Leveraging ClimateBERT for Climate Legislation Analysis: Evaluating Sampling Strategies on the Polianna Dataset

Sven Lutz

January 22, 2025

Abstract

ABC

List of Abbreviations

ADASYN Adaptive Synthetic Sampling

AI Artificial Intelligence

BERT Bidirectional Encoder Representations from Transformers

IEKP Integriertes Energie- und Klimaprogramm

ML Machine Learning

NLP Natural Language Processing

SMOTE Synthetic Minority Oversampling Technique

TPE Tree-structured Parzen Estimators

EU European Union

IAA Inter-Annotator Agreement

HPO Hyperparameter Optimisation

Contents

Chapter 1

Introduction

Climate policy and energy policy are closely linked in legislation, and the terms are sometimes used interchangeably [1]. The origins of international climate policy go back further than the milestones known to many [2, 3], such as the 2015 Paris Agreement [4] or the Kyoto Conference [5]. An important starting point was the adoption of the United Nations Framework Convention on Climate Change [6], agreed at the 1992 Earth Summit in Rio de Janeiro. This Convention forms the basis for numerous other international and national measures in the field of climate protection law. At the national level, for example, the Federal Climate Protection Act of 2019 [7] plays a key role [8].

Another important example is the Integrated Energy and Climate Programme Integriertes Energie- und Klimaprogramm (IEKP) adopted by the Federal Cabinet in August 2007 [9]. This programme, consisting of 29 specific measures, should help to reduce greenhouse gas emissions in Germany by 40% by 2020 compared to 1990 levels [8]. However, such policies and legislation require regular evaluation to check their effectiveness and efficiency [10]. A monitoring process has been in place since 2011, documenting climate policy developments in annual and triennial progress reports [11, 12]. The process is supported by an independent panel of eminent energy experts whose expertise ensures that the assessment is both technically sound and relevant to practice. Their opinions play a key role in categorising the monitoring results and formulating recommendations for future climate policy.

In addition to these legal and policy developments, there is the question of how technological innovation, particularly in the field of Artificial Intelligence (AI) [13], can support the evaluation and further development of climate laws. The focus here is on human-AIcollaboration, i.e. the cooperation between human experts and AIsystems in order to analyse complex legal texts more efficiently and to evaluate their effects more precisely [14]. Modern methods of Natural Language Processing (NLP), in particular those based on Transformer technology [15], offer promising potential here.

An important step in this direction is the development of the Polianna dataset, which was created specifically for the analysis of legal texts in the field of climate change. The aim of this work is to investigate how this dataset can be optimally used to generate well-founded predictions and analyses using NLP methods. The focus will be on the application of a specialised transformer model, ClimateBERT, which has particular strengths [16] in the processing of climate-related texts due to its domain adaptation.

The central research questions of this study can be formulated as follows:

1. **Precision of NLP methods:** Can modern NLP approaches make accurate predictions about the effects of legislative measures?

2. **Effectiveness of sampling strategies:** What sampling strategies are appropriate to overcome the challenges of a limited and unbalanced data set while improving model performance?

3. **Comparison of different modelling approaches:** Does direct prediction of law features lead to better results compared to layer level analysis?

In order to answer the research questions of this thesis, a systematic methodology was developed that included both the adaptation of the dataset and the evaluation of different sampling strategies and modelling approaches. First, the Polianna dataset had to be adapted and cleaned to provide a consistent and reliable basis for the subsequent analyses and to meet the requirements for modelling with NLP techniques.

A key aspect of the methodology was to deal with an unbalanced dataset in which some classes are significantly more represented than others. Various sampling strategies were used to address this, including oversampling, undersampling and hybrid methods such as Synthetic Minority Oversampling Technique (SMOTE) [17], Adaptive Synthetic Sampling (ADASYN) [18] and SMOTEEN (a combination of SMOTE and Edited Nearest Neighbours) [?].

The experiments were based on a classical train-test-validate split, with 80% of the data used for training. To obtain the best results, the different sampling strategies were subjected to Hyperparameter Optimisation (HPO) in combination with ClimateBERT. This was done using Optuna, a framework for efficient and automated HPO [19]. Optuna uses a combination of Bayesian optimisation and Tree-structured Parzen Estimators (TPE) to identify promising parameter configurations to maximise model performance [20].

This systematic approach allowed us to compare the performance of all tested combinations of datasets, sampling strategies and model parameters. The aim was to identify the optimal sampling strategy and the best hyperparameters for ClimateBERT to enable accurate prediction of law characteristics.

In the following chapters, the Polianna dataset and the data pre-processing steps are described in detail. ClimateBERT, the central model of this study, is then presented in detail, followed by a concrete description of the sampling strategies used and the necessity of their application. Furthermore, Optuna, the framework for HPO, is explained in detail. In the Approach and Results chapter, the individual approaches and the associated results are discussed in detail. The subsequent Discussion chapter is dedicated to the interpretation and categorisation of these results, before the concluding Outlook and Limitations chapter provides further insights and possible perspectives for future work.

Chapter 2

Background

2.1 The POLIANNA dataset

The *POLIANNA* dataset (*POLIcy design ANNotAtions*) is a comprehensive tool that supports the analysis and evaluation of climate policies through Machine Learning (ML). This section provides a detailed introduction to the dataset, including its creation, structure and potential applications in research.

The dataset was developed by Sewerin et al. [21] to enable systematic and efficient approaches to the analysis of policy design. It comprises annotated text spans from 18 European Union (EU) directives and regulations focusing on climate change and renewable energy. These texts have been extracted from the **EUR-Lex!** (**EUR-Lex!**) database, an official EU platform for access to legal documents. **EUR-Lex!** provides EU legal acts such as regulations, directives, decisions and opinions in an open format, making it easier to search and compare documents.

A multidimensional coding scheme forms the methodological basis of the dataset. It covers three central levels of policy design: *Policy Instrument Types*, *General Policy Design Characteristics* and *Technology Specificity*. These layers are hierarchically organised into features and tags as shown in Table 5.1, Table 5.2 and Table 5.3. An overview of the distribution of annotations can be found in the graphs 2.2, 2.3 and 2.4, while the visualisation 2.5 provides a consolidated view of all layers. The *Policy Instrument Types* include policy instruments such as tax incentives, subsidies or regulatory measures. The *General Policy Design Characteristics* include elements such as actors, compliance mechanisms or timeframes. The *Technology Specificity* focuses on the promotion of low-carbon technologies and energy sources.

The annotation was performed using the open source software INCEpTION, which allows for hierarchical structures and overlapping tags. INCEpTION is a platform for machine-aided annotation that enables the annotation of complex hierarchical data structures and thus effectively supports ML [22]. The annotated text spans contain an average length of three tokens. A *token* denotes a unit of text separated by spaces or punctuation, such as a word or a number. The short average length illustrates the precision of the annotation, as individual concepts could be accurately identified and separated. To ensure consistency and quality, Inter-Annotator Agreement (IAA) metrics such as the γ score were applied. The γ score is an extended measure of agreement between different annotators that takes into account both the selection of areas (unitising) and the categorisation. This measure corrects the agreement by the expected random value and is particularly useful for complex annotations with overlapping ranges.

In addition, the *POLIANNA* dataset provides a robust basis for the development of methods for the automated analysis of policy texts. Potential applications include the identification of trends in policy design, the study of interactions between policies, the derivation of recommendations for action in climate policy, and the development of tools to support manual coding processes.

Datenpreprocessing

Working with a data set requires not only a deep understanding of its structure and content, but also careful preparation of the data. Preprocessing steps are crucial to ensure the quality of the dataset, to provide a consistent basis for analysis, and to prepare the dataset for machine learning.

A key aspect of data cleansing was the `Article_State` column, which contained two rare states (`ANNOTATION_IN_PROGRESS` and `CURATION_IN_PROGRESS`), each of which occurred only once. As these entries were not statistically relevant, they were removed to simplify processing. The focus was then on fully annotated lines to ensure a consistent and meaningful database.

In addition to this cleansing, further steps were taken to remove irrelevant information, optimise the data structure and prepare the text data to meet the requirements of machine learning processes. These steps laid the foundation for accurate and reliable analysis.

```
df.drop(
    df[df['Article_State'].isin(['ANNOTATION_IN_PROGRESS',
    'CURATION_IN_PROGRESS'])].index, inplace=True
    )
```

Listing 2.1: Entfernen irrelevanter Daten

Cleansing the annotator columns

Analysis of the annotator columns revealed that only annotators A, B, C and F were actively annotating. The columns for inactive annotators (E, G and D) contained no data and were therefore removed to simplify the data structure and reduce unnecessary complexity. This clean-up helps to improve the efficiency of further data processing.

```
# Dropping empty columns corresponding to inactive annotators
# This reduces noise and ensures the dataset is more concise
df.drop(columns=['E', 'G', 'D'], inplace=True)
```

Listing 2.2: Bereinigen leerer Annotator-Spalten

text data cleanup

The text data contained control characters such as `\r\n`, `\xa0` and other characters inserted during extraction. These characters have been removed to ensure a consistent and high quality text base. In addition, multiple spaces were removed, as well as leading and trailing spaces, to make the texts easier to read and process.

```
# Cleaning text data by removing unwanted control characters
# and ensuring consistent spacing in text fields
annotations_df['Text'] = (
    annotations_df['Text']
    .str.replace('\r\n', ' ', regex=False)  # Replace Windows-style newlines
    .str.replace('\xa0', ' ', regex=False)  # Replace non-breaking spaces
    .str.replace('\n', ' ', regex=False)    # Replace Unix-style newlines
    .str.replace(r'\s+', ' ', regex=True)   # Replace multiple spaces with a single space
    .str.strip()                            # Remove leading and trailing spaces
)
```

Listing 2.3: Bereinigen von Textdaten

Extraction and structuring of annotations

The annotations in the *POLIANNA* dataset were extracted using regular expressions and converted into a structured form. This step facilitates analysis and visualisation by breaking down the complex annotations into individual components and allowing statistical evaluation.

```
# Extracting annotation components (layers, features, tags) using regular expressions
# This enables structured analysis and better visualization of the annotations
import re
from collections import Counter

layer_pattern = r"layer:([A-Za-z0-9_]+)"
feature_pattern = r"feature:([A-Za-z0-9_]+)"
tag_pattern = r"tag:([A-Za-z0-9_]+)"

sum_layers, sum_features, sum_tags = [], [], []

```

```
for curation_list in df['Curation']:
    curation_str = " ".join(map(str, curation_list))
    sum_layers.extend(re.findall(layer_pattern, curation_str))
    sum_features.extend(re.findall(feature_pattern, curation_str))
    sum_tags.extend(re.findall(tag_pattern, curation_str))

layer_distribution = Counter(sum_layers)
feature_distribution = Counter(sum_features)
tag_distribution = Counter(sum_tags)
```

Listing 2.4: Extrahieren von Annotationen

Visualisation of the annotations

In order to comprehensively analyse the annotations of the *POLIANNA* dataset, several visualisations have been created. These are used to illustrate the distribution of annotators, layers, features and tags, and to highlight important patterns in the data.

annotator distribution

Figure 2.1 shows a bar chart representing the number of annotations made by each annotator. This visualisation illustrates the different contributions of the annotators and helps to better understand the annotation activity.

```
# Plotting the frequency of annotations made by each annotator
# This highlights the contribution of each annotator to the dataset
import matplotlib.pyplot as plt

df['Finished_Annotators'].explode().value_counts().plot(
    kind='bar',
    xlabel='Annotators',
    ylabel='Count',
    title='Annotator Frequencies',
    rot=0
).bar_label(plt.gca().containers[0])
plt.show()
```

Listing 2.5: Visualisierung der Annotator-Frequenzen

layer distribution

Figure 2.2 illustrates the distribution of layers in the *POLIANNA* dataset. This visualisation is based on the number of annotations assigned to each layer and provides an insight into the frequency of the different layers.

```
# Extract labels and values for the layers
labels = list(layer_distribution.keys())
values = list(layer_distribution.values())

# Customize layer labels for better readability
custom_labels = [
    label.replace("Instrumenttypes", "Instrument types")
         .replace("Policydesigncharacteristics", "Policy design characteristics"
            )
         .replace("Technologyandapplicationspecificity", "Technology specificity
            ")
    for label in labels
]

# Generate colors for the pie chart
colors = plt.cm.Paired(range(len(labels)))

# Create the pie chart to display the layer distribution
```

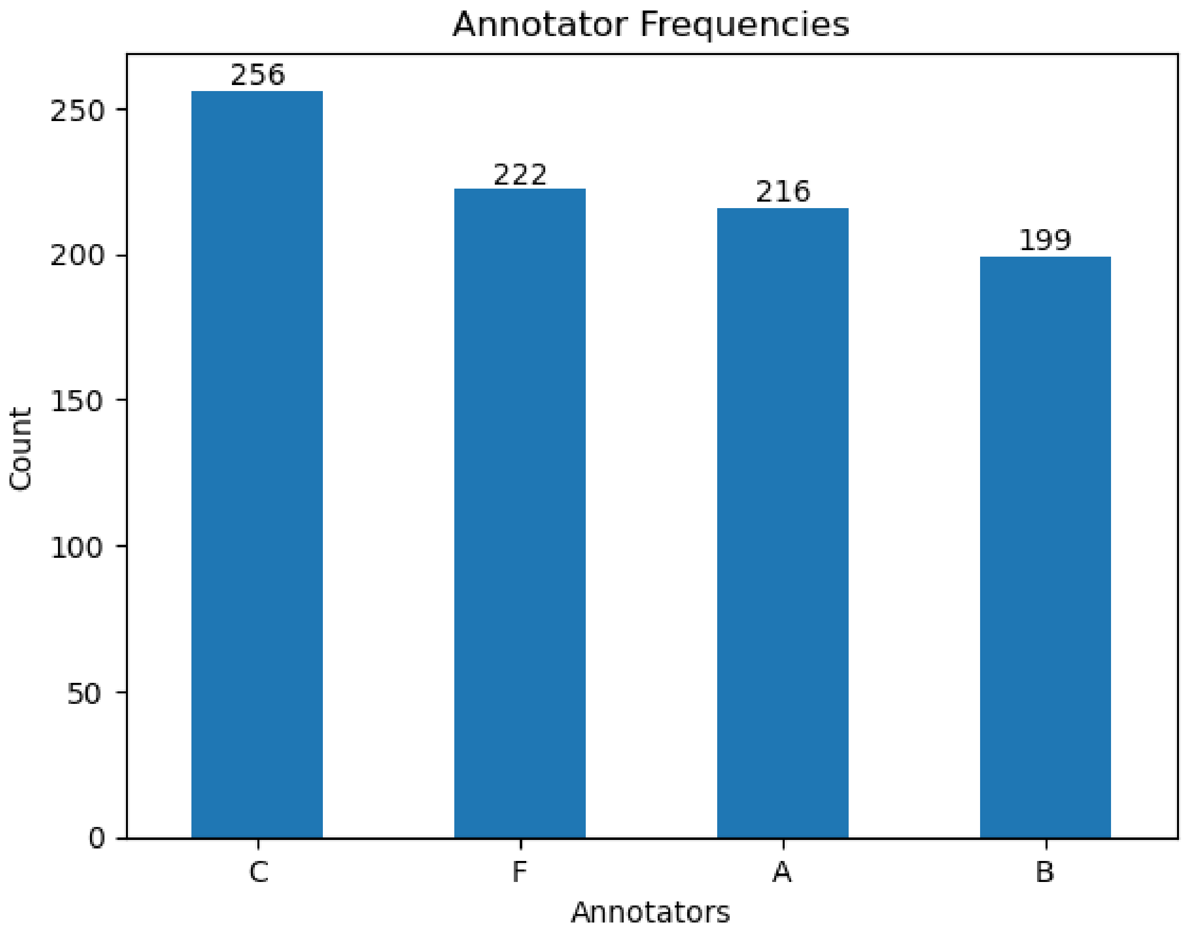

Figure 2.1: Annotator distribution in the POLIANNA dataset. Annotator C contributed the most annotations (256), followed by F (222), A (216), and B (199).

```
plt.figure(figsize=(5, 5))
plt.pie(
    values,
    labels=custom_labels,
    autopct=lambda p: f'{p:.1f}%\n({int(p*sum(values)/100)})',
    colors=colors,
    startangle=90
)
plt.title('Layer Distribution', fontsize=14)
plt.tight_layout()
plt.show()
```

Listing 2.6: Visualisierung der Layer-Verteilung

feature distribution

Figure 2.3 shows a pie chart illustrating the distribution of features in the *POLIANNA* dataset. The frequencies of the features have been calculated based on the number of corresponding annotations and are used to analyse their relevance.

```
# Extract labels and values for the features
labels = list(feature_distribution.keys())
values = list(feature_distribution.values())

# Customize feature labels for better readability
custom_labels = [
```

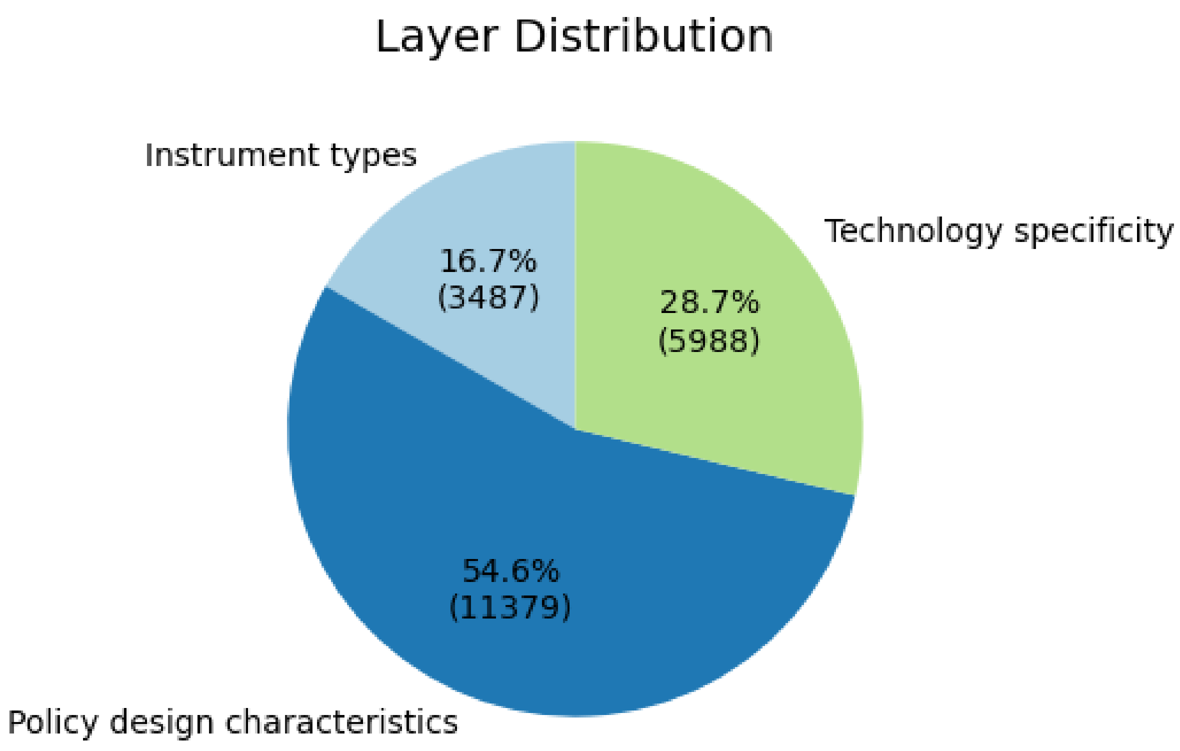

Figure 2.2: Distribution of layers in the POLIANNA dataset. The largest portion (54.6%) is associated with "Policy design characteristics", followed by "Technology specificity" (28.7%) and "Instrument types" (16.7%).

```
    label.replace("ApplicationSpecificity", "Application")
         .replace("EnergySpecificity", "Energy")
         .replace("Technologyandapplicationspecificity", "Technology specificity")
    for label in labels
]

# Generate colors for the pie chart
colors = plt.cm.Paired(range(len(labels)))

# Create the pie chart to display the feature distribution
plt.figure(figsize=(20, 20))
plt.pie(
    values,
    labels=custom_labels,
    autopct=lambda p: f'{p:.1f}%\n({int(p*sum(values)/100)})',
    colors=colors,
    startangle=90
)
plt.title('Feature Distribution', fontsize=14)
plt.tight_layout()
plt.show()
```

Listing 2.7: Visualisierung der Feature-Verteilung

tag distribution

Figure 2.4 shows the frequency of tags in the *POLIANNA* dataset. This visualisation highlights the dominant categories and allows a detailed analysis of the distribution of different tags.

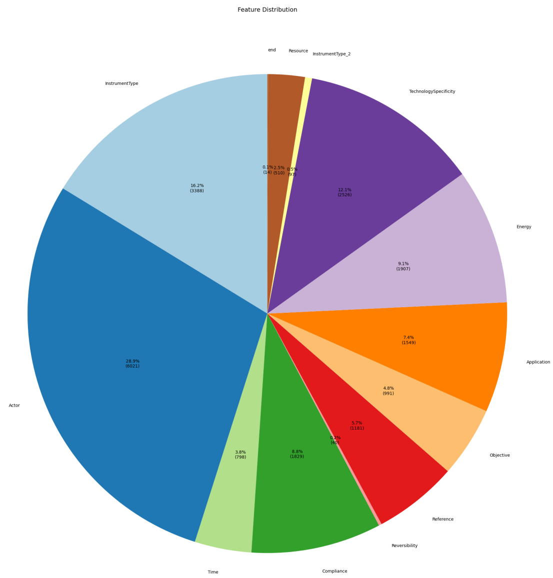

Figure 2.3: Feature distribution in the POLIANNA dataset. The most frequent feature is "Actor" (28.9%), followed by "InstrumentType" (16.7%) and "Technology specificity" (12.1%).

```python
# Import seaborn for enhanced data visualization
import seaborn as sns

# Prepare data for the bar chart
data = pd.DataFrame({
    'Tag': list(tag_distribution.keys()),
    'Frequency': list(tag_distribution.values())
})

# Sort the data by frequency in descending order for clarity
data = data.sort_values(by='Frequency', ascending=False)

# Create the bar chart to display the tag distribution
plt.figure(figsize=(12, 8))
sns.barplot(data=data, x='Frequency', y='Tag', palette='viridis')
plt.title('Tag Distribution', fontsize=16)
plt.xlabel('Frequency')
plt.ylabel('Tags')
plt.tight_layout()
plt.show()
```

Listing 2.8: Visualisierung der Tag-Verteilung

Analysis of the tag distribution 2.4 shows that the tag `Addressee_default` is by far the most common. This is followed by the tags `Form_monitoring` and `Tech_LowCarbon`, which are also key elements of the policies analysed. These dominant categories reflect key aspects of the *POLIANNA* dataset and provide valuable insights into the focus of the annotations.

The diagram 2.5 by Sewerin et al. [21] shows the hierarchical structure and distribution of the annotations across the different layers, features and tags. It clarifies the relationships between annotation levels and provides an intuitive overview of the data structure.

This visualisation provides a clear overview of the focus of the annotations and shows how the different layers and features are distributed across the dataset. The visualisation is particularly useful for understanding the hierarchy and relationships between annotations.

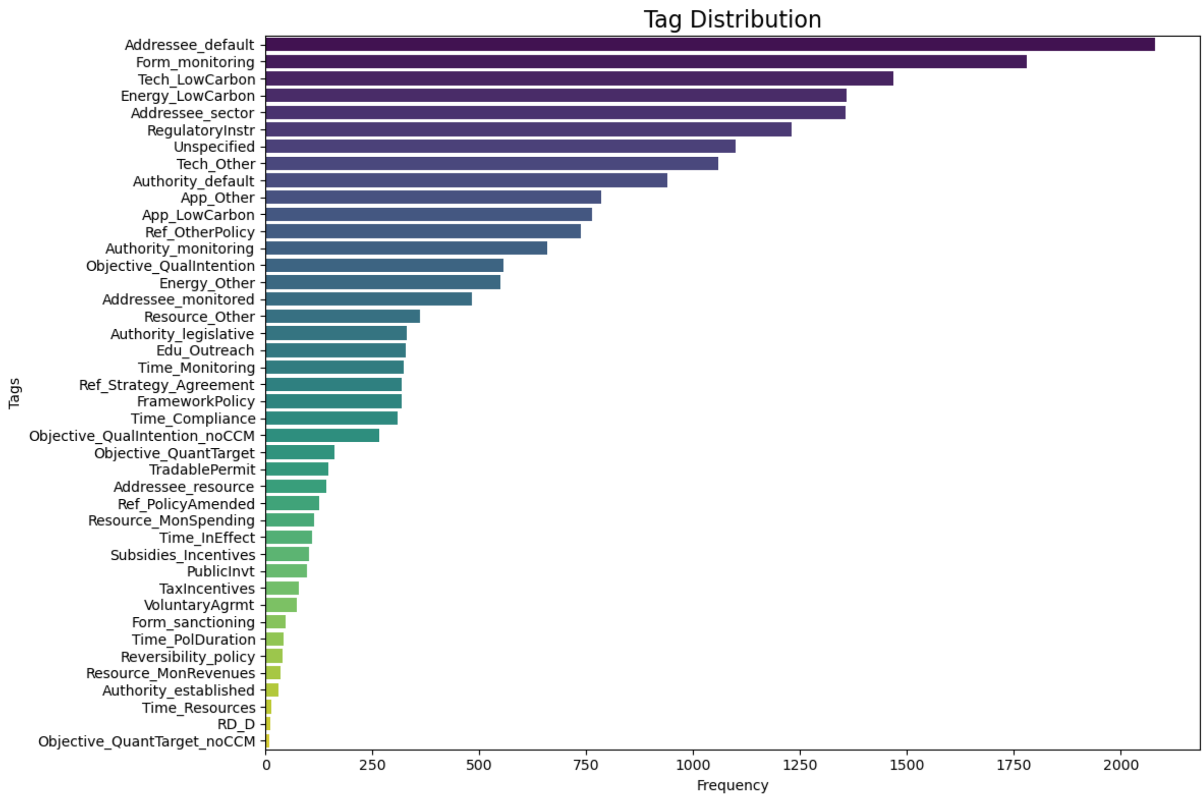

Figure 2.4: Tag distribution in the POLIANNA dataset. The most frequent tag is "Addressee_default", followed by "Form_monitoring" and "Tech_LowCarbon".

2.2 Introduction to ClimateBert

NLP has emerged as a valuable tool for understanding climate-related discourse, assessing public sentiment, and enabling effective communication strategies. General-purpose language models such as Bidirectional Encoder Representations from Transformers (BERT) (Bidirectional Encoder Representations from Transformers) have revolutionised NLP tasks, but their application to niche domains such as climate-related texts often falls short due to the unique linguistic features and specialised vocabulary inherent to these domains [?].

To fill this gap, ClimateBERT has been developed as a domain-specific language model tailored to climate-related texts [16]. Building on the foundation of pre-trained models such as BERT, ClimateBERT undergoes further domain-adaptive pre-training using a large corpus of climate-related texts. This additional pre-training phase enhances its ability to understand and process the nuanced language of climate science, policy and communication [?].

ClimateBERT builds on the architecture of BERT, using its bidirectional transformer mechanism to effectively capture contextual information. Its domain-adaptive pre-training includes large corpora of over 2 million climate-related paragraphs from diverse domains such as news articles, corporate disclosures and scientific publications [16]. The inclusion of these specialised corpora ensures that ClimateBERT is better equipped to deal with domain-specific challenges such as terminological variation and contextual ambiguity.

The development of ClimateBERT follows a three-phase training approach:

- **General Pretraining**: Use of a general corpus to establish a robust basic understanding of the language.
- **Domain Adaptive Pretraining**: Refining the model on climate-specific corpora to improve its performance for niche applications.
- **Fine Tuning**: Optimise the model for downstream tasks such as sentiment analysis, text classification and fact checking in the climate domain [23].

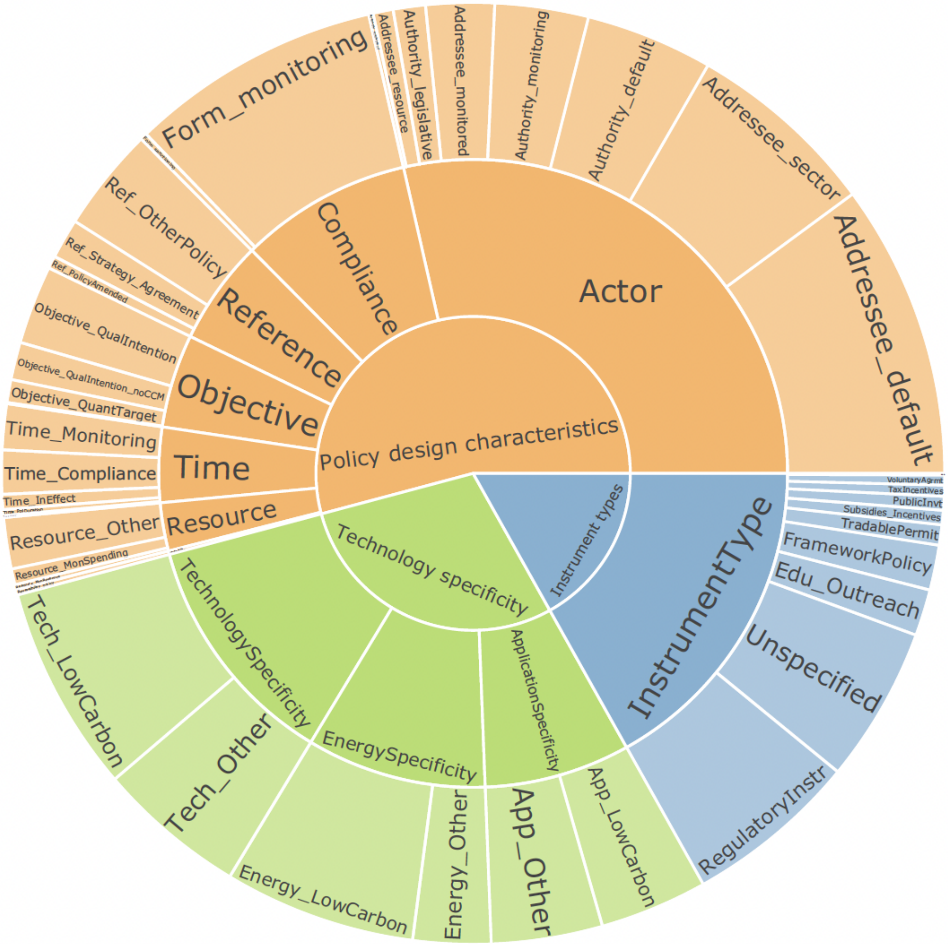

Figure 2.5: Distribution of annotations in the dataset. The circles correspond to layers, features, and tags (from the inside out). The colors highlight the three layers we annotated: Blue comprises layers, features and tags belonging to instrument types, orange those belonging to policy design characteristics, and green those belonging to technology specificity. The wider a wedge, the larger is the share of annotations belonging to this particular layer, feature, or tag. [21]

Applications

ClimateBERT has demonstrated significant improvements in several NLP tasks tailored to climate-related contexts. Key applications include

- **Sentiment Analysis**: Understanding public attitudes towards climate change as reflected in social media and news content [24].
- **Text Classification**: Categorisation of corporate disclosures and scientific articles based on climate-related topics [25].
- **Topic Modelling**: Identifying and analysing trends in climate change narratives over time and across regions [24].
- **Fact-Checking**: Verifying climate-related claims in media and research [16].

ClimateBERT represents a significant step forward in the use of AI for climate change research and communication. By enabling accurate analysis of specialised texts, it enables policy makers, researchers and stakeholders to better understand and address the complexity of climate-related challenges. Future directions include expanding its corpus, refining its architecture, and exploring novel applications in adjacent domains such as biodiversity and sustainability.

2.3 Tackling Class Imbalance with Diverse Sampling Strategies

Class imbalance is a common and significant challenge in machine learning [26].It occurs when one or more classes in a dataset are disproportionately represented compared to others [27]. This imbalance often leads to models that favour the majority class, potentially overlooking critical insights from the minority class. Such biases can lead to poor performance, particularly in high-stakes domains such as fraud detection, medical diagnosis, and rare event prediction, where minority classes often hold the most valuable information [28].

Effectively addressing class imbalance requires specialised strategies to ensure equitable learning across all classes. A fundamental starting point is the classic train-test split, which serves as the baseline for model evaluation. While simple, this approach alone is not sufficient to mitigate imbalance-related problems. Therefore, we explore advanced sampling strategies to improve model performance and robustness. Our evaluation incorporates Optuna for HPO, which provides a fair and consistent framework for comparison [19].

```
    train_data, test_val_data = train_test_split(
    annotations_df,
    test_size=0.2,
    stratify=annotations_df['Layer_encoded'],
    random_state=42
)
val_data, test_data = train_test_split(
    test_val_data,
    test_size=0.5,
    stratify=test_val_data['Layer_encoded'],
    random_state=42
)
train_dataset = TextDataset(train_data, tokenizer)
val_dataset = TextDataset(val_data, tokenizer)
test_dataset = TextDataset(test_data, tokenizer)
val_results, test_results = train_and_evaluate(
    strategy="baseline",
    train_df=train_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_baseline"
)
```

This work deals with several widely used techniques designed to rebalance datasets either by changing the distribution of training samples or by generating synthetic examples [27]. These methods, which are based on mathematical principles, are described in detail below:

2.3.1 Oversampling

Oversampling involves replicating samples from the minority class to increase their representation in the dataset [29]. Mathematically, let $D = \{(x_i, y_i)\}_{i=1}^N$ represent the dataset, where $y_i \in \{0, 1\}$ indicates the class labels. Oversampling augments the minority class C_1 such that the size of C_1 matches the majority class C_0 by duplicating samples:

$$D' = D \cup \{(x_j, y_j) \mid (x_j, y_j) \in C_1\}, \quad \text{for } |C_1| < |C_0|.$$

While simple to implement, oversampling risks overfitting, especially if the duplicated samples dominate the learning process.

```
    oversampler = RandomOverSampler(random_state=42)
X_resampled, y_resampled = oversampler.fit_resample(
    train_data[['Text']], train_data['Layer_encoded']
)
oversampled_df = pd.DataFrame({"Text": X_resampled.squeeze(), "Layer_encoded":
    y_resampled})
oversampled_dataset = TextDataset(oversampled_df, tokenizer)

val_results, test_results = train_and_evaluate(
```

```
    strategy="oversampling",
    train_df=oversampled_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_oversampling"
)
```

2.3.2 Undersampling

In contrast, undersampling reduces the majority class by randomly removing samples to balance the class distribution [29]. Formally, the dataset is modified as:

$$D' = D \setminus \{(x_k, y_k) \mid (x_k, y_k) \in C_0,\ |C_0| > |C_1|\}.$$

Although this method ensures balance, it may discard critical information, reducing the model's ability to generalize effectively.

```
undersampler = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = undersampler.fit_resample(
    train_data[['Text']], train_data['Layer_encoded']
)
undersampled_df = pd.DataFrame({"Text": X_resampled.squeeze(), "Layer_encoded":
    y_resampled})
undersampled_dataset = TextDataset(undersampled_df, tokenizer)

val_results, test_results = train_and_evaluate(
    strategy="undersampling",
    train_df=undersampled_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_undersampling"
)
```

2.3.3 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic samples for the minority class by interpolating between existing samples [17]. For a given minority sample x_i, a synthetic sample is created as:

$$x_{\text{new}} = x_i + \lambda(x_i - x_{\text{nearest}}), \quad \lambda \sim U(0, 1),$$

where x_{nearest} is one of the k nearest neighbors of x_i in the feature space. This approach introduces diversity into the dataset, reducing the risk of overfitting.

```
    vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(train_data['Text'])

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_tfidf, train_data['Layer_encoded
    '])

X_resampled_text = [
    train_data['Text'].iloc[idx] if idx < len(train_data) else "Generated Text"
    for idx in range(len(y_resampled))
]

smote_df = pd.DataFrame({
    "Text": X_resampled_text,
    "Layer_encoded": y_resampled
})

smote_dataset = TextDataset(smote_df, tokenizer)

```

```python
val_results, test_results = train_and_evaluate(
    strategy="smote",
    train_df=smote_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_smote"
)
```

2.3.4 SMOTEENN (SMOTE + Edited Nearest Neighbors)

SMOTEENN combines SMOTE with Edited Nearest Neighbors (ENN), which removes noisy samples after synthetic generation [30, 31]. Using ENN [32], a sample is retained only if its class matches the majority of its k nearest neighbors. Mathematically:

$$\text{Retain } (x_i, y_i) \text{ if } \sum_{j=1}^{k} I(y_i = y_j) > \frac{k}{2},$$

where I is the indicator function.

```python
    from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.combine import SMOTEENN


vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(train_data['Text'])

X_dense = X_tfidf.toarray()

smoteenn = SMOTEENN(random_state=42)
X_resampled, y_resampled = smoteenn.fit_resample(X_dense, train_data['
    Layer_encoded'])

smoteenn_df = pd.DataFrame(X_resampled, columns=vectorizer.get_feature_names_out
    ())
smoteenn_df['Layer_encoded'] = y_resampled

smoteenn_df['Text'] = [
    " ".join([vectorizer.get_feature_names_out()[i] for i, val in enumerate(row)
        if val > 0])
    for row in X_resampled
]

smoteenn_dataset = TextDataset(smoteenn_df[['Text', 'Layer_encoded']], tokenizer
    )

val_results, test_results = train_and_evaluate(
    strategy="smoteenn",
    train_df=smoteenn_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_smoteenn"
)
```

2.3.5 ADASYN (Adaptive Synthetic Sampling)

ADASYN focuses on generating synthetic samples near difficult-to-classify instances [18]. For each minority sample x_i, the number of synthetic samples G_i is determined based on the classification difficulty d_i, defined as:

$$d_i = \frac{\text{Number of majority neighbors}}{k}.$$

Then, synthetic samples are generated as in SMOTE but weighted by d_i, prioritizing regions where the model struggles most.

```python
    from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import ADASYN

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf_vectorizer.fit_transform(train_data['Text']).toarray()
y = train_data['Layer_encoded']

adasyn = ADASYN(random_state=42, n_jobs=-1)
X_resampled, y_resampled = adasyn.fit_resample(X_tfidf, y)

inverse_vocab = {v: k for k, v in tfidf_vectorizer.vocabulary_.items()}
resampled_texts = [
    " ".join([inverse_vocab[i] for i in np.where(row > 0)[0]])
    for row in X_resampled
]

adasyn_df = pd.DataFrame({
    "Text": resampled_texts,
    "Layer_encoded": y_resampled
})

adasyn_dataset = TextDataset(adasyn_df, tokenizer)

val_results, test_results = train_and_evaluate(
    strategy="adasyn",
    train_df=adasyn_dataset,
    val_dataset=val_dataset,
    test_dataset=test_dataset,
    output_dir="./results_adasyn"
)
```

These sampling strategies offer diverse tools for practitioners to address class imbalance, each with unique strengths and trade-offs. Selecting the appropriate method depends on the specific dataset characteristics and the application context. In the following sections, we delve deeper into the implementation and comparative evaluation of these techniques, highlighting their practical implications and effectiveness in real-world scenarios.

2.4 Hyperparameter Optimization with OPTUNA

HPO is a critical step in machine learning workflows, as hyperparameters such as learning rate, batch size and number of epochs significantly influence model performance. While traditional methods such as grid search and random search are commonly used, these approaches are often computationally expensive and inefficient, especially in high-dimensional search spaces. To address these challenges, advanced frameworks for automated HPO have been developed, among which Optuna stands out as a state-of-the-art tool [19].

Optuna is an open source framework designed to provide a flexible, efficient and easy-to-use solution for hyperparameter tuning. Unlike previous frameworks, Optuna takes a `define-by-run` approach, allowing the user to dynamically construct the hyperparameter search space at runtime. This feature, combined with intelligent sampling strategies such as the `Tree-Structured Parzen Estimator (TPE)` [20], enables Optuna to effectively balance exploration and exploitation. In addition, Optuna supports early stopping through pruning, which terminates poorly performing trials based on intermediate evaluation results, saving significant computational resources.

In this work, we will explore how Optuna can be used for HPO for a comparative analysis of different sampling strategies for class imbalance. Sampling techniques are widely used to balance datasets, particularly in classification tasks, but their performance depends on the underlying hyperparameter settings. Using Optuna, we aim to find the best hyperparameters for each sampling strategy, allowing a fair and robust comparison of their effectiveness.

```python
    import optuna

def objective(trial, train_dataset, val_dataset):
    learning_rate = trial.suggest_float('learning_rate', 1e-5, 1e-4, log=True)
```

```python
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])
    num_epochs = trial.suggest_int('num_epochs', 3, 10)

    model = AutoModelForSequenceClassification.from_pretrained(
        layer_model_name, num_labels=len(set([x['labels'].item() for x in
            train_dataset]))
    )

    training_args = TrainingArguments(
        output_dir="./optuna_results",
        eval_strategy="epoch",
        learning_rate=learning_rate,
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=batch_size,
        num_train_epochs=num_epochs,
        weight_decay=0.01,
        logging_dir="./logs",
        save_strategy="epoch",
        logging_steps=10,
        load_best_model_at_end=True,
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        tokenizer=tokenizer,
        data_collator=DataCollatorWithPadding(tokenizer),
        compute_metrics=compute_metrics,
    )

    trainer.train()
    eval_metrics = trainer.evaluate()
    return eval_metrics['eval_accuracy']

study = optuna.create_study(direction="maximize")
study.optimize(lambda trial: objective(trial, train_dataset, val_dataset),
    n_trials=10)
best_params = study.best_params
optuna_results_file = f"optuna_results_{strategy_name}.csv"
```

Chapter 3

Approach

Chapter 4

Discussion and Results

4.1 Discussion

4.2 Results

Chapter 5

Limitations and Outlook

5.1 Limitations

5.2 Outlook

Appendix

Layer	Description
Instrument types	Types of political instruments used to implement measures.
Policy design characteristics	Characteristics shaping the design of a policy, such as objectives, responsibilities, and monitoring.
Technology specificity	The degree to which a policy targets specific technologies or applications, e.g., renewable energy sources or specific technologies.

Table 5.1: Layers in the POLIANNA dataset

Feature	Description
Instrument type	Classification of specific instruments, e.g., voluntary agreements or subsidies.
Actor	Roles of affected parties or authorities, such as legislative or monitored authorities.
Compliance	Mechanisms ensuring policy compliance, such as sanctioning forms.
Objective	Policy objectives, either quantitative (e.g., emission targets) or qualitative.
Resource	Resources required for implementing the policy, such as financial means.
Reversibility	Mechanisms allowing the amendment or reversal of policies.
Time	Temporal parameters such as duration, monitoring periods, or compliance timelines.
Technology specificity	Focus on specific technologies, such as low-carbon technologies.
Energy specificity	Energy-related aspects, such as promoting renewable energy sources.
Application specificity	Application-specific aspects, such as sectoral or regional measures.

Table 5.2: Features in the POLIANNA dataset

Tag	Description
Voluntary agreement	Voluntary agreements between actors to implement climate measures.
Framework policy	Framework policies providing general guidelines or principles.
Tradable permit	Tradable certificates, such as emission allowances.
Regulatory instrument	Regulatory instruments, e.g., standards or mandates.
Tax incentives	Tax-based incentives promoting certain measures.
Subsidies and direct incentives	Subsidies or direct financial support.
Research, Development & Demonstration (RD&D)	Measures promoting research and development.

Tag	Description
Public Investment	Public investments in infrastructure or programs.
Education and outreach	Educational and informational programs.
Unspecified	Instruments without specific classification.
Default authority	Standard authority responsible for implementing or overseeing measures.
Legislative authority	Legislative bodies involved in policymaking.
Newly established authority	Authorities newly established specifically for implementation.
Monitoring authority	Authorities responsible for monitoring compliance.
Default addressee	Default addressee of measures, e.g., citizens or companies.
Resources addressee	Addressees affected by resource-related measures, such as funding or infrastructure.
Monitored addressee	Addressees monitored as part of the measures.
Sector addressee	Addressees in specific sectors, e.g., transport or energy.
Sanctioning form	Mechanisms enforcing compliance, e.g., penalties for non-compliance.
Monitoring form	Forms of monitoring, such as reports or inspections.
Reference to other policy	Reference to other political measures or strategies.
Amendment of policy	Amendments or updates to existing policies.
Reference to strategy or agreement	References to specific strategies or international agreements.
Quantitative target	Targets with measurable outcomes, e.g., reducing emissions by 40%.
Qualitative intention	Statements of intent that are not directly measurable, e.g., fostering innovation.
Monetary revenues	Revenue generated by measures, e.g., through taxes or fees.
Monetary spending	Financial expenditures for implementing measures.
Provision for reversibility	Mechanisms allowing for the amendment or reversal of measures.
Policy duration time	Timeframe during which the policy is active.
Monitoring time	Timeframe for monitoring the implementation.
Resources time	Timeframe for the provision of resources.
Compliance time	Timeframe for meeting compliance requirements.
In-effect time	Time period during which measures are effective.
Low-carbon technology	Technologies with low CO_2 emissions.
Other technology	Other technologies not explicitly categorized.
Low-carbon energy source	Low-carbon energy sources, such as renewables.
Other energy source	Other energy sources not explicitly categorized.
Low-carbon application	Applications with low CO_2 emissions.
Other application	Other applications not explicitly categorized.

Table 5.3: Tags in the POLIANNA dataset

Bibliography

[1] C. Dupont and S. Oberthür, "Insufficient climate policy integration in eu energy policy: The importance of the long-term perspective," *Journal of Contemporary European Research*, vol. 8, no. 2, pp. 228–247, 2012.

[2] United Nations Framework Convention on Climate Change, "Unfccc timeline: Key milestones in the evolution of international climate policy," 2025.

[3] United Nations Framework Convention on Climate Change, "The kyoto protocol: A key step in combating climate change," 2025.

[4] U. N. F. C. on Climate Change (UNFCCC), "Adoption of the paris agreement," 2015.

[5] S. Oberthür and H. E. Ott, *The Kyoto Protocol: International Climate Policy for the 21st Century.* Berlin, Heidelberg: Springer, 2001.

[6] U. Nations, *United Nations Framework Convention on Climate Change (UNFCCC).* 1992.

[7] B. Deutschland, "Bundes-klimaschutzgesetz (ksg)," 2019.

[8] Deutscher Bundestag, "Debatte zum bundes-klimaschutzgesetz," 2019.

[9] B. für Wirtschaft und Energie (BMWi), "Integriertes energie- und klimaprogramm," 2007.

[10] E. Chinchio, P. Martin-Olmedo, N. Di, and A. Franco, "Quantification of health impacts in eu chemical policy evaluations," *European Commission Publications*, 2025.

[11] B. für Wirtschaft und Klimaschutz (BMWK), "Monitoringbericht der expertenkommission zum energiewende-monitoring," tech. rep., n.d.

[12] B. für Wirtschaft und Klimaschutz (BMWK), "Monitoring-prozess zur energiewende," n.d.

[13] L. Chen, Z. Chen, Y. Zhang, Y. Liu, A. I. Osman, M. Farghali, J. Hua, A. Al-Fatesh, I. Ihara, D. W. Rooney, and P.-S. Yap, "Artificial intelligence-based solutions for climate change: a review," *Environmental Chemistry Letters*, vol. 21, pp. 2525–2557, 2023.

[14] N. Ahmed, A. K. Saha, M. A. Al Noman, J. R. Jim, M. Mridha, and M. M. Kabir, "Deep learning-based natural language processing in human–agent interaction: Applications, advancements, and challenges," *Natural Language Processing Journal*, vol. 9, p. 100112, 2024.

[15] J. I. Lewis, A. Toney, and X. Shi, "Climate change and artificial intelligence: assessing the global research landscape," *Discover Artificial Intelligence*, vol. 4, p. 64, 2024.

[16] N. Webersinke, M. Kraus, J. A. Bingler, and M. Leippold, "Climatebert: A pretrained language model for climate-related text," *arXiv preprint*, 2022. Presented at the Association for the Advancement of Artificial Intelligence (AAAI) conference.

[17] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, 06 2002.

[18] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," pp. 1322–1328, 2008.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[20] S. Watanabe, "Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance," 2023. Available at https://arxiv.org/abs/2304.11127.

[21] S. Sewerin, L. H. Kaack, J. Küttel, F. Sigurdsson, O. Martikainen, A. Esshaki, and F. Hafner, "Towards understanding policy design through text-as-data approaches: The policy design annotations (polianna) dataset," *Scientific Data*, vol. 10, p. 896, 2023.

[22] J.-C. Klie, M. Bugert, B. Boullosa, R. E. de Castilho, and I. Gurevych, "The inception platform: Machine-assisted and knowledge-oriented interactive annotation," in *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, (Santa Fe, New Mexico, USA), pp. 5–9, August 20–26 2018. This work is licensed under a Creative Commons Attribution 4.0 International License. License details: `http://creativecommons.org/licenses/by/4.0/`.

[23] J. A. Bingler, M. Kraus, M. Leippold, and N. Webersinke, "Cheap talk and cherry-picking: What climatebert has to say on corporate climate risk disclosures," *Finance Research Letters*, vol. 47, p. 102776, 2022. This is an open-access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

[24] V. S. Anoop, T. K. A. Krishnan, A. Daud, A. Banjar, and A. Bukhari, "Climate change sentiment analysis using domain specific bidirectional encoder representations from transformers," *IEEE Access*, vol. 12, pp. 114912–114922, 2024. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.

[25] N. Hastreiter, "Can investor coalitions drive corporate climate action?," Working Paper 415, Grantham Research Institute on Climate Change and the Environment, London School of Economics and Political Science, 2024.

[26] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Cham, Switzerland: Springer Nature Switzerland AG, 2018.

[27] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[28] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[29] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[30] G. Husain, D. Nasef, R. Jose, J. Mayer, M. Bekbolatova, T. Devine, and M. Toma, "Smote vs. smoteenn: A study on the performance of resampling algorithms for addressing class imbalance in regression models," *Algorithms*, vol. 18, no. 1, 2025.

[31] I. learn developers, *SMOTEENN: Combine over- and under-sampling using SMOTE and Edited Nearest Neighbours.*, 2025.

[32] I. learn developers, *Edited Nearest Neighbours: Under-sampling method.*, 2025.