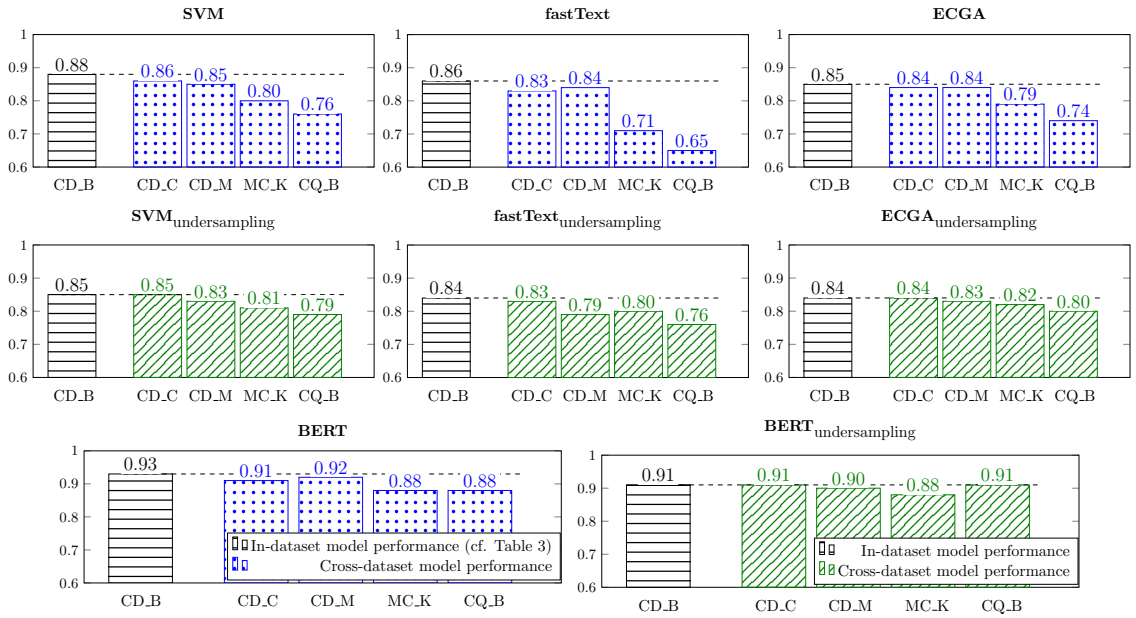


Figure 1: Cross-dataset evaluation for the classification of major positions (mpos) and premises (prem) among argumentative sentences. Results are averaged macro F_1 values of the five models trained on CD_B. (Romberg and Conrad, 2021)

2.4 Research Goal

This thesis deals with the automatic classification of argument component types with the help of argument mining techniques. The objective of this work is to help automating the evaluation process of public participation processes concerning urban planning and beyond.

The related work by Romberg and Conrad (2021) produced encouraging results for the classification of major positions and premises among argumentative sentences. However, sentences in real-life applications may contain both argument component types (see example sentence with id 5 from Table 1). This edge case was not considered by Romberg and Conrad. The authors divided argumentative sentences in either major positions or premises and left aside sentences that are labeled with both classes. Hence, this thesis will extend their work and focus on improving the classification of major positions and premises while taking this edge case into account. In other words, the classification problem will be viewed as a *multi-label classification task* instead of a traditional single-label classification task.

Since multi-label classification requires different strategies from single-label classification, different techniques will be considered and applied in combination with suited classification algorithms. For training the models, the provided datasets by Romberg et al. (2022) will be used. Similar to Romberg and Conrad (2021), the models will be trained and tested on each dataset individually in an intra-dataset evaluation. Next, the models' generalizability will be investigated in a cross-dataset evaluation, using one or more datasets for training and the remaining datasets for testing. Moreover, the results of the

multi-label classification task will be compared to the results achieved by Romberg and Conrad for the single-label classification of major positions and premises.

3 Multi-Label Classification

Argument component classification will be viewed as a *multi-label classification problem* in this thesis. First, the multi-label classification problem is defined and distinguished from the traditional single-label classification problem based on Herrera et al. (2016) and Tsoumakas and Katakis (2007). Most classification algorithms do not support multi-label classification directly, hence different strategies are required. Multi-label classification approaches are commonly divided into *problem transformation methods* and *algorithm adaption methods* in the literature (Tsoumakas and Katakis 2007; Gibaja and Ventura 2015; Trohidis et al. 2008). These are presented next and discussed. Eventually, suitable strategies are selected.

3.1 Single-Label vs. Multi-Label Classification

In general, classification describes a predictive task that aims to learn a model from labeled data in order to predict the label or class of new, unseen examples. The attributes of a dataset used for classification can be grouped into *input features* and *output attributes*. The input features are the variables that will be used for the prediction and the output attributes are the assigned labels or classes to a particular instance. According to the number of output attributes that can be assigned to an instance, different types of classification tasks can be defined, e.g., *single-label classification* and *multi-label classification*.

In single-label classification, there is only one output attribute for each instance and it can take any value from a pre-defined set of labels. In other words, each sample is associated with a single label l from a set of disjoint labels L, $|L| > 1$. In case there are only two possible labels ($|L| = 2$), then one also speaks of *binary classification*. In most cases, the labels of a binary classification problem are referred to as the *positive* and the *negative* class or simply as *true* and *false*. A typical binary classification problem is, for example, spam filtering, for which a classifier can be trained that differentiates between spam and non-spam emails. Also, Romberg and Conrad (2021) viewed the classification of major positions and premises among argumentative sentences as a binary classification task. In case there are more than two labels ($|L| \geq 2$), then one also speaks of *multi-class classification*.

On the other hand, in multi-label classification each sample is associated with a subset of labels $Y \subseteq L$ instead of only a single label. Usually, the associated labels of the data instances are represented as a vector of length $|L|$, in which each element is a binary value that indicates if the corresponding label is assigned to the instance or not. For example, if there are two labels l_1 and l_2 and an instance is only labeled with l_2, then the corresponding vector representation of the associated labels is $\binom{0}{1}$. Multi-label classification is commonly used for text-categorization tasks, in which documents can have several, not mutually exclusive categories. This is also the case for our classification task. As mentioned before, the objective of this work is to identify major positions and premises among argumentative sentences and to additionally consider the case of sentences that contain both argument component types. Hence, the samples can be associated with two labels, turning our classification problem into a multi-label classification task.

(a) single-label classification (b) multi-label classification

Figure 2: Single-label classification vs. multi-label classification. Single-label classifiers take an instance and produce exactly one output label, whereas multi-label classifiers produce one or multiple output labels. (Kang and Bai, 2016)

In contrast to single-label classification, the classification algorithms used for multi-label classification must be able to make multiple predictions for each instance. Figure 2 illustrates the difference between single-label and multi-label classification. A single-label classifier takes an instance and outputs always exactly one label, whereas a multi-label classifier can output several labels at the same time. However, most classification algorithms are single-label classifiers and are not able to do multi-label classification directly. Hence, for multi-label classification different strategies are required. Possible strategies include transforming the original dataset in order to be able to apply one or more single-label classifiers or to adapt the single-label classification algorithm itself to do multi-label classification. Thus, multi-label classification methods are divided into *problem transformation methods* and *algorithm adaption methods*. These methods are further described and discussed in Section 3.2 and 3.3.

3.2 Problem Transformation Methods

Problem transformation methods (PTMs) transform a multi-label classification problem into one or multiple single-label classification problems. In the following, five different PTMs are described. Tsoumakas and Katakis (2007) introduced the problem transformation methods PTM1 to PTM4 and Read et al. (2009) introduced PTM5.
In order to exemplify these methods, we will consider the following multi-label dataset that consists of three made-up example sentences belonging to at least one of the classes *major position* (mpos) and *premise*:

ex.	content	mpos	premise
1	"The bicycle lane should be expanded."	x	
2	"The road surface is uneven."		x
3	"Design wider bicycle lanes; they are too narrow."	x	x

Table 5: Multi-label dataset that consists of three examples. The examples belong to one or both of the following classes: major position (mpos) or premise.

PTM1
A trivial way to transform a multi-label classification problem into a single-classification problem is by randomly selecting one label of the multiple labels of each multi-label

instance.
Table 6 shows the resulting dataset after applying PTM1 on the example dataset from Table 5. Since example 3 was a multi-label example, only one of the two labels is selected:

ex.	content	mpos	premise
1	"The bicycle lane should be expanded."	x	
2	"The road surface is uneven."		x
3	"Design wider bicycle lanes; they are too narrow."	x	

Table 6: Transformed dataset after applying problem transformation method PTM1. For each multi-label example, one label is selected (randomly).

A major disadvantage of this method is that it discards a lot of information content of the original multi-label data. The trained model will not be able to identify sentences that contain both major positions and premises, as the training examples originally marked with both labels were turned into single-label examples. Since we aim to classify and label multi-label sentences accordingly, this method will not be further considered in this work.

PTM2
Another trivial approach to solve a multi-label classification problem is by simply discarding every multi-label instance from the dataset.
Table 7 shows the resulting dataset after applying PTM2 on the running example dataset. Example 3 was a multi-label example, hence it is removed from the dataset:

ex.	content	mpos	premise
1	"The bicycle lane should be expanded."	x	
2	"The road surface is uneven."		x

Table 7: Transformed dataset after applying problem transformation method PTM2. Multi-label examples are removed from the dataset.

However, by removing all example sentences that contain both major positions and premises, a lot of useful information is discarded again. As for PTM1, the trained model will not be able to identify sentences that contain both major positions and premises since all multi-label examples were removed from the training data. Hence, PTM2 will not be further considered.

PTM3 (Label Power-Set)
A different approach, also known as the *label power-set* method, is to consider each different set of labels of the multi-label dataset as a single label. The goal is to learn a single-label classifier $H : X \to P(L)$ that maps an instance $x \in X$ to a member of the power set $P(L)$ of label set L.
Table 8 shows the resulting transformed dataset after applying the label power-set

method on the running example dataset. Since we have two labels (*mpos* and *premise*), $P(L)$ consists of three label combinations: *mpos*, *premise*, and *(mpos* $\wedge$ *premise)*. For each label-combination, a separate class is generated. Example 3, which was marked with both of the labels *mpos* and *premise*, belongs now to the class *(mpos* $\wedge$ *premise)*:

ex.	content	mpos	premise	(mpos $\wedge$ premise)
1	"The bicycle lane should be expanded."	x		
2	"The road surface is uneven."		x	
3	"Design wider bicycle lanes; they are too narrow."			x

Table 8: Transformed dataset after applying problem transformation method PTM3 (label power-set). Each label combination is considered as a separate class.

One disadvantage of label power-set is that it may lead to datasets with a large number of classes ($2^{|L|} - 1$) and only few examples per class. Read et al. (2008) also mention that this method may lead to overfitting because it can only classify examples with label combinations observed in the training data. However, the provided datasets by Romberg et al. (2022) have two labels, which would lead to only three classes. Also, all classes are represented in the datasets. Therefore, this approach will be applied. Nevertheless, it should be noted that the label combination *(mpos* $\wedge$ *premise)* is under-represented in all datasets compared to the single labels, which may affect the ability of the classifier to recognize this particular class negatively.

PTM4 (Binary Relevance)
PTM4, also called *binary relevance* method, is the most commonly used problem transformation method. It decomposes the multi-label classification problem into separate, binary single-label classification problems. This method learns one binary classifier for each label $l \in L$, i.e., in total $|L|$ classifiers: $H_l : X \rightarrow \{l, \neg l\}$. The multi-label dataset is transformed into $|L|$ datasets that contain all examples of the original dataset. In the transformed datasets, examples are labeled as l, if the labels of this particular example in the original dataset contained label l. Otherwise, the example is labeled as $\neg l$. In order to classify a new instance x, binary relevance outputs the union of the labels that are output by all classifiers: $H(x) = \cup_{l \in L}\{l\} : H_l(x) = l$.
Table 9 shows the transformed dataset after applying binary relevance. For each of the labels major position and premise, a separate dataset is generated that contains all of the examples. In both datasets, each example is marked with the respective label l or $\neg l$, depending on the labels of the example in the original dataset, e.g., example 1 was labeled with *mpos* only and is now labeled with *mpos* and $\neg$*(premise)* in the corresponding datasets.

In contrast to label power-set, binary relevance does not tend to overfit label combinations because it does not expect examples to be only associated with label combinations that occurred in the training dataset (Read et al., 2011). Moreover, since label combinations are not considered as separate classes, having less examples for a particular label combination does not affect the ability of the classifier to recognize that class. On the

ex.	content	mpos	¬(mpos)
1	"The bicycle lane should be expanded."	x	
2	"The road surface is uneven."		x
3	"Design wider bicycle lanes; they are too narrow."	x	

(a) Dataset for the label major position (mpos).

ex.	content	premise	¬(premise)
1	"The bicycle lane should be expanded."		x
2	"The road surface is uneven."	x	
3	"Design wider bicycle lanes; they are too narrow."	x	

(b) Dataset for the label premise.

Table 9: Transformed dataset after applying problem transformation method PTM4 (binary relevance). For each label, a separate dataset is generated that contains all of the examples.

other hand, binary relevance assumes that the labels are mutually exclusive and ignores any underlying relations between the classes, which may be a disadvantage in the case of correlating and overlapping classes. Per definition, major positions and premises are not overlapping, but the possibility that both argument component types have some form of relationship cannot be excluded.

In order to determine if there is any association between the classes major position and premise in the datasets, *Cramér's V* (Cramér, 1946) was calculated, which measures the association between two nominal variables and is based on Pearson's chi-squared test (Pearson, 1900). In order to calculate Cramér's V for the classes, a 2×2 contingency table was generated that displays the frequency distribution of the classes.
Let χ^2 be the value of the chi-squared statistic, n the total number of observations, k the number of columns in the contingency table and m the number of rows. Cramér's V is defined as follows: (Cleff, 2008)

$$V = \sqrt{\frac{\chi^2}{n \cdot (\min(k, m) - 1)}} \tag{2}$$

The measure takes values between 0 and 1, while values close to 0 indicate no association. According to Akoglu (2018), values above 0.25 represent a very strong association. Table 10 shows the calculated Cramér's V values for each of the five datasets. The calculated Cramér's V values for the datasets lie between 0.50 and 0.72, hence the classes major position and premise have a very strong association in all datasets. The datasets CD_B and CD_M have noticeably higher Cramér's V values (above 0.7) compared to the remaining datasets.

However, despite being commonly criticized in the literature because of its disadvantage in terms of related classes, Luaces et al. (2012) have shown that the binary relevance method can still lead to good results when a correct implementation of the method is used and an appropriate base learner is selected. Therefore, this approach will still be

	CD_B	CD_C	CD_M	CQ_B	MC_K	avg.
Cramér's V	0.70	0.55	0.72	0.50	0.54	0.60

Table 10: Calculated Cramér's V values for each dataset in order to determine the association between the classes major position and premise.

applied on our multi-label classification task.

PTM5 (Classifier Chains)
Read et al. (2009) introduced a problem transformation method called *classifier chains*, which is based on the binary relevance method, but takes possible relations between the classes into account. This method forms a chain $h = (h_1, ..., h_L)$ of binary classifiers, one for each label. Each classifier h_j in the chain learns to predict, whether an instance is associated with the jth label or not. In contrast to binary relevance, each classifier additionally takes all prior binary relevance predictions into account by extending the features with binary values that indicate, which of the previous labels were assigned to the instance. Thereby, label information is passed between the classifiers. The transformed dataset after applying classifier chains on the running example dataset is identical to Table 9, which was generated by binary relevance.

Since classifier chains includes the advantages of the problem transformation method binary relevance while taking potential relations between the classes into account, this method is selected as well to be applied on our multi-label classification task.

3.3 Algorithm Adaptation Methods

While problem transformation methods are algorithm independent, algorithm adaptation methods contain methods that extend existing single-label classification algorithms in order to handle multi-label data directly (Trohidis et al., 2008). In other words, instead of transforming multi-label data for a single-label classifier, the single-label classifier itself is adapted to handle multi-label data.
Algorithm adaption methods have the advantage that multi-label data can be passed to the models directly without having to transform it. However, although algorithm adaption methods are adaptations of specific algorithms, at their core they are often based on problem transformation methods (Tsoumakas and Katakis, 2007). Furthermore, the flexibility of problem transformation methods allows them to be applied on any classifier, providing a wider range of applicable machine learning algorithms for the multi-label classification task. Therefore, algorithm adaptation methods are not further considered in this work.

4 Methods

The considered machine learning algorithms for the multi-label classification of major positions and premises among argumentative sentences are SVM, BERT, DistilBERT, XGBoost, fastText and ECGA. In the following, these methods are further described and discussed. Eventually, suitable methods among these are selected.

4.1 SVM

SVM (Cortes and Vapnik, 1995) stands for **S**upport **V**ector **M**achine and is originally a binary classifier that separates data points into two classes by constructing a hyperplane as a decision boundary. The idea of the SVM classifier is to find a hyperplane that optimally separates the data into two classes, such that the distance between the closest data points from each class, the *margin*, is as large as possible. The data can be separated linearly in the input space, which is referred to as a *linear SVM*, or it can be mapped non-linearly into a higher-dimensional space with a mathematical technique called *kernel trick*. Since data is usually noisy and data points from different classes are not always strictly separable, some additional properties have to be considered in order to grant a high generalization.

Liebeck et al. (2016) proposed the use of SVMs for the classification of argument component types and Romberg and Conrad (2021) used SVMs as well for the single-label classification of major positions and premises, relying on the best SVM setup by Liebeck et al. and achieving satisfactory results. Hence, this method is chosen to be applied as a base learner for our multi-label classification task in combination with the selected problem transformation methods mentioned in Section 3.2.

In the following, the basic concepts of the SVM classifier such as the construction of the hyperplane, the kernel trick and regularization are explained based on Cortes and Vapnik (1995) and Müller and Guido (2016).

4.1.1 Hyperplane Construction

Consider the case of *linearly separable* training data without errors. Given a set of labeled data points $(x_1, y_1), ..., (x_l, y_l)$ with labels $y_i \in \{-1, 1\}$, there exists a vector w and a scalar b, such that the following constraints are satisfied:

$$w \cdot x_i + b \geq +1, \quad \text{if } y_i = +1 \tag{3}$$

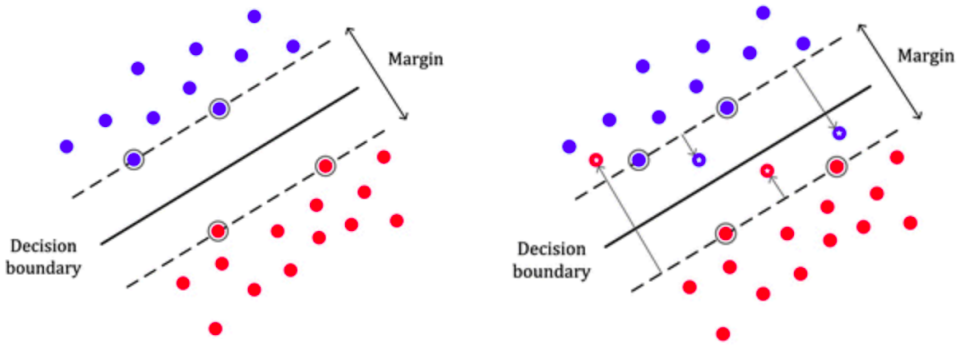
$$w \cdot x_i + b \leq -1, \quad \text{if } y_i = -1 \tag{4}$$

The constraints can be summarized as follows:

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, ..., l \tag{5}$$

Then, an *optimal* hyperplane $w_0 x + b_0$ can be found that separates the data with a maximal margin. In case the data is linearly separable and no misclassification is allowed, this is referred to as *hard margin classification*.

However, in most cases the data is not linearly separable without error. In this case, the goal is to separate the data with a minimal number of errors while keeping the margin



(a) hard margin classification (b) soft margin classification

Figure 3: Difference between hard margin and soft margin classification. In hard margin classification, an optimal hyperplane is constructed that separates the data points into two classes without error while maximizing the margin, whereas in soft margin classification errors are allowed. The different colors represent the associated classes of the data points. The data points defining the margin of the largest separation between the two classes (marked with a thick rim) are called *support vectors*. (Shrivastav and Ramudu, 2020)

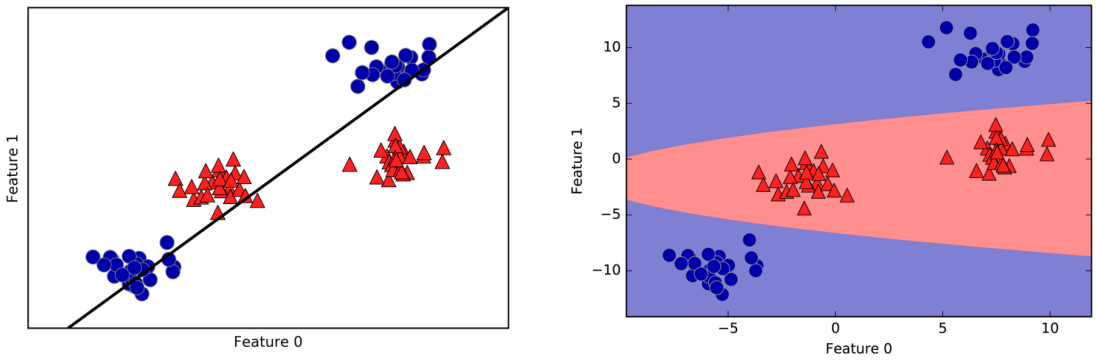
as large as possible. This procedure is called *soft margin classification*. For that, a slack variable $\xi_i \geq 0$ with $i = 1, .., l$ is introduced that corresponds to the distance between the data point x_i and the margin of the associated class if x_i lies on the wrong side of the margin, i.e., on the side that belongs to the other class. If the data point lies on the correct side, then ξ_i is set to zero. In the case of a soft margin classification, the constraints in equation 5 are adapted in order to allow errors:

$$y_i(w * x_i + b) \geq 1 - \xi_i, \quad i = 1, ..., l \tag{6}$$

The difference between a hard margin and soft margin classification is illustrated in Figure 3. In hard margin classification, no data points are allowed to lie within the margin or on the wrong side of the decision boundary, i.e., no misclassification is allowed. On the other hand, in soft margin classification misclassifications are allowed.

4.1.2 The Kernel Trick

A linear SVM can be quite inflexible in low-dimensional spaces, as it can only separate data points using a line. However, if a linear decision boundary does not exist, the data can be mapped into a higher dimensional feature space instead, where a linear decision boundary can be found. This is achieved using a mathematical trick called *kernel trick*. The kernel trick allows to learn a classifier in a higher-dimensional feature space without having to compute the larger representation of the data points. Instead, the distances in the higher-dimensional space between data points are calculated directly. The difference between a linear SVM and a kernelized SVM is illustrated in Figure 4, which shows the found decision boundaries for an example dataset that is not linearly-separable. The kernelized SVM finds a non-linear decision boundary that successfully separates the data points of the different classes.



(a) linear SVM (b) kernelized SVM

Figure 4: On the left, the decision boundary found by a linear SVM for a non-linearly separable example dataset is shown. The triangles and circles represent the associated classes of the data points. On the right, the same decision boundary is shown as a function of the original two features using the kernel trick. (Müller and Guido, 2016)

Gaussian kernel

There are different kernel functions for applying the kernel trick. A popular example is the *radial basis function* (RBF), also called the *Gaussian kernel*. Let x_1 and x_2 be two data points. The hyperparameter γ controls the width of the Gaussian kernel, i.e., how far the influence of a single training example reaches. Low values for γ mean that the influence reaches far and high values correspond to a low reach. Using the Gaussian kernel, the distance between x_1 and x_2 is measured as follows:

$$k_{\text{rbf}} = \exp(-\gamma||x_1 - x_2||^2) \tag{7}$$

Liebeck et al. (2016) and Romberg and Conrad (2021) applied a kernelized SVM with a Gaussian kernel for the classification of argument component types, which is why a kernelized SVM with a Gaussian kernel is also chosen in this work.

4.1.3 Regularization

The SVM classifier is regularized with a trade-off hyperparameter C in order to avoid overfitting. Especially in higher dimensions, guarding against overfitting becomes more important. A lower value for C leads to a larger margin, but more data points are allowed to be outside the own class boundary. On the other hand, a higher value for C puts more importance on classifying each data point correctly, leading to a smaller margin. However, in the case of a SVM with a Gaussian kernel, the behavior of the model is also sensitive to the hyperparameter γ. If γ is set too large, then in the worst case the influence of a support vector would not reach any further training examples and thus regularization with the hyperparameter C would not help to avoid overfitting.

The impact of the hyperparameters C and γ using a SVM with a Gaussian kernel is shown in Figure 5 for an example dataset. A small value for C leads to a more restrictive model, where the data points have less influence on the created decision boundary. On the other hand, a higher value for C allows the data points to have more influence and bend the

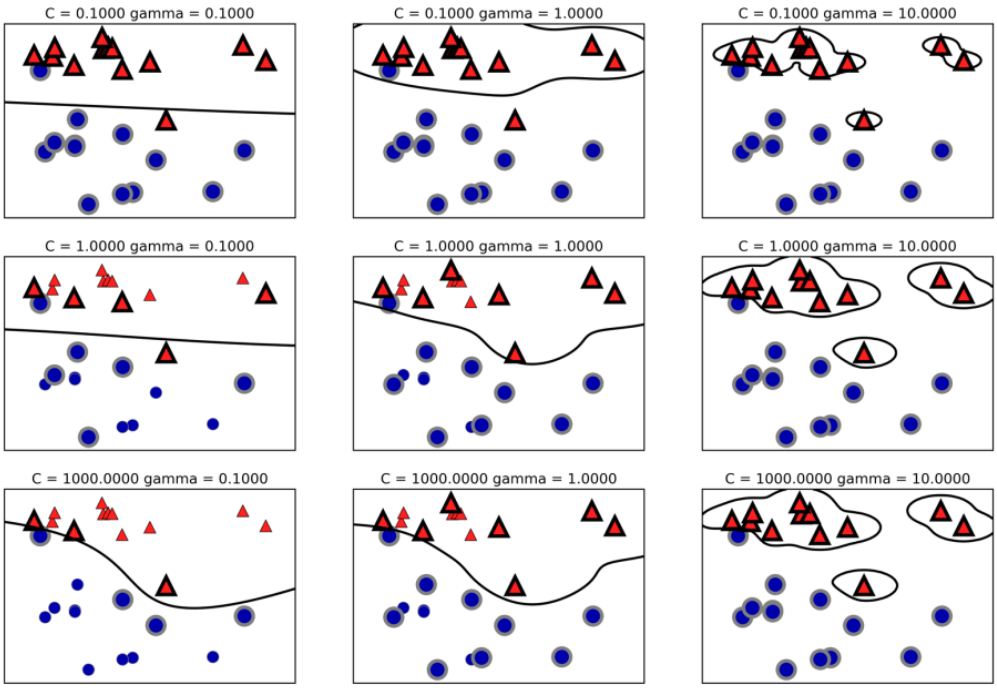


Figure 5: Decision boundaries and support vectors for different settings of the hyperparameters C and γ using the Gaussian kernel. The triangles and squares represent the associated classes of the data points. Support vectors are marked with a thick rim. (Müller and Guido, 2016)

decision boundary. A smaller γ value leads to a larger radius for considering points, creating a model with smooth decision boundaries, whereas a larger value leads the model to pay more attention to single points.

Since C and γ influence the found decision boundaries by the SVM model and thus have a strong impact on the performance, the goal is to find an optimal setting for those hyperparameters. Therefore, different settings for C and γ will be tested in experiments, relying on the considered values by Romberg and Conrad (2021).

4.2 BERT

BERT (Devlin et al., 2019) stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers and is a *transformer*-based language representation model developed by Google AI in 2018. The basic concept of BERT is a two-step training approach: In the *pre-training* step, deep bidirectional representations from unlabeled text are trained over different pre-training tasks. In the *fine-tuning* step, the pre-trained model is first initialized with the parameters obtained from pre-training, which are then fine-tuned using labeled data in order to adapt the model to a specific NLP task by adding an additional task-specific layer.

The advantage of this two-step training approach is that it reduces the need for developing elaborate task-specific architectures and that only few parameters need to be learned

from scratch. Moreover, BERT achieved state-of-the-art results for various NLP tasks such as question answering and language inference, outperforming many task-specific architectures (Devlin et al., 2019). Furthermore, Romberg and Conrad (2021) applied BERT for the single-label classification of major positions and premises, achieving highly promising and stable results over the different datasets. Thus, BERT is selected to be applied on our multi-label classification task as well.

In the following, the model architecture of BERT, the input and output representations, the pre-training step and the fine-tuning step are described in more detail based on Devlin et al. (2019). Furthermore, different German BERT and BERT-based models are selected for this work.

4.2.1 BERT Model Architecture

A characteristic of BERT is that the pre-trained model architecture and the final model architecture have only minimal differences. BERT's architecture is based on a multi-layer bidirectional transformer encoder and its implementation is almost identical to the original implementation of a transformer, which is described by Vaswani et al. (2017). Devlin et al. provide two models of different sizes: BERT-base and BERT-large. BERT-base consists of 12 encoder layers stacked on top of each other, while BERT-large consists of 24 of such layers. Moreover, BERT is able to represent both a single sentence and a pair of sentences as an input. In the following, the original model architecture of a transformer is described in more detail.

Transformer
A transformer is a neural sequence transduction model. It takes a text as an input sequence and outputs another text, e.g., for translation tasks. Like most other sequence transduction models, transformers are based on an encoder-decoder structure, containing a stack of N=6 identical encoder layers and a stack of N=6 identical decoder layers. The architecture of a transformer with its encoder and decoder stacks is illustrated in Figure 6.
First, the transformer converts an input sequence of symbol representations $(x_1, ..., x_n)$ into embeddings using learned embeddings. Then, the embeddings are fed to the encoder stack, which maps them to a sequence of continuous encoded representations $z = (z_1, ..., z_n)$ that captures the meaning and position of each word. Next, the encoded representation z is fed into the decoder stack, which processes it into an output sequence $y = (y_1, ..., y_m)$ of symbols. At each step, the transformer considers the previously generated output symbols for producing the next output symbol.
In contrast to other sequence transduction models, transformers are solely based on *attention* mechanisms. Attention is a technique that enables a model to focus on other parts of the sequence that are closely related and relevant while processing a word. An attention layer performs an *attention function*, which maps a query and a set of key-value pairs, all represented as vectors, to an output vector. In particular, the used attention function in the transformer is called *scaled dot-product attention*.
Let Q, K and V be sets of queries, keys and values, packed into matrices. Let d_k be the dimension of the queries and keys, and d_v the dimension of the values. The output of the scaled dot-product attention function is a vector containing weights for each value and

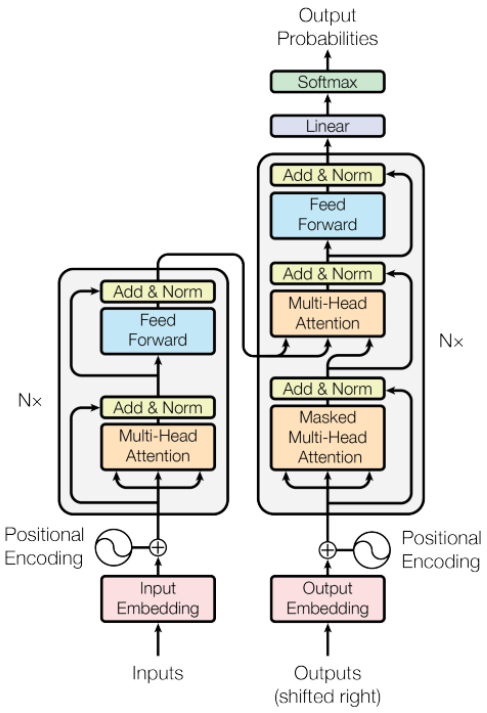


Figure 6: The transformer model architecture. The encoder stack is represented on the left and the decoder stack on the right. Nx (N-times) indicates multiple blocks. (Vaswani et al., 2017)

is computed as follows:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{8}$$

In words, the scaled dot-product attention computes the dot-products of the query with all keys and divides each dot-product by $\sqrt{d_k}$. Then, a *softmax function* is applied, which calculates the weights on the values.
Let x be a vector of real numbers of length n. The softmax function, also called *normalized exponential function*, takes an entry x_i of the vector x as an input and outputs a probability (a real number between 0 and 1), while the sum of all outputs for each entry of x is 1: (Zeng et al., 2020)

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{i=1}^{n} x_i} \tag{9}$$

Instead of performing only a single attention function by one attention head, it is more advantageous to perform several similar attention calculations by multiple attention heads and combining their information. This technique is called *multi-head attention*. In this technique, the queries and keys are each linearly projected h times with learned projections to d_k, and the values h times to d_v. On each of these projections an attention function is performed in parallel, creating d_v-dimensional outputs, which are then concatenated and projected again. The scaled dot-product attention and multi-head attention

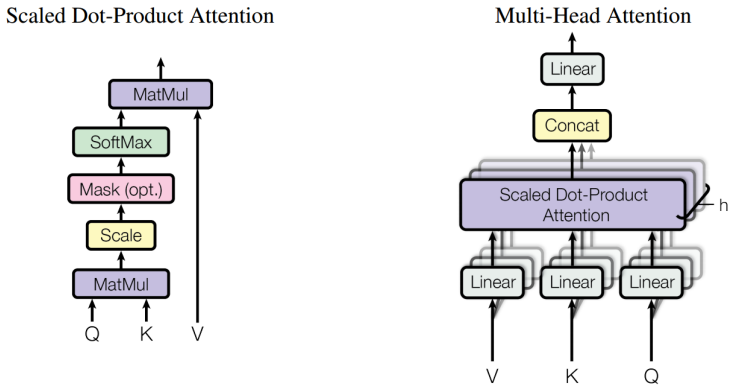


Figure 7: On the left, scaled dot-product attention is illustrated, which performs only a single attention. On the right, multi-head attention is shown, which consists of h attention heads running in parallel. (Vaswani et al., 2017)

are illustrated in Figure 7.
Each layer of the encoder stack of the transformer (cf. Figure 6) has two sub-layers consisting of a multi-head attention mechanism and a simple feed-forward network. The layers of the decoder stack include a third sub-layer in addition to the two sub-layers, which performs multi-head attention over the output of the encoder.

4.2.2 BERT Input and Output Representation

As mentioned before, BERT is able to represent both a single sentence and a pair of sentences, e.g, in the form ⟨question, answer⟩ for question answering tasks. In the case of classification tasks, only single sentences are considered. Note that here a *sentence* refers to any text sequence.

To the beginning of each input, BERT adds a special *classification token* `[CLS]`. The last hidden state of BERT corresponding to this token is used for classification tasks. In the case of sentence pairs as an input, the sentences are put together in a single input sequence using a special *separator token* `[SEP]` to mark the end of each sentence. Moreover, BERT tokenizes the input text. Each token of a sentence is represented as a sum of the token embedding, the segmentation embedding and the position embedding of the corresponding token, which is visualized in Figure 8 given a pair of example sentences as an input. For the token embeddings, BERT uses WordPiece embeddings (Wu et al., 2016), which has a vocabulary of 30,000 tokens. The segment embedding of a token indicates, which sentence the token belongs to, whereas the position embedding shows at which position of the input text the token occurs. The symbols `##` before a token indicate that the corresponding token is a subtoken belonging to the token before, e.g., in the example, the original token `playing` was tokenized into the subtokens `play` and `##ing`.

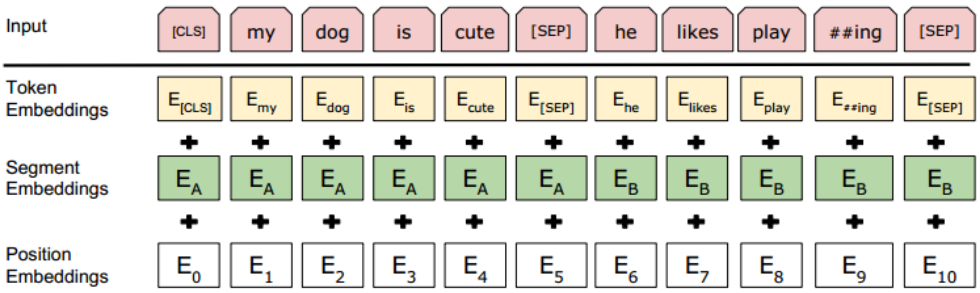


Figure 8: Input representation of BERT. E is the input embedding. Each token of a sentence is represented as a sum of the token embeddings, the segmentation embeddings and the position embeddings. (Devlin et al., 2019)

4.2.3 Pre-Training BERT

In the pre-training step, BERT is trained on a large corpus using two unsupervised tasks: *masked language model* and *next-sentence prediction*. The original BERT implementation is trained on the BooksCorpus data (Zhu et al., 2015) and on English Wikipedia[5] texts, making a total of about 3,300M words. The pre-training step is illustrated on the left side of Figure 9. In the following, the unsupervised training methods are further explained.

Masked Language Model (MLM)
In the *masked language model* (MLM) task, the model randomly masks some percentage of the input tokens, replacing the original token with a `[MASK]` token. Then, the model aims to predict the vocabulary id of the masked token based only on its context. This method allows the model to fuse the left and the right context and produce deep bidirectional representations, capturing the relationships between the words. Furthermore, in the masking procedure some words are kept unchanged, which helps to bias the representation towards the actual token. The masking procedure forces the model to keep a distributional contextual representation of every input token since the model does not know which tokens have been replaced by random tokens and which tokens it will have to predict.

Consider the following example sentence taken from the dataset CQ_B (cf. Table 1 in Section 2.2) as an input to the MLM model:

```
[CLS] Das gibt oft gefährliche Situationen . [SEP]
```

For the masking procedure, 15% of the input tokens are randomly chosen (e.g., `oft`), of which 80% are replaced with the `[MASK]` token. For example, in case the chosen token is replaced with `[MASK]`, then the example input looks as follows:

[5] https://www.wikipedia.org

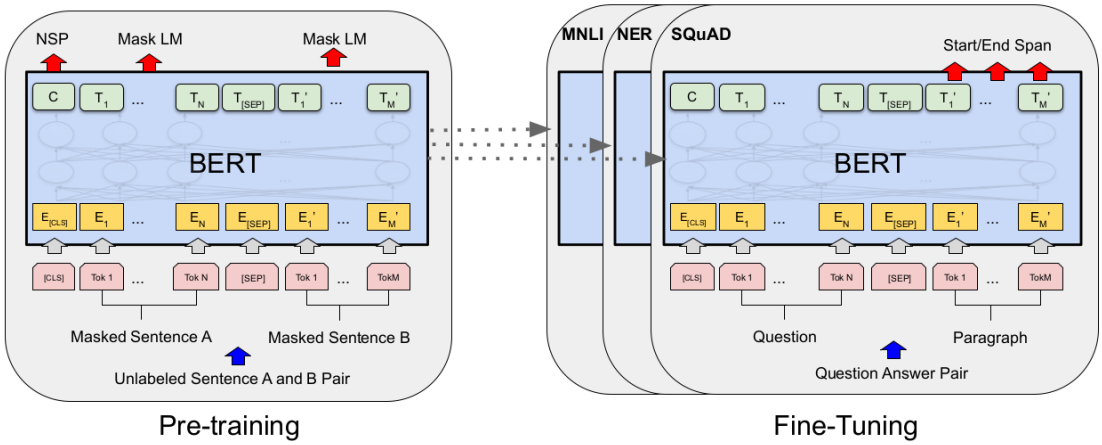


Figure 9: Pre-training (on the left) and fine-tuning (on the right) procedures for BERT. The architectures used for pre-training and fine-tuning have the same structure, except for the output layer. Each down-stream task, e.g., named entity recognition (NER) or question answering (sQuAD), has a separate fine-tuned model, although they are all initialized with the same pre-trained hyperparameters. (Devlin et al., 2019)

```
[CLS] Das gibt [MASK] gefährliche Situationen . [SEP]
```

10% of the randomly chosen tokens are replaced with a random word and another 10% stay unchanged. In the following example, `oft` is replaced by the random token `Haus`:

```
[CLS] Das gibt Haus gefährliche Situationen . [SEP]
```

Next-Sentence Prediction
In order to capture the relation between the sentences, a *next-sentence prediction* task is performed. Given two sentences A and B, the model learns to predict whether B is the next sentence after A and returns the label `IsNext` or `NotNext`. This task can be demonstrated with the following example sentences taken from the dataset CQ_B (cf. Table 1 in Section 2.2):

Input 1:
`[CLS] An Ampel [MASK] ##gängen kreu ##zt sich der [MASK] - & Fußgänger ##weg . [SEP] Das gibt [MASK] gefährliche Situationen [SEP]`

Label for Input 1:
`IsNext`

Input 2:
`[CLS] An Ampel [MASK] ##gängen kreu ##zt sich der [MASK] - &`

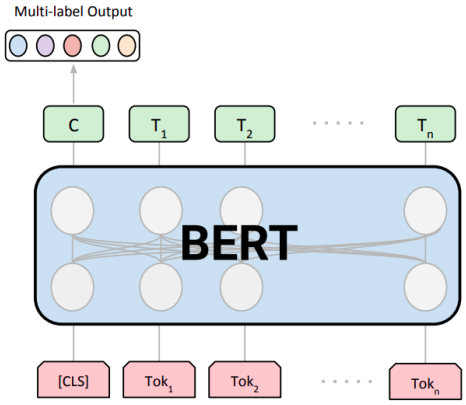


Figure 10: Fine-tuning BERT on a multi-label classification task. (Zahera et al., 2019)

```
Fußgänger ##weg . [SEP] Gera ##dl ##ini [MASK] Weg ##führung !
[SEP]
```

Label for Input 2:
`NotNext`

The two sentences are separated using the `[SEP]` token. In the first example, the model predicts that the second sentence comes next after the first one. In the second example, the model predicts that the second sentence is not the next sentence.

4.2.4 Fine-Tuning BERT

The purpose of the fine-tuning step is to adjust the model to various downstream language processing tasks, e.g., to named entity recognition or question answering, by adding an additional, task-specific output layer. For that, the model is first initialized with the hyperparameter values obtained from pre-training. Then, these parameters are fine-tuned using labeled data for the specific task. Compared to the pre-training step, the fine-tuning step is less expensive.

For classification tasks, which is the present case in this work, the final hidden state corresponding to the special classification token `[CLS]` is used as the aggregate sequence representation. At the output, this representation is fed into a dense layer with a softmax activation function for predicting the class label. However, in order to perform multi-label classification, a dense layer with a *sigmoid activation function* is chosen that returns the probabilities of all classes instead of only the class label with the highest probability among all (Zahera et al., 2019). Then, a threshold can be specified (in this work 0.5) to output the labels that have a probability greater than the specified threshold. Hence, for BERT the use of a problem transformation method is not required in order to perform multi-label classification. The fine-tuning procedure for a multi-label classification task is illustrated in Figure 10.

4.2.5 BERT Models

The original BERT model by Devlin et al. (2019) is trained for English texts, however, in this work only German texts are considered. The most popular German BERT model is GermanBERT[6], which is provided by deepset[7], an open source framework for natural language processing. This model was trained on German Wikipedia texts[8], legal documents from OpenLegalData[9], and news texts. Romberg and Conrad (2021) chose the case-sensitive GermanBERT model, i.e., the input text is not converted to lower case before applying the model.

In late 2020, deepset released another German BERT model called GBERT (Chan et al., 2020), which was additionally trained on the German OSCAR corpus (Suárez et al., 2019) and the OPUS dataset (Tiedemann, 2012), a collection of texts from movies, parliamentary speeches, and books. GBERT is also available in a larger version with significantly more parameters, about three times as many as the original model. However, Chan et al. (2020) have shown that the performance of the larger model was not remarkably higher than the performance of the original model. Also, Chan et al. remarked that the large model trains on much more tokens than the base model due to its different training process and that the advantages of the large model cannot be quantified as easily. Due to these reasons and limited computational resources such as time and memory, only the base model will be considered in this work.

Another considered model in this work is the BERT-based DistilBERT model (Sanh et al., 2019). DistilBERT is significantly smaller and faster than the regular BERT models. The model is learned from the original BERT model using *knowledge distillation* (Hinton et al., 2015), a compression technique for making compact yet fast and highly accurate models. The regular BERT-Base models contain 110M parameters, whereas DistilBERT contains 66M parameters and is thus more than 40% smaller. In general, the architecture of DistilBERT is equal to the original BERT's architecture with some adjustments, e.g., token-type embeddings and the pooler layer are removed. Also, the number of layers in DistilBERT is reduced by a factor of 2. Furthermore, many operations in the transformer architecture are optimized. In an experiment, Sanh et al. (2019) have shown that DistilBERT retains about 97% of the original BERT's performance while being 60% faster. Sanh et al. have also shown that their compressed models are small enough to run on mobile devices. Hence, in this work German DistilBERT[10] will be applied as well in order to compare its performance and efficiency with the regular BERT models.

All of these BERT and DistilBERT models are available on the machine learning platform Hugging Face[11].

[6] https://www.deepset.ai/german-bert
[7] https://www.deepset.ai
[8] https://de.wikipedia.org
[9] https://de.openlegaldata.io/
[10] https://huggingface.co/distilbert-base-german-cased
[11] https://huggingface.co/models

4.3 XGBoost

XGBoost (T. Chen and Guestrin, 2016) stands for e**X**treme **G**radient **B**oosting. *Boosting* describes a method that combines various simple and weak models to make better predictions. Weak models are models that perform slightly better than making random predictions. In boosting, models are added sequentially in a greedy procedure until no more improvements in the predictions occur. In the case of XGBoost, *decision trees* are used as models, which are added by a special gradient algorithm called *gradient descent algorithm*.

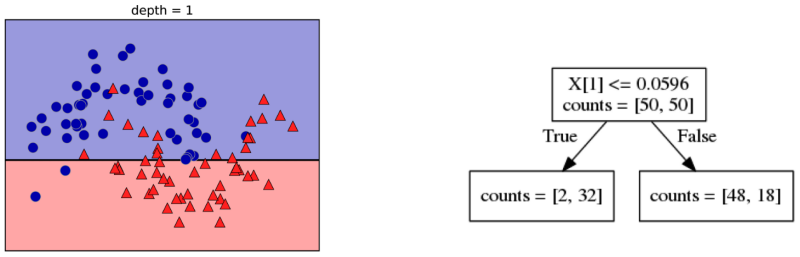
In Schaefer and Stede (2020), XGBoost was applied on German tweets for argument component mining in the domain of climate change. Their argumentation model slightly differs from ours, as they differentiate between claims (a standpoint towards a topic) and evidences (unit that supports or attacks a claim). However, Schaefer and Stede achieved promising results (robust F_1 scores around 0.8). XGBoost is selected to be applied in this work since it would be interesting to find out whether the model can achieve similarly good results for the detection of major positions and premises and whether it can compete with the other selected methods. In order to perform multi-label classification with XGBoost, it is used in combination with the selected problem transformation methods mentioned in Section 3.2.

In the following, the basic concepts of XGBoost such as decision trees, the gradient descent algorithm and two methods for preventing the model from overfitting are explained in more detail based on T. Chen and Guestrin (2016) and Müller and Guido (2016).

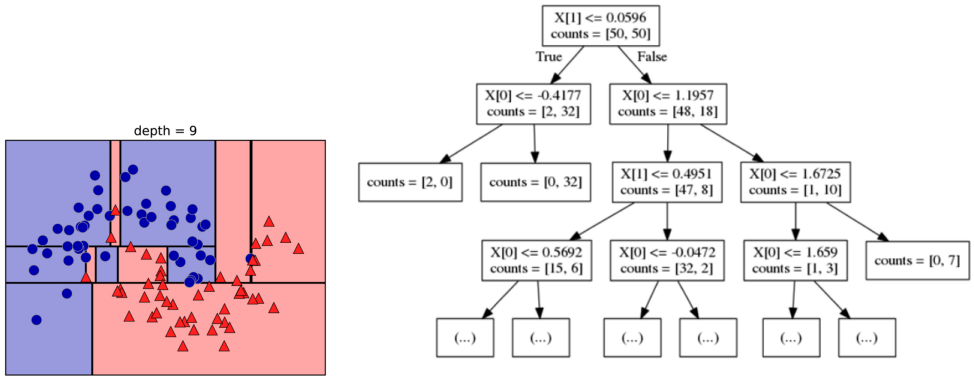
4.3.1 Decision Trees

XGBoost is based on an ensemble of *decision trees*. Decision trees are binary trees, i.e., data structures in which each node has at most two children, which learn simple decision rules in the form of "if/else" questions called *tests* in order to perform classification tasks. For example, a test could be "is feature i greater than value a?". The goal is to learn a hierarchy consisting of as few tests as possible while making correct predictions. During training, the model searches over all possible tests to find the ones that are most informative for the classification.
In a decision tree, the inner nodes represent the tests and the leaves one of the classes. In order to assign an instance to a class, the tree is traversed from the root to a leaf. To build a decision tree during training, the algorithm searches in a recursive process over all possible tests to find those that lead the fastest to the determination of the associated classes.

Figure 11 shows the steps of building a decision tree for an example dataset. The data points are split by a test such that as many data points as possible from the same class lie in the same region. The corresponding decision trees record in the nodes which tests were done to split the data points. In the leaves, the number of data points associated to each class that lie within a partition is recorded. Since both partitions after the first split still contain data points from different classes (cf. Figure 11a), the data points continue to be separated according to this scheme until each partition contains only data points from the same class. Figure 11b shows the final result of the partitioning and the corresponding



(a) Decision boundaries and corresponding decision tree with depth 1.



(b) Final decision boundaries and a part of the corresponding decision tree.

Figure 11: Recursive process of building a decision tree for an example dataset. The recursive process is continued until each partition contains only data points from the same class. The triangles and circles represent the associated classes of the data points. (Müller and Guido, 2016)

decision tree.

4.3.2 Gradient Descent Algorithm

In *boosting*, models are added to an ensemble of learners sequentially until no more improvements in the predictions occur. Each learner learns from the mistakes of the previous learner. *Gradient boosting* is a boosting technique that uses the *gradient descent algorithm* in order to add the models. In the following, gradient boosting for tree-based models is described.

Given a dataset with n examples and m features $\mathcal{D} = \{(x_i, y_i)\}$ ($|\mathcal{D}| = n$, $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$) and an independent set of *regression trees* $\mathcal{F} = \{f(x) = w_{q(x)}\}$ ($w \in \mathbb{R}^T$) with tree structure $q : \mathbb{R}^m \to T$. Tree structure q maps an example to a leaf index. Let T be the number of leaves. Regression trees are decision trees, where the leaves are labeled with a continuous score. Here, w_i represents the score on the ith leaf. An ensemble model consisting of K regression trees uses K additive functions for predicting the output:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{10}$$

Given an example, the final prediction is calculated by summing up the output scores of each regression tree.

Furthermore, the goal is to make the model learn a set of functions that are simple and predictive. Let l be a loss function that is differentiable and convex, which measures the difference between prediction $\hat{y}_i$ and target value y_i. The objective is to minimize the following regularized term:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \quad \text{where } \Omega(f) = \gamma T + \tfrac{1}{2}\lambda ||w||^2 \tag{11}$$

Ω penalizes the complexity of the regression tree functions and helps to smooth the learned weights in order to avoid overfitting. However, the regularization term includes functions as parameters, hence, it cannot be optimized using traditional optimization methods in the Euclidean space. In gradient tree boosting, the model is trained greedily instead by adding functions that minimize $\mathcal{L}(\phi)$ and improve the model the most. Let $\hat{y}_i^{(t)}$ be the prediction of the ith instance at the tth iteration. A function f_t is added to minimize the following term:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{12}$$

This term is minimized using approximation and gradient statistics.

Furthermore, the quality of a tree is determined by using gradient statistics. Let $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ be the first and second order gradient statistics on the loss function l. Let $I_j = \{i | q(x_i) = j\}$ be the instance set of leaf j. In order to measure the quality of a tree structure q at iteration step t, the first order and second order gradient statistics are summed up on each leaf and then the following scoring formula is applied to get the quality score:

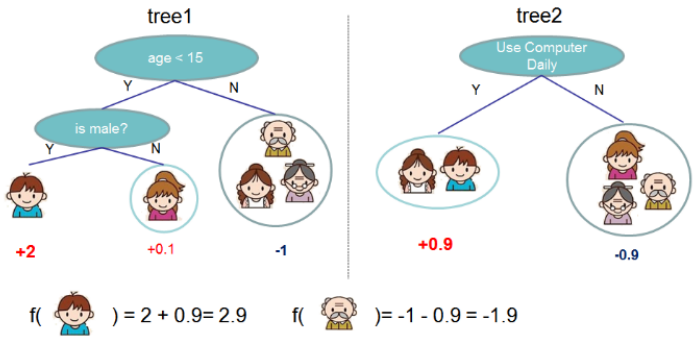
$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{13}$$

The smaller the score is, the better the quality of the tree structure is. The gradient descent algorithm starts from a single leaf and iteratively adds branches to the tree, since it is usually not possible to enumerate all possible tree structures.

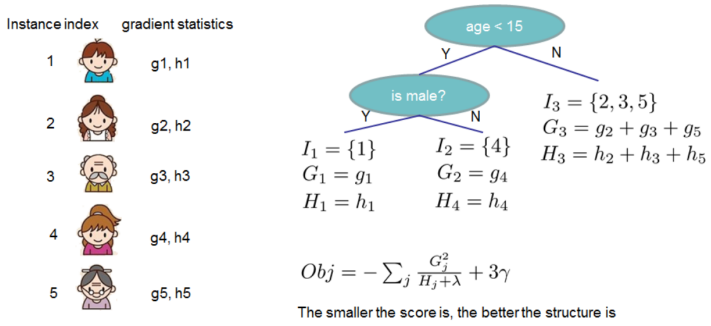
Let I_L and I_R be the instance sets of the left and right nodes after a split (test) and $I = I_L \cup I_R$. For evaluating the split candidates of the trees, the reduction of the loss after splitting is calculated:

$$\mathcal{L}_{\text{split}} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{14}$$

The gradient descent algorithm is demonstrated in 12 based on an example with two classes. The illustrated ensemble learner (cf. Figure 12a) consists of two regression trees. The inner nodes represent the tests and the leaves record the examples (here persons)



(a) Ensemble learner consisting of two regression trees.



(b) Quality score calculation for tree1 of the ensemble learner.

Figure 12: Demonstration of the gradient descent algorithm on an example. (T. Chen and Guestrin, 2016)

that fulfill the same tests. The leaves are labeled with class scores. In order to make a final prediction, e.g., for the young boy, the class scores of the corresponding leaves from both trees are summed up. The positive class score indicates that the boy belongs to a specific class.
The computation of the quality score for tree1 of the example ensemble learner is illustrated in Figure 12b. First, the first order and second order gradient statistics are summed up on each leaf. Then, the scoring formula is applied to measure the quality of the tree structure.

4.3.3 Shrinkage and Column Subsampling

Regression trees tend to overfit the training data quickly. In order to protect the gradient boosted regression trees from overfitting, two additional strategies are used: *shrinkage* and *column subsampling*.

One way to slow down the learning process of each individual tree is the *shrinkage* technique (Friedman, 2002), which applies a weighting factor $0 < \eta \leq 1$ to newly added weights. This method helps to control the learning rate of the individual trees and to leave some space for future added trees in order to improve the model.

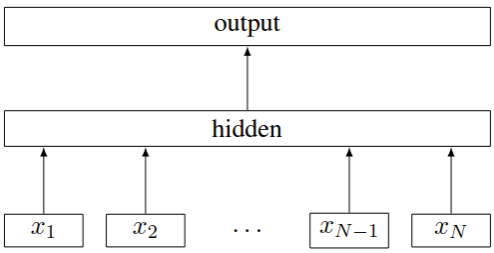


Figure 13: Model architecture of fastText for a sentence, which is represented as N ngram features $x_1, ..., x_N$. The ngram features are embedded and averaged to form the hidden variable. (Joulin et al., 2017)

The *column subsampling* method, also called *feature subsampling*, helps to reduce overfitting by taking a subsample of the columns (corresponding to the features) for fitting each individual tree. This method is also used for random forests (Breiman, 2001).

4.4 Excluded Methods

Aside from BERT and SVM, Romberg and Conrad (2021) also applied the methods fastText and ECGA for the single-label classification of major positions and premises. However, these methods were excluded in this work due to shortcomings. In the following, both methods are presented and their shortcomings discussed.

4.4.1 FastText

FastText (Joulin et al., 2017) is a linear classifier developed by Facebook AI. Given an example sentence, the individual words are first represented as ngram features (cf. Section 5.1). Next, the word representations are averaged to form a representation of the whole sentence, which is a hidden variable. Eventually, the sentence representation is fed to a linear classifier. This procedure is demonstrated in Figure 13.

The goal of the classifier is to minimize the negative log-likelihood over the classes during training: Let A and B be weight matrices and f a softmax function, which computes the probability distribution over the classes. Given N example text inputs, let x_n be the normalized bag-of-ngrams of the nth examples and y_n the label of the nth example. The negative log-likelihood over the classes is defined as follows: (Joulin et al., 2017)

$$-\frac{1}{N}\sum_{n=1}^{N} y_n \log(f(BAx_n)) \tag{15}$$

Romberg and Conrad (2021) used fastText for identifying major positions and premises among argumentative sentences. However, fastText was clearly outperformed by SVM and BERT with an average macro F_1 score of 0.78 in the intra-dataset evaluation. In comparison, BERT and SVM achieved average macro F_1 scores of 0.91 and 0.82. Moreover, the model performed poorly for the minority class of each dataset and undersampling

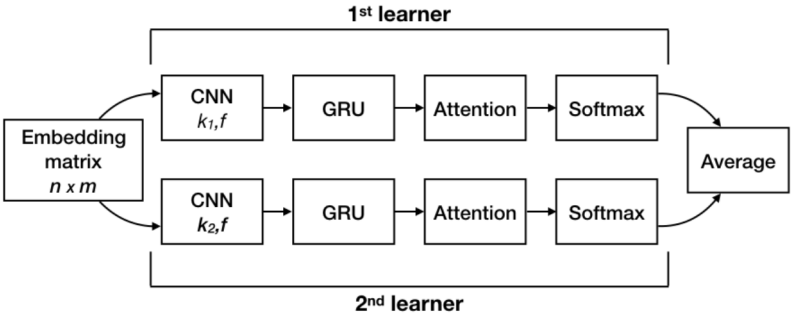


Figure 14: ECGA architecture with two learners. Each learner consists of three neural components: a CNN, a bidirectional GRU and an attention component. The final prediction is calculated by averaging the predictions from the individual learners. (Giannakopoulos et al., 2019)

the majority class did not improve the results. Furthermore, fastText showed weaknesses in generalizing on the datasets CQ_B and MC_K in the cross-dataset evaluation. Hence, fastText will not be further considered in this work.

4.4.2 ECGA

ECGA (Giannakopoulos et al., 2019) stands for **E**nsemble of **C**NN-**G**RU-**A**ttention and is an ensemble classifier consisting of three neural components: convolutional neural networks (Lecun et al., 1998), bidirectional gated recurrent neural units (Li et al., 2017) and attention modules.

Convolutional neural networks (CNNs) are neural networks that were initially developed for image processing. The core component of CNNs is *convolution*. The convolution operation can be interpreted as a filter. A kernel, which is a weight matrix, is slid across the input matrix and produces a convoluted feature output by computing the dot product between the input matrix values that are within the kernel's bounds and the kernel. CNNs work best, when there is some sort of structure in the input data such as repeated patterns. (Patterson and Gibson, 2017)

Bidirectional gated recurrent neural units (BiGRUs) are sequence processing models that consist of two gated recurrent neural units (Cho et al., 2014). A gated recurrent neural unit (GRU) is a gating mechanism that is used in Recurrent Neural Networks (RNNs) to help the model learn dependencies between distant words. One of the GRUs takes the input sequence in a forward direction, while the other one takes it in a backwards direction. (Sammani et al., 2021)

The architecture of an ECGA model is illustrated in Figure 14. First, an input matrix of size $n \times m$, where n is the number of words and m the number of features, is fed into a CNN with f filters of kernel size k, i.e., each filter slides over k words (a k-gram). The output of the filters is then concatenated, creating an matrix of size $(n-k+1) \times f$, where the jth row is a feature of the jth k-gram. Next, the output of the CNN component is given as an input to a BiGRU and subsequently to an attention module, which calculates

a weighted sum of all output states of the BiGRU component. After applying attention, the model puts the output through a softmax layer, attaining a final feature vector of the input text. This procedure is repeated by multiple learners, e.g., with different kernel sizes k_i, in order to train with diverse features and improve the performance. The final prediction of the ensemble model is calculated by averaging the predictions from the individual learners. (Giannakopoulos et al., 2019)

ECGA profits from the usage of the three different neural components and reduces overfitting since it is an ensemble classifier and the final prediction is done by averaging the results of the individual learners (Giannakopoulos et al., 2019). However, in Romberg and Conrad (2021), ECGA was clearly outperformed by SVM and BERT for the single-label classification task of identifying major positions and premises. The model achieved an average macro F_1 score of 0.77 in the intra-dataset evaluation, whereas BERT and SVM reached an average macro F_1 score of 0.91 and 0.82. Similar to fastText, the model also showed weaknesses in generalizing on the datasets CQ_B and MC_K and recognizing the minority class of the datasets. Hence, ECGA will not be further considered in this work.

5 Feature Extraction

Different machine learning algorithms require different input formats. A feature is a numerical representation of information from data that can be passed to the classifiers. There are different types of features and different ways, features can be extracted from text data. For the selected methods BERT and DistilBERT, the required input format described in Section 4.2.2 is obtained by applying the models' tokenizers, which are all available on the machine learning platform Hugging Face[12]. Hence, for these models no further feature extraction step is required. Regarding the chosen classifiers SVM and XGBoost, we consider *bag-of-ngrams*, the distribution of *POS-tags* and the distribution of *dependencies* in the input texts as features in our experiments. In the following, the considered features are described.

5.1 Bag-of-ngrams

Bag-of-ngrams featurization is a commonly used method for representing text data for machine learning algorithms. An *ngram* is a sequence of n tokens, e.g., a single word is a 1-gram, also called *unigram*, and a sequence of two words is a 2-gram, also referred to as *bigram*. (Zheng and Casari, 2018)

For example, tokenizing the example sentence "Engstelle führt direkt auf Gleise" (engl. "Narrow passage leads directly onto tracks") from the cycling dialogue dataset CD_B would generate the following unigrams:

```
['Engstelle', 'führt', 'direkt', 'auf', 'Gleise']
```

On the other hand, tokenizing the example sentence into bigrams would generate:

```
['Engstelle führt', 'führt direkt', 'direkt auf', 'auf Gleise']
```

A *bag-of-ngram* is created by the following three steps: First, each document is tokenized into ngrams. Second, a vocabulary is built based on the ngrams and third, each document is converted into a vector of counts that contains an entry for every ngram in the vocabulary. For example, if an ngram occurs t times in a document, then the feature vector has a count of t in the position corresponding to that ngram. A bag-of-ngram representation in the case of unigrams is also called *bag-of-words*.
The larger n is chosen, the more of the original sentence structure and semantic meaning is maintained, since more context is given. Hence, the generated ngrams can be more informative. However, the larger n becomes, the more unique ngrams will be generated and the sparser the bag-of-ngrams feature will become. (Müller and Guido 2016; Zheng and Casari 2018)

Rescaling with Tf-idf
Tf-idf stands for **t**erm **f**requency - **i**nverse **d**ocument **f**requency and is a commonly used

[12] https://huggingface.co/models

measure to normalize the frequencies of tokens for a bag-of-ngrams model. The goal of rescaling the model with the tf-idf measure is to reduce the influence of terms that occur in many documents, e.g., of pronouns and conjunctions, as these terms usually are less discriminative than those that occur in fewer documents. Terms that appear often in a particular document but only in few documents of the whole corpus are given a higher weight, since they are more likely discriminative and descriptive of the document's content. (Müller and Guido, 2016)

Let N be the total number of documents in the corpus and $\text{df}(t)$ the *document frequency* of term t, i.e. the number of documents, in which t occurs. The *inverse document frequency* is a measure for determining the informativeness of a term. The inverse document frequency of term t, $\text{idf}(t)$, is originally defined as follows: (Salton and Buckley, 1988)

$$\text{idf}(t) = \log(\frac{N}{\text{df}(t)}) \tag{16}$$

However, it is common to add 1 to the denominator in order to avoid zero division. The implemented version of tf-idf in the Python library scikit-learn[13] also adds 1 to $\text{idf}(t)$ so that terms with $\text{idf}(t) = 0$, i.e., terms that occur in all documents, will not be entirely ignored. Moreover, the implemented version adds the constant 1 to both, the numerator and denominator, pretending that an extra document was seen, in which every term of the vocabulary occurs exactly once in order to smooth out the output:[14]

$$\text{idf}(t) = \log(\frac{1+N}{1+\text{df}(t)}) + 1 \tag{17}$$

For example, if a term t occurs in many documents, then $\text{idf}(t)$ returns a value close to 1. On the other hand, if t occurs rarely, then $\text{idf}(t)$ increases.

Let $\text{tf}(t, d)$ be the *term frequency* of term t in document d, i.e., the number of occurrences of t in d. The *tf-idf score* of term t in document d is defined as the product of $\text{tf}(t, d)$ and $\text{idf}(t)$:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t) \tag{18}$$

A high value for $\text{tf-idf}(t, d)$ is achieved when the frequency of token t in document d is high, but the token occurs overall in very few documents from the given set of documents.

Lastly, the vector representations of the computed tf-idf scores are normalized by the *Euclidean norm*, also called L_2*-norm*, such that the vector representation of each document has the Euclidean length 1. A vector v is normalized as follows:

$$v_{\text{norm}} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + ... + v_n^2}} \tag{19}$$

Liebeck et al. (2016) used ngrams for the classification of argument component types and state that ngrams are able to capture the text content well because certain words are

[13] https://scikit-learn.org/

[14] https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting

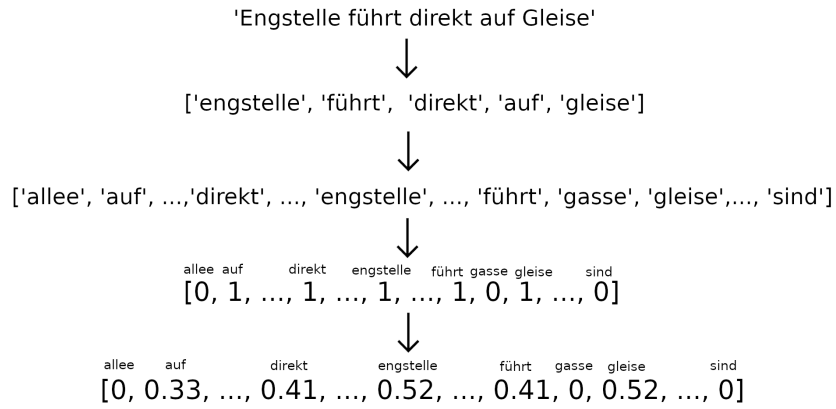


Figure 15: Demonstration of the tf-idf rescaled bag-of-words featurization applied in this work on the example corpus from Table 11. First, each document is tokenized into lowercased unigrams. Second, a vocabulary over all documents is built based on the unigrams. Third, each document is converted into a vector of counts that contains an entry for every unigram in the vocabulary. Lastly, the vector of counts are rescaled with tf-idf and normalized by the Euclidean norm.

used repeatedly in sentences belonging to the same argument component type. For their classification task, they tested unigrams and bigrams for the bag-of-ngrams features and achieved the best results using lowercased unigrams. Romberg and Conrad (2021) used unigrams as well, relying on the results by Liebeck et al. Thus, in this work lowercased unigrams are chosen. Furthermore, in this work tf-idf rescaling was applied in order to normalize the bag-of-words representation, i.e., each token was weighted according to its tf-idf score. The rescaled bag-of-words representation was then used as a feature for the SVM and XGBoost models.

Consider the following example corpus shown in Table 11 that consists of four documents (here sentences):

id	content
1	“Engstelle führt direkt auf Gleise”
2	“Radweg führt direkt in eine Gasse”
3	“Auf der Fahrbahn sind keine Markierungen”
4	“Risse auf der Kempener Allee”

Table 11: Example corpus consisting of four sentences.

Creating the bag-of-words representation for the example corpus and rescaling the representaion with scikit-learn’s TfidfVectorizer[15] is demonstrated in Figure 15. As earlier mentioned, lower-cased unigrams were chosen for the bag-of-words representation. In

[15] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer

the first example sentence, the unigrams "engstelle" and "gleise" have a higher tf-idf score compared to the remaining unigrams since they occur only once in the whole corpus.

In order to further improve the performance of the models and reduce the size of the bag-of-words feature, unigrams that occurred very frequently or extremely infrequently were eliminated from the built vocabulary. For that, unigrams with a document frequency higher than 40% and unigrams with a document frequency lower than 0.1% were removed after experimenting with different values.

5.2 Grammatical Distributions

Liebeck et al. (2016) observed that the user comments in their public participation data contain different tenses and sentence structures for different argument component types. In order to capture this information, Liebeck et al. (2016) and Romberg and Conrad (2021) represented sentences as a L_2-normalized *POS-tag* distribution and a L_2-normalized distribution of *dependencies* and used these representations in addition to bag-of-words as features for their classification models. Hence, in this work the L_2-normalized POS-tag distribution and the L_2-normalized distribution of dependencies were considered as features for the classifiers SVM and XGBoost as well. In the following, POS-tagging and dependency parsing are explained in more detail.

POS-tagging

POS-tags are pre-defined **p**art-**o**f-**s**peech categories and *POS-tagging* describes the process of assigning POS-tags to each word of a text. The Stuttgart Tübingen Tagset (STTS) by Schiller et al. (1999) consists of over 50 POS-tags and represents a quasi-standard for the German language (Rehbein and Schalowski, 2013).

Consider the example sentence "Sollen Radler*innen hier wirklich warten müssen?" (engl. "Shall cyclists really have to wait here?") from the cycling dialogue dataset CD_B. Applying the natural language processing library spaCy[16] for POS-tagging returns the following STTS POS-tags for the given example sentence (given in parentheses):

```
Sollen(VMFIN) Radler*innen(NN) hier(ADV) wirklich(ADJD)
          warten(VVINF) müssen(VMINF) ?($.)
```

For instance, the POS-tag `ADV` stands for an adverb, `NN` for a noun and `$.` for a sentence-final punctuation mark.

Dependency parsing

Dependency parsing is the process of analyzing the grammatical structure of a sentence and finding relations between the words.

Given the example sentence from earlier, spaCy assigns the following dependency tags using the TIGER treebank annotation scheme (Brants et al., 2002):

[16] https://spacy.io/

```
Sollen(ROOT) Radler*innen(sb) hier(mo) wirklich(mo) warten(oc)
                    müssen(oc) ?(punct)
```

For example, the dependency tag `sb` stands for subject, `mo` for modifier and `oc` for clausal object.

6 Experimental Setup

In this section, the experimental setup for evaluating the different models is described. First, the used evaluation metrics for the multi-label classification task are defined. Next, the data partitioning strategy *k-fold cross-validation* and the hyperparameter-tuning technique *grid search* are explained. Furthermore, a required data preprocessing step was conducted before applying the proposed methods. Lastly, the considered features for SVM and XGBoost and combinations of them are tested in experiments before selecting the most suitable feature or feature combination for these methods.

6.1 Evaluation Metrics

In multi-label classification, the evaluation slightly differs from traditional single-label classification since it has to be considered that multiple labels can be assigned to an example and thus predictions can be partially correct. For the evaluation of the models, the commonly used metrics *accuracy*, F_1 *score* and *hamming loss* were considered.

In order to define the metrics, first the different types of predictions or errors in a multi-label classification task have to be explained: A *false positive* is a sample, for which the classifier predicted a label that is incorrect, whereas a *false negative* is a sample, for which a true label was missed by the classifier. Moreover, a sample is referred to as a *true positive* if the classifier correctly assigned a label. On the other hand, a sample is called *true negative* if the classifier correctly predicted the non-existence of a label. The number of false positives, false negatives, true positives and true negatives of the predictions will be denoted as FP, FN, TP and TN respectively. (Lentzas et al., 2022)
In the following, the metrics are defined based on Lentzas et al. (2022) and Müller and Guido (2016) and discussed.

6.1.1 Accuracy

Accuracy is defined as the fraction of correctly predicted samples. In multi-label classification, a prediction is considered correct, if the entire set of predicted labels matches with the set of true labels. The accuracy score is computed as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

The metric takes values between 0 and 1, with 1 being the optimal value. Accuracy is the most intuitive metric, however, this score does not provide information about how well each individual class is predicted by the classifier. Particularly in the case of highly imbalanced datasets, which is the case for the used datasets in this work, a high accuracy does not necessarily mean that the classifier recognizes each class equally well. Moreover, in the case of a multi-label classification task, the accuracy metric does not give credit to partially correct predictions, i.e., when a part of the labels were assigned correctly. Therefore, this score will not be used for evaluating the models.

6.1.2 F_1 Score

In order to introduce the F_1 score, first the metrics *precision* and *recall* have to be defined. Precision is the proportion of correct predictions among all predictions of a certain class, i.e., it measures how many of the assigned instances to a class actually belong to it:

$$\text{precision} = \frac{TP}{TP + FP} \tag{21}$$

Recall measures the proportion of examples of a certain class that have been recognized correctly by the classifier as being part of the class:

$$\text{recall} = \frac{TP}{TP + FN} \tag{22}$$

The F_1 *score* is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{23}$$

Since the F_1 score takes both precision and recall into consideration, it is often chosen as an overall measure for evaluating a classifier. Precision, recall and F_1 score take values between 0 and 1, with 1 being the optimal value. The average F_1 score over all classes is called *macro* F_1. However, a disadvantage of the F_1 score in case of a multi-label classification task is that it does not give credit to partially correct predictions, either. Therefore, in addition to the F_1 score, another metric should be used for evaluating the models, which takes partially correct predictions into account.

6.1.3 Hamming Loss

Hamming loss is the average *hamming distance* between two samples and takes partially correct or incorrect predictions into account by calculating the fraction of labels that are predicted incorrectly to the total number of labels. The hamming distance between two sets equals the symmetric difference between them.

Let N be the total number of samples and L the number of labels. Let $y_{i,j}$ be the target, i.e., if the jth label belongs to the ith sample, then $y_{i,j} = 1$, otherwise, $y_{i,j} = 0$. Let $z_{i,j}$ be the classifier's prediction, whether the jth label belongs to the ith sample. The hamming loss is computed as follows:

$$\text{Hamming} = \frac{1}{N \cdot L} \sum_{i=1}^{N} \sum_{j=1}^{L} \oplus(y_{i,j}, z_{i,j}) \tag{24}$$

$\oplus$ is the *xor operator*, i.e., if $y_{i,j} = z_{i,j}$ holds, then $\oplus(y_{i,j}, z_{i,j}) = 0$, otherwise, $\oplus(y_{i,j}, z_{i,j}) = 1$. Hamming loss also takes values between 0 and 1. In contrast to the other metrics, 0 is the optimal value since hamming loss is a loss function.

Since hamming loss takes partially correct predictions into account, it is an important metric for evaluating multi-label classification tasks. Hence, hamming loss was used together with the F_1 metric in order to evaluate our multi-label classification models.

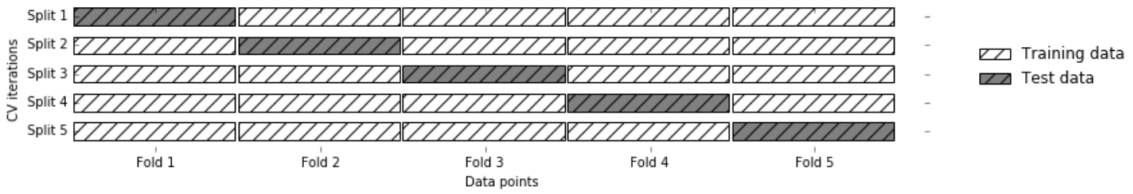


Figure 16: Train and test splits of a 5-fold cross-validation. (Müller and Guido, 2016)

6.2 k-fold Cross-Validation

Splitting data into only one train and test set can lead to overfitting and produce unreliable results, particularly in the case of a limited amount of data. *Cross-validation* is a commonly used data partitioning strategy that helps to evaluate a model in a more generalizable way by resampling the dataset several times using different subsets of the dataset for training and testing.

In a *k-fold cross-validation*, the dataset is partitioned into k subsets of approximately the same size, which are called *folds*. The first model uses the first fold as a test set and the remaining $k-1$ folds for training. The second model uses the second fold as a test set and the remaining folds for training. This procedure is repeated k times until each of the k folds is used for testing once. In total, k models are trained and evaluated. The results of each evaluated model can then be averaged in order to obtain more generalizable results. (Müller and Guido, 2016)

Usually, for k the values 5 or 10 is chosen. Romberg and Conrad (2021) applied a 5-fold cross-validation for evaluating the methods on each dataset. Hence, in this work 5 was chosen for k as well. The data-splitting process of a a 5-fold cross-validation is illustrated in Figure 16.

6.3 Grid Search

Hyperparameters are the parameters of a model that are specified by the user and not learned in the learning process. The values of the hyperparameters have an influence on the performance of the models, hence it is important to adjust them. The goal is to find the configuration of the hyperparameters that produces the best results.

Grid search is a commonly used method for hyperparameter-tuning, in which the selected hyperparameters are set with different values and each possible combination is evaluated. In this work, grid search was applied on each train set in order to obtain a model with optimized hyperparameters. In order to evaluate the different hyperparameter configurations, the actual train set was split once again in train and validation sets. For the SVM and XGBoost models, a 3-fold cross-validation was applied for splitting the train set. For the BERT and DistilBERT models, the train set was split only once in a train and validation set (80:20) due to high computational costs and limited resources. After obtaining the model with the best hyperparameter configuration from grid search, the model is then applied and evaluated on the actual test set.

	CD_B	CD_C	CD_M	CQ_B	MC_K	all
total	9,289	1,507	1,811	1,333	1,577	15,517
mpos	2,589 (27.9%)	556 (36.9%)	359 (19.8%)	960 (72.0%)	892 (56.6%)	5,356
premise	6,438 (69.3%)	904 (60.0%)	1,407 (77.3%)	250 (18.8%)	616 (39.1%)	9,615
mpos+premise	262 (2.8%)	47 (3.1%)	45 (2.5%)	123 (9.2%)	69 (4.4%)	546

Table 12: Distribution of sentences among the different argument component categories per dataset after removing non-argumentative sentences.

Romberg and Conrad (2021) applied grid search as well. For the kernelized SVMs with a Gaussian kernel, they tuned the regularization parameter C and the kernel coefficient γ, choosing values from $C \in \{1, 10, 100\}$ and $\gamma \in \{0.001, 0.01, 0.1\}$. For the BERT models, they tuned the *batch size*, the number of *epochs* and the *learning rate*. The batch size is the number of samples that are processed before the model weights are updated and the number of epochs is the number of complete passes through the entire train dataset. The learning rate controls how strong the model reacts and adjusts its weights in response to prediction errors. Devlin et al. (2019) suggest the batch sizes 16 or 32, maximum 4 epochs and the learning rates 2e-5, 3e-5 or 5e-5 for BERT. These values were also considered by Romberg and Conrad.

In this work, we optimized the mentioned hyperparameters for the SVM and BERT models relying on the same hyperparameter values chosen by Romberg and Conrad. For the DistilBERT models, we also rely on the same hyperparameter values as for the BERT models since the developers Sanh et al. (2019) did not suggest any specific values. For XGBoost, the number of regression trees, the learning rate and the maximum depth of the trees was optimized. Choosing too large values for the number of trees and the maximum depth increases the computational costs and may lead to overfitting, while choosing too low values may lead to underfitting. The developers T. Chen and Guestrin (2016) did not recommend any specific hyperparameter values for XGBoost. After experimenting with different hyperparameter values, the number of trees 200 and 300, the learning rates 0.1 and 0.2 and the maximum depths 8 and 10 were selected for grid search.

6.4 Dataset

Since for the classification of major positions and premises only argumentative sentences are considered, non-argumentative sentence were filtered out. Table 12 shows the distribution of the sentences among the argument component categories for each dataset after filtering out non-argumentative sentences. The dataset CD_B is significantly larger than the remaining datasets with a total number of 9,289 sentences. In contrast, the remaining datasets comprise between 1,333 and 1,811 sentences.

As already mentioned in Section 2.2, the frequencies of the different argument component classes are noticeably imbalanced in each dataset, which may affect the ability of the applied classification algorithms to recognize the minority class negatively. In the datasets CD_B, CD_C and CD_M, premise forms the majority class, whereas in CQ_B and MC_K the class major position (mpos) occurs most frequently. Furthermore, the multi-label *mpos+premise* is highly under-represented in contrast to the single labels.

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW	0.12	0.85	0.12	0.86
BOW + POS	**0.11**	**0.86**	0.12	0.86
BOW + DEP	0.12	**0.86**	0.12	0.86
BOW + POS + DEP	**0.11**	**0.86**	0.12	0.86

(a) Results for label power-set.

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW	0.12	0.86	0.12	0.86
BOW + POS	0.12	0.86	0.12	0.86
BOW + DEP	0.12	0.86	0.12	0.86
BOW + POS + DEP	0.12	0.86	0.12	0.86

(b) Results for binary relevance.

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW	0.12	0.86	0.12	0.86
BOW + POS	0.12	0.86	0.12	0.86
BOW + DEP	0.12	0.86	0.12	0.86
BOW + POS + DEP	0.12	0.86	0.12	0.86

(c) Results for classifier chains.

Table 13: Comparison of different feature combinations for SVM and XGBoost using different problem transformation methods. The models were trained on the largest dataset CD_B for the multi-label classification of major positions and premises among argumentative sentences. Considered features were the tf-idf rescaled bag-of-words (BOW), L_2-normalized POS-tag distribution (POS) and L_2-normalized distribution of dependencies (DEP). The scores are averaged for the 5-fold cross-validation.

6.5 Feature Selection

For the classifiers SVM and XGBoost, the tf-idf rescaled bag-of-words representation, the L_2-normalized POS-tag distribution and the L_2-normalized distribution of dependencies were considered as features (cf. Section 5). Romberg and Conrad (2021) used the combination of all three feature types for their SVM models, relying on the suggestion by Liebeck et al. (2016).

Before deciding on the features for SVM and XGBoost, it was decided to experiment with the different features and combinations of them on the largest dataset CD_B. Table 13 presents the results of the experiment for the SVM and XGBoost models using the proposed algorithm adaption methods label power-set, binary relevance and classifier chains. The evaluation metric hamming loss is abbreviated to *h-loss* in the tables.

Overall, it was observed that using the grammatical distributions as additional features in combination with the bag-of-words representation did not noticeably improve the results. In fact, The different feature combinations achieved mostly identical scores. The

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW with stopwords	**0.12**	**0.85**	**0.12**	**0.86**
BOW without stopwords	0.18	0.76	0.18	0.76

(a) Results for label power-set.

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW with stopwords	**0.12**	**0.86**	**0.12**	**0.86**
BOW without stopwords	0.18	0.76	0.18	0.76

(b) Results for binary relevance.

features	SVM		XGBoost	
	h-loss	macro F_1	h-loss	macro F_1
BOW with stopwords	**0.12**	**0.86**	**0.12**	**0.86**
BOW without stopwords	0.18	0.76	0.18	0.76

(c) Results for classifier chains.

Table 14: Comparison between removing stopwords and keeping stopwords prior to extracting the tf-idf rescaled bag-of-words representation (BOW) for the dataset CD_B. The scores are averaged for the 5-fold cross-validation.

only difference was noticed for SVM using label power-set, in which the feature combinations achieved slightly better scores that differed at most by only 0.01 points. Hence, it was concluded that the grammatical distributions do not benefit the classification of major positions and premises. Thus, the grammatical distributions will not be included in the feature set for the SVM and XGBoost models. In the further course of the work, the results of SVM and XGBoost always refer to the models using solely the tf-idf rescaled bag-of-words representation as an input.

Furthermore, for the SVM and XGBoost models an additional preprocessing step was considered. The elimination of *stopwords* is a commonly performed technique to reduce the feature size of text data. Stopwords are words that are commonly used in a language and may not include relevant information, e.g., articles and connecting words. Especially the bag-of-words representation can become sparse if the generated vocabulary is very large. Hence, by removing stopwords the feature size and computational costs can be reduced. Moreover, removing stopwords can potentially help to improve the performance of the models since they can focus on more meaningful words. (Müller and Guido, 2016)

In an experiment on the largest dataset CD_B, stopwords were filtered out before extracting the tf-idf rescaled bag-of-words representation. For the removal of stopwords, the natural language processing library spaCy[17] was used. The results of the experiment are shown in Table 14. It was observed that all models achieved significantly better results without the elimination of stopwords. The removal of stopwords seemed to eliminate certain words that are relevant for the classification of the different argument component types. Hence, this prepocessing step was discarded.

[17] https://spacy.io/

Regarding BERT and DistilBERT, no further feature selection and preprocessing step was considered. As mentioned in Section 5, for the BERT and DistilBERT models the required input format is obtained by applying the models' tokenizers. Moreover, for BERT and BERT-based models it is not common to conduct a preprocessing step since these pre-trained language models make use of all of the information in a sentence.

7 Results

In this section, the proposed methods SVM, XGBoost, BERT and DistilBERT for the multi-label classification task of identifying major positions and premises among argumentative sentences from German public participation user content are evaluated. First, the models are evaluated on each dataset separately in an intra-dataset evaluation in order to assess how good they are at classifying multi-label argument components in an "optimal" setting. Next, it is investigated whether the findings generalize well in a cross-dataset evaluation.

For reference, the SVM and XGBoost models were run on a regular CPU (Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz). Due to a long execution time and large memory usage, the BERT and DistilBERT models were run on a GPU (Tesla P100-PCIE-16GB) using the cloud-based Jupyter notebook environment Google Colaboratory[18].

7.1 Intra-Dataset Evaluation

In the following, the results of the intra-dataset evaluation are presented and discussed, i.e., the models are trained and tested on each dataset separately. First, the results of SVM and XGBoost using different problem transformation methods (label power-set, binary relevance and classifier chains) are compared. Next, the results of BERT and DistilBERT are presented and discussed. The results are then compared to the single-label classification results by Romberg and Conrad (2021). Lastly, it is examined whether undersampling the majority class helps to produce more constant results for both argument component classes.

7.1.1 Results of SVM and XGBoost

In order to perform multi-label classification with the single-label classifiers SVM and XGBoost, applying a problem transformation method is required. In Section 3.2, label power-set, binary relevance and classifier chains were proposed. The intra-dataset results of SVM and XGBoost using these problem transformation methods are presented in Table 15. In the following course of the work, the classes major position and premise will be abbreviated to *mpos* and *prem* in the tables. The evaluation metric hamming loss will be abbreviated to *h-loss*.

The SVM models achieved macro F_1 scores between 0.77 and 0.85 and hamming losses between 0.11 and 0.20 across the datasets. Comparing the different problem transformation methods, all lead to nearly identical macro F_1 scores and hamming, except for the dataset CQ_B. In this particular dataset, binary relevance performed noticeably better with an macro F_1 score of 0.81 and a hamming loss of 0.14. In contrast, label power-set and classifier chains attained the macro F_1 scores 0.78 and 0.77 and both a hamming loss of 0.16.

In contrast to SVM, the XGBoost models achieved in general lower and less constant scores across the datasets. The macro F_1 scores of XGBoost lie between 0.72 and 0.84

[18] https://colab.research.google.com/

dataset	PTM	h-loss	F_1 scores		
			mpos	prem	macro
CD_B	LP	0.12	**0.79**	0.92	0.85
	BR	0.12	0.77	0.92	0.85
	CC	0.12	**0.79**	0.92	0.85
CD_C	LP	0.17	0.77	0.86	0.82
	BR	0.17	**0.78**	0.86	0.82
	CC	0.17	**0.78**	**0.87**	0.82
CD_M	LP	0.11	0.71	0.93	0.82
	BR	0.11	**0.73**	0.93	**0.83**
	CC	0.11	**0.73**	0.93	**0.83**
CQ_B	LP	0.16	0.91	0.65	0.78
	BR	**0.14**	**0.92**	**0.69**	**0.81**
	CC	0.16	0.91	0.63	0.77
MC_K	LP	0.20	0.84	0.75	0.80
	BR	0.20	**0.85**	**0.76**	0.80
	CC	0.20	**0.85**	0.75	0.80

(a) Results for SVM.

dataset	PTM	h-loss	F_1 scores		
			mpos	prem	macro
CD_B	LP	**0.12**	0.77	**0.92**	0.84
	BR	0.13	0.77	0.91	0.84
	CC	0.13	0.77	**0.92**	0.84
CD_C	LP	0.21	0.72	0.84	0.78
	BR	**0.20**	**0.73**	0.84	**0.79**
	CC	**0.20**	**0.73**	0.84	**0.79**
CD_M	LP	**0.14**	0.61	0.91	0.76
	BR	0.15	0.61	0.91	0.76
	CC	**0.14**	0.61	0.91	0.76
CQ_B	LP	**0.18**	0.90	0.62	**0.76**
	BR	**0.18**	0.90	**0.63**	**0.76**
	CC	0.19	0.90	0.55	0.72
MC_K	LP	**0.24**	**0.81**	**0.71**	**0.76**
	BR	**0.24**	**0.81**	0.70	**0.76**
	CC	0.25	0.80	0.70	0.75

(b) Results for XGBoost.

Table 15: Intra-dataset evaluation. Comparison of SVM and XGBoost using the different problem transformation methods (PTMs) label power-set (LP), binary relevance (BR) and classifier chains (CC) for the multi-label classification of major positions and premises among argumentative sentences. Scores are averaged for the 5-fold cross-validation.

and the hamming losses between 0.12 and 0.25. For this classifier, the different problem transformation methods performed similarly as well. Mostly, the scores differ by only 0.01 points in each dataset. However, as for SVM, the results of the different approaches show a more noticeable difference for CQ_B. In this dataset, binary relevance and label power-set achieved an macro F_1 score that is 0.04 points better than the one achieved by classifier chains. Nevertheless, the hamming losses for CQ_B only differ by at most 0.01 points, i.e., considering partially correct predictions, the three problem transformation methods still performed similarly for CQ_B.

Overall, the performances of the different problem transformation methods show very minimal differences for both classifiers, which is contrary to the expected outcome. As mentioned in Section 3.2, binary relevance assumes that the labels are mutually exclusive and ignores any underlying relations between them, whereas classifier chains takes potential correlation between the classes into consideration. Also, in Table 10 (cf. Section 3.2), Cramér's V was calculated for the classes major position and premise in order to asses any underlying association between the classes. Cramér's V lied between 0.50 and 0.72 for the datasets, indicating a strong relation between the argument component types. Especially for the datasets CD_B and CD_M, high Cramér's V values were calculated. However, no noticeable disadvantage was observed for binary relevance over the other two methods in any dataset. In fact, binary relevance achieved clearly better results compared to classifier chains for one of the datasets. Hence, the existent relation between the two argument component types seems to not restrict the performance of binary relevance in this case. This observation also underlines the statement by Luaces et al. (2012) that binary relevance can still lead to good results when a correct implementation of the method is used and an appropriate base learner is selected.

Another finding contrary to the expected outcome is that label power-set achieved comparable results to the other two problem transformation methods. In Section 3.2, it was mentioned that having less examples for label combinations, which is the case for the multi-label *mpos+premise*, may affect the performance of label power-set negatively since this approach considers each label combination as a separate class. However, in our classification task, label power-set achieved comparable results to the other two problem transformation methods. This may be due to the reason that only three classes are considered (*mpos*, *premise* and *mpos+premise*), hence the number of classes is still low.

On closer inspection of the results across the different datasets, all models achieved the highest macro F_1 scores and lowest hamming losses for the dataset CD_B. Also, only for CD_B, XGBoost produced comparable results to SVM. Since CD_B is significantly larger than the remaining datasets, the models are trained on much more training examples compared to the remaining datasets, thus it was expected that the models achieve better results for this particular dataset. On average, the lowest macro F_1 scores and highest hamming losses were attained for CQ_B and MC_K by both classifiers. The lower results for MC_K may be due to the fact that the dataset originates from a public participation process that concerns the more general topic mobility in contrast to the remaining datasets that focus on cycling. Hence, the user comments of MC_K may show more differences in their argument structure. Since the dataset CQ_B originates from a postal survey, the contained arguments also may have a different structure compared to the arguments in the cycling dialogue datasets.

Examining the results for the different argument component classes, SVM and XGBoost attained better F_1 scores for the majority class of each dataset. This was expected since the provided datasets are highly imbalanced. In CD_B, CD_C and CD_M, premise is the majority class, for which better F_1 scores were achieved than for the class major position, whereas in CQ_B and MC_K, major position forms the majority class and was recognized better.
The largest differences between the results of the classes were observed for the datasets CD_M and CQ_B. For CD_M, the SVM models showed on average a difference of 0.21 points between the F_1 scores of both classes and the XGBoost models on average a difference of 0.30. This was expected, as the datasets CD_M and CQ_B are more imbalanced in contrast to the remaining datasets.
It was also expected that the results for the different classes differ the least for MC_K since this particular dataset is less imbalanced compared to the remaining datasets. However, this was not the case. Instead, the least difference between the results of the classes was observed for CD_C, the second least imbalanced dataset after MC_K. Again, this may be due to the fact that the user comments in MC_K concern a more general topic, which may affect the ability of the classifiers to recognize certain argument component types.

7.1.2 Results of BERT and DistilBERT

Next, the results of the intra-dataset evaluation for the applied BERT models and DistilBERT are presented. The results are shown in Table 16. As already mentioned, for BERT and BERT-based models the use of a problem transformation method is not required. In order to do multi-label classification, simply the output layer of the models have to be

dataset	BERT model	h-loss	F_1 scores		
			mpos	prem	macro
CD_B	GermanBERT	0.07	0.89	0.95	0.92
	GBERT	**0.06**	**0.91**	0.95	**0.93**
	DistilBERT	0.07	0.90	0.95	0.92
CD_C	GermanBERT	0.09	**0.90**	**0.93**	**0.91**
	GBERT	**0.08**	**0.90**	**0.93**	**0.91**
	DistilBERT	0.10	0.88	0.92	0.90
CD_M	GermanBERT	0.06	0.87	0.95	0.91
	GBERT	**0.05**	**0.89**	**0.96**	**0.93**
	DistilBERT	0.06	0.86	**0.96**	0.91
CQ_B	GermanBERT	0.08	0.96	0.84	0.90
	GBERT	**0.06**	**0.97**	**0.88**	**0.92**
	DistilBERT	0.07	**0.97**	0.85	0.91
MC_K	GermanBERT	0.13	0.90	0.84	0.87
	GBERT	**0.11**	**0.92**	**0.86**	**0.90**
	DistilBERT	0.12	**0.92**	**0.86**	0.89

Table 16: Intra-dataset evaluation. Comparison of GermanBERT, GBERT and German DistilBERT for the multi-label classification of major positions and premises among argumentative sentences. Scores are averaged for the applied 5-fold cross-validation.

adjusted (cf. Section 4.2.4).

Overall, GermanBERT, GBERT and DistilBERT achieved remarkably good results that lie very close to another and outperformed SVM and XGBoost by far. In most cases, GBERT achieved slightly better scores than GermanBERT and DistilBERT. For GBERT, the average macro F_1 score and hamming loss for the datasets lies at 0.92 and 0.07. Meanwhile, the average macro F_1 scores of DistilBERT and GermanBERT lie both at 0.91 and their hamming losses at 0.08 and 0.09. It was expected that GBERT performs better than GermanBERT since it is the successor model of GermanBERT. However, it was not expected that the significantly smaller DistilBERT model performs equally well as GermanBERT. This shows that a bigger model size does not necessarily lead to a better performance and that DistilBERT can compete with the BERT models.

Comparing the results across the datasets shows that all models achieved the best results for CD_B and the worst results for MC_K. As earlier mentioned in the intra-dataset evaluation of SVM and XGBoost, CD_B is significantly larger than the remaining datasets, hence the models are trained on much more training examples. Thus, it was expected that the models achieve better results for this particular dataset. As for SVM and XGBoost, the lower performance for MC_K may indicate that the user comments of this particular dataset show more differences in their argument structure since the dataset originates from a public participation process that concerns a more general topic.
Moreover, compared to the other two models, GBERT produced the most constant results for the different datasets. GBERT's macro F_1 scores differ at most by 0.03 points and the hamming losses by 0.06. GermanBERT shows the largest difference with macro F_1 scores that differ at most by 0.05 points and hamming losses that differ at most by 0.07. However, the results are still remarkably constant.

model	CD_B	CD_C	CD_M	CQ_B	MC_K
GermanBERT	1h 3min	11min	13min	10min	11min
GBERT	1h 4min	11min	13min	10min	11min
DistilBERT	**35min**	**6min**	**7min**	**5min**	**6min**

Table 17: Comparison of the run times for the intra-dataset evaluation of the applied BERT and DistilBERT models. The run times include the 5-fold cross-validation and grid search.

As already mentioned in the evaluation of the SVM and XGBoost models, it is expected that the models attain better F_1 scores for the majority class of each dataset since the datasets are highly imbalanced and for those classes clearly more training examples are provided. Thus, BERT and DistilBERT recognize the class premise better in the datasets CD_B, CD_C and CD_M, whereas in CQ_B and MC_K major positions are recognized better.
Also, the largest difference between the results for both classes was again observed for the datasets CD_M and CQ_B. For CD_M, the three models showed on average a difference of 0.08 points for the F_1 scores of both classes and for CQ_B a difference of 0.11, which may be due to the reason that CD_M and CQ_B are more imbalanced than the other datasets. On the other hand, it was also expected that the results for the different classes differ the least for MC_K since it is less imbalanced than the remaining datasets. However, as for SVM and XGBoost, this was not the case for BERT and DistilBERT, either. Instead, the least difference was shown for CD_C, the second least imbalanced dataset after MC_K. This outcome may be again due to the fact that MC_K concerns a more general topic, which may affect the ability of the models to recognize certain argument component types negatively.

Moreover, in world applications efficiency also plays an important role besides performance since the objective is to provide tools that are widely applicable, e.g., on different types of devices. The results have shown that the BERT models and DistilBERT achieved comparable good results. Therefore, the run times of the different models were investigated next in order to determine which model may be more suitable in practice. In Table 17, the run times of the models on the GPU are listed, which include the grid search and 5-fold cross-validation. As expected, the models for the largest dataset CD_B required a significantly longer run time compared to the models for the other, smaller datasets, for which similar run times were shown. However, in comparison with the bigger sized models, running DistilBERT took only about half as much time for all datasets. Since DistilBERT achieved competitive results to the BERT models while having a model size that is about 40% smaller and being about twice as fast, it may be more advantageous to use DistilBERT in world applications, especially if limited computational resources are given.

7.1.3 Comparison to Romberg and Conrad (2021)

Overall, the multi-label classification and single-label classification of major positions and premises lead to comparable results. The results achieved by Romberg and Conrad (2021)

for the single-label classification task were presented in Table 4 (cf. Section 2.3). Romberg and Conrad applied GermanBERT and SVM as well. Their GermanBERT model achieved an average macro F_1 score of 0.91 for the datasets, which is equal to the average macro F_1 score of our GermanBERT and DistilBERT models. However, our GBERT model attained a slightly better average macro F_1 score of 0.92 and more constant results with macro F_1 scores that differed at most by 0.03 points. In contrast, the macro F_1 scores of the applied GermanBERT model by Romberg and Conrad showed a difference of at most 0.06 points. For reference, GermanBERT's macro F_1 scores differed at most by 0.05 in this work for the multi-label classification task. Regarding SVM, Romberg and Conrad achieved an average macro F_1 score of 0.82, which is also the case for our SVM model using the problem transformation method binary relevance.
Another similarity seen between the results of both works is that the models behave similarly for the different datasets, achieving the best results for the largest dataset CD_B and the lowest results for the datasets CQ_B and MC_K. Also, the models of both works tend to recognize the majority class of each dataset better.

7.1.4 Undersampling

For world applications it may be beneficial to provide models that perform approximately equally well for the different argument component classes since the number of major positions and premises can vary in each dataset. Given the provided imbalanced datasets, all models, especially the SVM and XGBoost models, performed much better for the majority class of the datasets. Undersampling the majority class, which was also carried out by Romberg and Conrad (2021) for their ECGA and fastText models, may help to produce more constant results.

The majority class of each dataset was undersampled to be equal to the number of examples of the minority class by picking random samples. For instance, in CD_B, the majority class premise comprised 6,438 examples and was undersampled to 2,589 examples, which is the number of examples for the minority class major position. Table 18 presents the results of the best BERT model (GBERT), DistilBERT, SVM and XGBoost for the undersampling versions of the datasets (marked with an asterisk). For SVM and XGBoost, the problem transformation method binary relevance was used.

Applying undersampling produced remarkably more stable results for the different argument component classes. For GBERT, the F_1 scores between the two classes differ at most by 0.03 points, for SVM at most by 0.02 points and for XGBoost and DistilBERT at most by 0.04 points. Moreover, the overall performances of the models are very similar to the performances of the models without undersampling (cf. Table 15 and 16). GBERT's macro F_1 scores differ at most by 0.02 points and the hamming losses by 0.04 for both versions. For DistilBERT, the scores differ even less. For DistilBERT, SVM and XGBoost, some macro F_1 scores even improved for some of the datasets.

A noticeable difference was seen for the dataset CQ_B, for which SVM achieved an improved macro F_1 score by 0.03 points and XGBoost an improved macro F_1 score by 0.04 points using undersampling. However, the hamming losses of both classifiers worsened by 0.06 points. Overall, a slight deterioration of the hamming losses was also observed for the remaining datasets and classifiers, which means that more partially incorrect pre-

dataset	classifier	h-loss	F_1 scores		
			mpos	prem	macro
CD_B*	GBERT	**0.08**	**0.93**	**0.92**	0.92
	DistilBERT	**0.08**	**0.93**	**0.92**	**0.93**
	SVM	0.15	0.86	0.86	0.86
	XGBoost	0.15	0.86	0.86	0.86
CD_C*	GBERT	0.10	0.91	**0.91**	**0.91**
	DistilBERT	**0.09**	**0.92**	0.90	**0.91**
	SVM	0.19	0.81	0.83	0.82
	XGBoost	0.23	0.79	0.78	0.79
CD_M*	GBERT	**0.09**	0.93	**0.91**	**0.92**
	DistilBERT	**0.09**	**0.94**	0.90	0.91
	SVM	0.19	0.82	0.81	0.82
	XGBoost	0.28	0.74	0.74	0.74
CQ_B*	GBERT	**0.09**	**0.94**	**0.91**	**0.93**
	DistilBERT	**0.09**	**0.94**	**0.91**	0.92
	SVM	0.20	0.84	0.84	0.84
	XGBoost	0.24	0.82	0.78	0.80
MC_K*	GBERT	0.13	**0.90**	0.88	0.88
	DistilBERT	**0.12**	**0.90**	**0.89**	**0.89**
	SVM	0.21	0.82	0.81	0.81
	XGBoost	0.25	0.78	0.77	0.78

Table 18: Intra-dataset evaluation after undersampling the majority class of each dataset. Scores are averaged for the 5-fold cross-validation.

dictions occurred for the undersampling versions. An underlying reason may be that the number of training examples were significantly reduced after undersampling the majority classes. Especially, the most imbalanced dataset CQ_B was extremely reduced in size from 1,333 samples to 500 samples. If more training examples were available for the minority classes, the models' performances may have improved.

7.2 Cross-Dataset Evaluation

So far, the models have been trained and tested on each dataset separately. However, for world applications it is advantageous to provide a tool that also works well on unseen data from different sources, e.g., from different public participation processes. Therefore, the performances of GBERT, DistilBERT, SVM and XGBoost are investigated in terms of generalizability. For that, the models are trained on one or more datasets and evaluated on the remaining, unseen datasets.

First, the results of the cross-dataset evaluation are presented and discussed. Then a comparison is drawn to the results of the intra-dataset evaluation, and lastly, the results are compared to the cross-dataset evaluation of Romberg and Conrad (2021).

7.2.1 Results of the Cross-Dataset Evaluation

Table 19 shows the results of the cross-dataset evaluation for the best BERT model (GBERT), DistilBERT, SVM and XGBoost. The classifiers SVM and XGBoost were applied in combination with binary relevance. First, all models were trained on CD_B, which was selected intentionally since it is the largest dataset among all and thus the best scores were achieved using this dataset. Moreover, Romberg and Conrad (2021) used CD_B for training as well in their cross-dataset evaluation. Next, in addition to CD_B one of the remaining datasets were used for training as well to examine whether the results improve adding additional datasets. When training on CD_B in combination for other datasets, it was sampled down to 2,000 randomly selected sentences in order to approximately match the size of the remaining datasets. Otherwise, the model would learn more from the largest dataset CD_B and the remaining, smaller datasets would have less impact.

The results have shown that the BERT and BERT-based models generalized very well. When using only CD_B for training, GBERT attained macro F_1 scores between 0.88 and 0.92 and hamming losses between 0.06 and 0.12. Similar scores were achieved by DistilBERT with macro F_1 scores that lie between 0.87 and 0.92 and hamming losses between 0.06 and 0.14. Moreover, the best scores were achieved for the dataset CD_M, which leads to the assumption that CD_M resembles the dataset CD_B the most. However, adding additional datasets for training did not improve the results on any dataset. In fact, in some cases the scores worsened, especially when adding CQ_B for training. This indicates that CQ_B differs most from the remaining datasets, which may be due to the fact that it originates from a postal survey in contrast to the other datasets.

Furthermore, SVM and XGBoost showed strong weaknesses in generalizing on the datasets CQ_B and MC_K and produced highly unstable results. Using solely CD_B for training, SVM reached macro F_1 scores between 0.69 and 0.85 and hamming losses between 0.10 and 0.31. XGBoost showed an even higher variance in the performance across the different datasets with macro F_1 scores between 0.61 and 0.84 and hamming losses between 0.10 and 0.41. As for GBERT and DistilBERT, the best scores were achieved for the dataset CD_M. However, in contrast GBERT and DistilBERT, adding the dataset MC_K for training improved the results for CQ_B. Especially XGBoost performed noticeably better with an improved macro F_1 score by 0.12 points and an improved hamming loss by 0.15 points for CQ_B when using additionally MC_K for training.

7.2.2 Comparison to Intra-Dataset Evaluation

Comparing the results of the cross-dataset evaluation using CD_B for training with the results of the intra-dataset evaluation (cf. Table 15 and 16), it is noticed that GBERT and DistilBERT produced results that lie very close to the dataset-internal results. In case of GBERT, the macro F_1 scores and hamming losses in both evaluations differ at most by 0.03 points, whereas for DistilBERT both scores differ at most by 0.04 points. This indicates that BERT and DistilBERT are capable of producing highly generalizable results.

On the other hand, the SVM and XGBoost models performed noticeably worse for the datasets CQ_B and MC_K in the cross-dataset evaluation, while the results for CD_C of

both evaluations lie very close to another. However, in the cross-dataset evaluation, the results for CD_M using CD_B for training improved surprisingly. Again, this indicates that the user comments in both datasets are structurally very similar. The improvement may be due to the larger size of CD_B since more training examples are given.

7.2.3 Comparison to Romberg and Conrad (2021)

Romberg and Conrad (2021) evaluated BERT and SVM for the single-label classification of major positions and premises in a cross-dataset evaluation as well. The results of their cross-dataset evaluation were presented in Table 1 (cf. Section 2.3). In contrast to this work, they only used CD_B for training and did not consider further dataset combinations.

Comparing our BERT models, both generalized very well and performed similarly across the datasets. Regarding the macro F_1 scores, the only differences were seen for the datasets CQ_B and MC_K, for which their GermanBERT model performed by 0.01 points better in contrast to our GBERT model.

Comparing the results of the SVM models, Romberg and Conrad attained in general higher macro F_1 scores and more stable results over the different datasets. Particularly for the datasets CQ_B and MC_K, their SVM model achieved noticeably better macro F_1 scores that are by 0.07 and 0.06 points better. This may indicate that using the problem transformation method binary relevance weakens the generalizability of SVM.

train datasets	CD_B		CD_C		CD_M		CQ_B		MC_K	
	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1
CD_B	-	-	**0.08**	**0.92**	**0.06**	**0.92**	**0.09**	**0.89**	**0.12**	**0.88**
CD_B + CD_C	-	-	-	-	**0.06**	**0.92**	0.12	0.86	**0.12**	**0.88**
CD_B + CD_M	-	-	**0.08**	**0.92**	-	-	0.11	0.87	0.15	0.86
CD_B + CQ_B	-	-	0.11	0.88	0.08	0.89	-	-	0.17	0.84
CD_B + MC_K	-	-	**0.08**	**0.92**	0.07	0.91	**0.09**	**0.89**	-	-

(a) GBERT

train datasets	CD_B		CD_C		CD_M		CQ_B		MC_K	
	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1
CD_B	-	-	**0.09**	**0.91**	0.06	**0.92**	**0.11**	**0.87**	**0.14**	**0.87**
CD_B + CD_C	-	-	-	-	0.06	**0.92**	0.15	0.84	**0.14**	**0.87**
CD_B + CD_M	-	-	0.10	0.90	-	-	0.12	0.86	0.15	0.86
CD_B + CQ_B	-	-	0.11	0.89	0.06	0.91	-	-	**0.14**	0.86
CD_B + MC_K	-	-	**0.09**	**0.91**	0.06	0.91	**0.11**	**0.87**	-	-

(b) DistilBERT

train datasets	CD_B		CD_C		CD_M		CQ_B		MC_K	
	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1
CD_B	-	-	**0.16**	**0.82**	**0.10**	**0.85**	0.31	0.69	**0.27**	0.73
CD_B + CD_C	-	-	-	-	0.12	0.83	0.34	0.67	0.29	0.72
CD_B + CD_M	-	-	0.18	0.80	-	-	0.41	0.60	0.34	0.67
CD_B + CQ_B	-	-	0.17	**0.82**	0.15	0.81	-	-	0.23	**0.78**
CD_B + MC_K	-	-	0.17	**0.82**	0.13	0.82	**0.28**	**0.71**	-	-

(c) SVM

train datasets	CD_B		CD_C		CD_M		CQ_B		MC_K	
	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1	h-loss	macro F_1
CD_B	-	-	**0.19**	0.79	**0.10**	**0.84**	0.41	0.61	0.35	0.65
CD_B + CD_C	-	-	-	-	0.14	0.81	0.33	0.68	0.30	0.71
CD_B + CD_M	-	-	0.23	0.74	-	-	0.53	0.49	0.40	0.60
CD_B + CQ_B	-	-	0.20	**0.80**	0.18	0.76	-	-	**0.25**	**0.76**
CD_B + MC_K	-	-	0.20	**0.80**	0.16	0.79	**0.26**	**0.73**	-	-

(d) XGBoost

Table 19: Cross-dataset evaluation. Comparison of GBERT, DistilBERT, SVM and XGBoost for the multi-label classification of major positions and premises among argumentative sentences.

8 Conclusion and Outlook

The objective of this thesis was to follow up on the work of Romberg and Conrad (2021) and identify major positions and premises among argumentative sentences originating from German public participation processes concerning urban planning and sustainable mobility. In contrast to Romberg and Conrad, additionally the case of sentences that contain both argument component types was considered, i.e., the classification task was viewed as a multi-label classification task instead of a single-label classification task.
For the classification of argument component types, four different classification algorithms were selected: BERT, DistilBERT, SVM and XGBoost. In comparison, Romberg and Conrad applied GermanBERT, SVM and two other machine learning algorithms, fastText and ECGA, for the single-label classification of major positions and premises. The latter two methods were not considered in this work since they were clearly outperfomed by BERT and SVM.
Since XGBoost and SVM do not support multi-label classification directly, different problem transformation methods were considered, which transform the problem into one or more single-label classification tasks. Among the considered problem transformation methods, label power-set, binary relevance and classifier chains were chosen.
First, all models were evaluated in an dataset-internal evaluation in order to examine how well they perform in an "optimal" setting. Then, it was investigated whether the models generalize well for unseen datasets in a cross-dataset evaluation, using one or more datasets for training and the remaining datasets for testing. In both evaluations, BERT and DistilBERT performed similarly well and clearly outperformed SVM and XGBoost by producing better and more stable results.
In the dataset-internal evaluation, BERT and DistilBERT achieved on average macro F_1 scores of 0.92 and 0.91, and hamming losses of 0.08 and 0.09. Despite being 40% smaller in model size and about twice as fast, DistilBERT could compete with BERT. Hence, in terms of efficiency, for world applications DistilBERT may be more advantageous, especially if limited computational resources are available. For the classifiers SVM and XGBoost, it was shown that the problem transformation methods perform very similarly, however, binary relevance scored slightly better. In comparison, the applied BERT model in Romberg and Conrad (2021) for the single-label classification of major positions and premises achieved an average F_1 macro score of 0.91 as well.
In the cross-dataset evaluation, BERT and DistilBERT produced highly generalizable results across the datasets. BERT achieved the best results, attaining macro F_1 scores between 0.88 and 0.92 and hamming losses between 0.06 and 0.12 using the largest dataset for training. Similar scores were reached with DistilBERT. In contrast, SVM and XGBoost showed strong weaknesses in generalizing for some of the datasets.
To conclude, among the different methods, BERT and DistilBERT performed best for the multi-label classification of major positions and premises among argumentative sentences. Furthermore, the achieved results were comparable to the results of Romberg and Conrad (2021) for the single label classification task.

In order to further improve the evaluation of public participation processes, future work may try to identify whether sequential sentences in contributions belong to the same major position or premise since statements are often made using several sentences. The

investigation of coherent sentences may help to better understand the ideas and thoughts of the participants. As this work highlighted, sentences can contain multiple argument components, thus solving the task on token-level instead of sentence-level for each contribution may give a more precise insight into the argument structure.
Another aspect that may be of interest for the automatic evaluation of public participation data is to identify relations between the argument component types, e.g., which premises support which major position. Identifying relations between the arguments may help to further understand the reasoning behind the statements.
Furthermore, future work may investigate whether the trained models of this work also work well on public participation data originating from different domains other than urban planning and mobility. It may be more advantageous to provide a model that works well on a wide branch of domains, hence, training additionally on public participation data from different sources and processes may help to produce a more generalizable model.
Lastly, future works that also focus on the multi-label classification of major positions and premises may consider to apply algorithm adaption methods and compare the results of the adapted classifiers to the original single-label classifiers using different problem transformation methods.

References

Haldun Akoglu (2018). "User's Guide to Correlation Coefficients". In: *Turkish journal of emergency medicine* 18.3, pp. 91–93.

M. Arana-Catania, F. A. Van Lier, Rob Procter, Nataliya Tkachenko, Yulan He, Arkaitz Zubiaga, and Maria Liakata (2021). "Citizen Participation and Machine Learning for a Better Democracy". In: *Digital Government: Research and Practice* 2, pp. 1–22.

Sherry R. Arnstein (1969). "A Ladder Of Citizen Participation". In: *Journal of the American Institute of Planners* 35.4, pp. 216–224.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith (2002). "The TIGER Treebank". In: *Proceedings of the workshop on treebanks and linguistic theories*. Vol. 168, pp. 24–41.

Leo Breiman (2001). "Random Forests". In: *Machine learning* 45.1, pp. 5–32.

Tuhin Chakrabarty, Christopher Hidey, Smaranda Muresan, Kathy McKeown, and Alyssa Hwang (Nov. 2019). "AMPERSAND: Argument Mining for PERSuAsive oNline Discussions". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, pp. 2933–2943.

Branden Chan, Stefan Schweter, and Timo Möller (Dec. 2020). "German's Next Language Model". In: *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics.

Kaiping Chen and Tanja Aitamurto (2019). "Barriers for Crowd Impact in Crowdsourced Policymaking: Civic Data Overload and Filter Hierarchy". In: *International Public Management Journal* 22.1, pp. 99–126.

Tianqi Chen and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. Association for Computing Machinery, pp. 785–794.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 1724–1734.

T. Cleff (2008). *Deskriptive Statistik und moderne Datenanalyse: eine computergestützte Einführung mit Excel, SPSS und STATA*. Gabler Lehrbuch. Gabler.

Oana Cocarascu, Elena Cabrio, Serena Villata, and Francesca Toni (2020). "A Dataset Independent Set of Baselines for Relation Prediction in Argument Mining". In: *COMMA 2020-8th International Conference on Computational Models of Argument* 326, pp. 45–52.

Corinna Cortes and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297.

Harald Cramér (1946). *Mathematical Methods of Statistics*. Princeton University Press.

James L Creighton (2005). *The public participation handbook: Making better decisions through citizen involvement*. Jossey-Bass.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (June 2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, pp. 4171–4186.

The Editors of Encyclopaedia Britannica (1998). *Municipality*. https://www.britannica.com/topic/municipality. Accessed: 2022-05-23.

Joseph Fleiss (Nov. 1971). "Measuring Nominal Scale Agreement Among Many Raters". In: *Psychological Bulletin* 76, pp. 378–382.

Jerome H Friedman (2002). "Stochastic Gradient Boosting". In: *Computational statistics & data analysis* 38.4, pp. 367–378.

Athanasios Giannakopoulos, Maxime Coriou, Andreea Hossmann, Michael Baeriswyl, and Claudiu Musat (June 2019). "Resilient Combination of Complementary CNN and RNN Features for Text Classification through Attention and Ensembling". In: *2019 6th Swiss Conference on Data Science (SDS)*, pp. 57–62.

Eva Gibaja and Sebastian Ventura (Apr. 2015). "A Tutorial on Multi-Label Learning". In: *ACM Computing Surveys* 47.

Kathleen Halvorsen (Sept. 2003). "Assessing the Effects of Public Participation". In: *Public Administration Review* 63, pp. 535–543.

F. Herrera, F. Charte, A.J. Rivera, and M.J. Jesus (2016). *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer International Publishing.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network". In: *NIPS Deep Learning and Representation Learning Workshop*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov (Apr. 2017). "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, pp. 427–431.

B.H. Kang and Q. Bai (2016). *AI 2016: Advances in Artificial Intelligence: 29th Australasian Joint Conference, Hobart, TAS, Australia, December 5-8, 2016, Proceedings*. Lecture Notes in Computer Science. Springer International Publishing.

Namhee Kwon, Stuart Shulman, and Eduard Hovy (Jan. 2006). "Multidimensional Text Analysis for eRulemaking". In: vol. 151, pp. 157–166.

David Le Blanc (2020). *E-participation: A Quick Overview of Recent Qualitative Trends*. UN Department of Economic and Social Affairs (DESA) Working Papers. UN.

Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner (Dec. 1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86, pp. 2278–2324.

Athanasios Lentzas, Eleana Dalagdi, and Dimitris Vrakas (2022). "Multilabel Classification Methods for Human Activity Recognition: A Comparison of Algorithms". In: *Sensors* 22.6, p. 2353.

Minglan Li, Yang Gao, Hui Wen, Yang Du, Haijing Liu, and Hao Wang (2017). "Joint RNN Model for Argument Component Boundary Detection". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, pp. 57–62.

Matthias Liebeck, Katharina Esau, and Stefan Conrad (Aug. 2016). "What to Do with an Airport? Mining Arguments in the German Online Participation Project Tempelhofer Feld". In: *Proceedings of the Third Workshop on Argument Minings*. Association for Computational Linguistics, pp. 144–153.

Oscar Luaces, Jorge Díez, José Barranquero, Juan José del Coz, and Antonio Bahamonde (2012). "Binary Relevance Efficacy for Multilabel Classification". In: *Progress in Artificial Intelligence* 1.4, pp. 303–313.

Andreas C Müller and Sarah Guido (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. "O'Reilly Media, Inc.".

Joonsuk Park and Claire Cardie (June 2014). "Identifying Appropriate Support for Propositions in Online User Comments". In: *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, pp. 29–38.

Josh Patterson and Adam Gibson (2017). *Deep Learning: A Practitioner's Approach*. "O'Reilly Media, Inc.".

Karl Pearson (July 1900). "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302, pp. 157–175.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank (2009). "Classifier Chains for Multi-label Classification". In: *ECML '09: 20th European conference on machine learning*. Springer, pp. 254–269.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank (2011). "Classifier Chains for Multi-label Classification". In: *Machine learning* 85.3, pp. 333–359.

Jesse Read, Bernhard Pfahringer, and Geoffrey Holmes (Dec. 2008). "Multi-label Classification Using Ensembles of Pruned Sets". In: *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 995–1000.

Ines Rehbein and Sören Schalowski (2013). "STTS goes Kiez–Experiments on annotating and tagging urban youth language". In: *Journal for Language Technology and Computational Linguistics (JLCL)* 28.1, pp. 199–227.

Nancy Roberts (Dec. 2004). "Public Deliberation in an Age of Direct Citizen Participation". In: *American Review of Public Administration - AMER REV PUBLIC ADM* 34, pp. 315–353.

Julia Romberg and Stefan Conrad (2021). "Citizen Involvement in Urban Planning - How Can Municipalities Be Supported in Evaluating Public Participation Processes for Mobility Transitions?" In: *Proceedings of the 8th Workshop on Argument Mining*, pp. 89–99.

Julia Romberg, Laura Mark, and Tobias Escher (2022). *CIMT PartEval Corpus - Argument Components (Subcorpus)*. https://github.com/juliaromberg/cimt-argument-mining-dataset.

Gene Rowe and Lynn J Frewer (2004). "Evaluating Public Participation Exercises: A Research Agenda". In: *Science, technology, & human values* 29.4, pp. 512–556.

Gerard Salton and Christopher Buckley (1988). "Term-weighting Approaches in Automatic Text Retrieval". In: *Information processing & management* 24.5, pp. 513–523.

Arjan Sammani, Ayoub Bagheri, Peter GM van der Heijden, Anneline SJM Te Riele, Annette F Baas, CAJ Oosters, Daniel Oberski, and Folkert W Asselbergs (2021). "Automatic Multilabel Detection of ICD10 Codes in Dutch Cardiology Discharge Letters using Neural Networks". In: *NPJ digital medicine* 4.1, pp. 1–10.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf (Oct. 2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108*.

Robin Schaefer and Manfred Stede (2020). "Annotation and Detection of Arguments in Tweets". In: *Proceedings of the 7th Workshop on Argument Mining*, pp. 53–58.

Michael Scharkow (2011). "Zur Verknüpfung manueller und automatischer Inhaltsanalyse durch maschinelles Lernen". In: *M&K Medien & Kommunikationswissenschaft* 59.4, pp. 545–562.

Anne Schiller, Simone Teufel, Christine Thielen, and Christine Stöckert (1999). "Guidelines für das Tagging deutscher Textcorpora mit STTS". In: *University of Stuttgart and Seminar für Sprachwissenschaft*.

Santosh Kumar Shrivastav and P Janaki Ramudu (2020). "Bankruptcy Prediction and Stress Quantification Using Support Vector Machine: Evidence from Indian Banks". In: *Risks* 8.2, p. 52.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary (2019). "Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures". In: *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*.

Peter Teufl, Udo Payer, and Peter Parycek (Sept. 2009). "Automated Analysis of e-Participation Data by Utilizing Associative Networks, Spreading Activat ion and Unsupervised Learning". In: *International Conference on Electronic Participation*. Vol. 5694, pp. 139–150.

Jörg Tiedemann (2012). "Parallel Data, Tools and Interfaces in OPUS". In: *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Citeseer, pp. 2214–2218.

Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, Ioannis P Vlahavas, et al. (2008). "Multi-Label Classification of Music into Emotions." In: *ISMIR*. Vol. 8, pp. 325–330.

Grigorios Tsoumakas and Ioannis Katakis (July 2007). "Multi-label classification: An overview". In: *International Journal of Data Warehousing and Mining (IJDWM)* 3, pp. 1–13.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need". In: *Advances in neural information processing systems* 30.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and Jeffrey Dean (Sept. 2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *ArXiv*.

Hamada M Zahera, Ibrahim A Elgendy, Rricha Jalota, and Mohamed Ahmed Sherif (2019). "Fine-tuned BERT Model for Multi-Label Tweets Classification." In: *TREC*, pp. 1–7.

J. Zeng, W. Jing, X. Song, and Z. Lu (2020). *Data Science: 6th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2020, Taiyuan, China, September 18-21, 2020, Proceedings*. Communications in Computer and Information Science. Springer.

Alice Zheng and Amanda Casari (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. "O'Reilly Media, Inc.".

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler (2015). "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *Proceedings of the IEEE international conference on computer vision*, pp. 19–27.

List of Figures

List of Tables