University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Data Science	Spring semester, 2023 Open / Restricted access
Writer: Audun Stjernelund Lien	…… (Writer's signature)
Faculty supervisor: Vinay Jayarama Setty	
Thesis title: Machine learning to detect corporate greenwashing.	
Credits (ECTS):	
Key words: - Web scraping - Transformers - Natural Language Processing - Multi-class classification - Multi-label classification - Transfer-learning	Pages: 54 + enclosure: 6 Stavanger, 13/06/2023 Date/year

Front page for master thesis
Faculty of Science and Technology
Decision made by the Dean October 30th 2009

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Machine learning to detect corporate greenwashing

Master's Thesis in Computer Science
by

Audun Stjernelund Lien

Internal Supervisors

Vinay Jayarama Setty

Reviewers

Reviewer 1

Reviewer 2

June 13, 2023

Abstract

This master thesis focuses on developing an automatic approach to detect corporate greenwashing. To achieve this, data must be collected, and green claims found from this data must be fact checked. The first step is to collect data by scraping. The web scrapers in this thesis were designed to extract comprehensive information about companies from their websites and reports using two datasets as benchmarks. The Fauna dataset was scraped using a recursive web scraper that extracted data from sub-pages linked to each company's website. The CICERO Shades of Green dataset was scraped using a scraper that visited each link in the dataset to extract the text from each report made by CICERO.

The collected datasets underwent preprocessing to ensure compatibility with machine learning models. The texts scraped from the Fauna dataset were often excessively long due to the abundance of information on the websites. These texts were summarized using a Transformer model, and irrelevant texts were manually removed from the dataset. In the case of the Cicero dataset, text augmentation was applied to expand the dataset and investigate its impact on model performance.

To address the limited data availability, transfer-learning techniques including zero, one, and two-shot learning were applied to both the Fauna and Cicero datasets. These techniques leverage pre-trained models to learn from a small amount of labeled data. Additionally, fine-tuned models were implemented specifically for the Cicero dataset to provide a basis for comparison. The trained models achieved superior performance to the transfer-learning models, suggesting that training large models with limited training data remains an effective approach.

Acknowledgements

I wish to sincerely thank my supervisor Vinay Setty for his excellent guidance through the assignment. His knowledge of AI models, natural language processing and claim detection helped me through many issues. I would never have gotten as far as I did without his expertise.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Greenwashing is a broad term. In general, it covers the organization's misleading information and communication on the performance of environmental public image. Corporate greenwashing refers to the act of making false or exaggerated environmental claims to improve a company's image or reputation. In today's society, environmental concerns are a big factor for consumers in their decision-making when buying products. In 2019, Accenture [1] conducted a survey where 72% claimed to buy more environmentally friendly products than they did five years ago. This causes an increase in the advertising of sustainable products. Many of these green claims are overstated or falsified. In, [2] greenwashing is defined as the practice of misleading consumers through unsubstantial, false, or vague sustainability claims. This kind of disinformation affects consumers, giving companies an edge in the market they should not have. In other words, unfair marketing can cause consumers to lose their trust in the green market. Another impact of greenwashing is that accurate measurement of environmental performance is impeded. So not only are the companies worse for the environment than they are claiming, but they are also hampering the research on the environment, which is an important key to saving the world. These are some aspects of why it's important to investigate greenwashing and hold companies accountable for their green claims. The consequences of greenwashing can be serious, both for consumers and for the environment.

Greenwatch [3] claims that one of the challenges when detecting greenwashing is the amount of communication channels organizations use to convey their performance on Sustainable Development Goals (SDG). These claims are mostly published on websites, reports, and social media. In Norway, there could realistically be a few hundred organizations worth investigating for greenwashing. When so many organizations have so

many channels to search for information, it's really difficult and time-consuming for a person to manually find green claims. This is one of the motivations for developing an automatic detector of green claims.

This master's thesis focuses on investigating techniques for identifying green claims across various communication channels and handling large volumes of data. The primary objective is to explore and develop methods that can effectively detect environmentally friendly assertions made by companies across diverse platforms, mainly websites and reports. By addressing the challenges posed by the sheer volume of data available, the thesis aims to contribute to developing an automatic solution to detect corporate greenwashing.

The thesis will explore various approaches for extracting the data, Additionally, this thesis will analyze different models and learning techniques. The research endeavor encompasses the utilization of the newest AI models such as `ChatGPT`, including the latest advancements in high-performance `Transformer` models. These cutting-edge models serve as the foundation for exploring innovative techniques and methodologies.

1.2 Objectives

This thesis will implement models and solutions that can be helpful to detect greenwashing. After the approach and results are discussed, the thesis will try to discuss the following research questions.

- Data Collection and Web Scraping
  - RQ1: What datasets currently exist for greenwashing detection and what is needed?
- Multi-class and Multi-label Classification
  - RQ2: Can greenwashing detection be modelled as a multi-class and multi-label classification task.
- Model Performance Analysis
  - RQ3: Which are the best suitable Transformer-based models for greenwashing detection task.

1.3 Approach and Contributions

Detecting greenwashing using machine learning is a new research area that has gained significant attention in recent years. The task of detecting greenwashing refers to distinguishing between misleading or false green claims from genuine environmentally friendly initiatives. With a machine learning approach, this means finding these claims automatically and then fact check them. To find these claims automatically, data must be collected, mainly by using web scraping methods. Data can be collected by programmatically extracting information from websites, such as corporate sustainability reports, press releases, product descriptions, and social media platforms. Web scraping allows for systematic and large-scale data collection from diverse sources. The aim is for this data to be used as input from a machine learning model to find a green claim. With this data and labeled datasets, machine learning models can be trained to learn to find green claims.

The challenges with this approach are many. Firstly, there is a small amount of labeled data that can be used. Also, labeled data is often very generalized, which can pose challenges when dealing with specific and nuanced claims. The existing labeled datasets may not cover the full range of deceptive practices and tactics employed by companies engaging in greenwashing. Consequently, ML models trained on such data may struggle to accurately detect and classify more subtle or context-dependent instances of greenwashing. The second challenge is that when the data is collected, there are usually a huge amount of text. Models designed for text processing often face limitations when handling large amounts of text, leading to performance degradation as the text size increases. Therefore, methods of reducing the size of the data must also be explored.

Let's take the example of a fictional company called "EcoTech Solutions" that claims to be an eco-friendly technology company. They claim that their computers are made of hundred percent recyclable products. This is stated in the product page of the computers. In a general case, there is no way of knowing where this information is, so all text from the web-site must be collected. Suppose the collected text has a total of 5,000 words from "EcoTech Solutions," but the model is only capable of handling a maximum of 2,000 words. In this scenario, the text must be summarized down to 2,000 words while ensuring that the crucial information about the green claim is preserved. If the model is well-trained, it can identify the claim regarding their computers, which can then be subjected to fact-checking.

This thesis utilizes analysis of Fauna and Cicero as the foundation for data collection and machine learning techniques, aiming to identify and detect corporate green claims.

Fauna is a company that rates companies in the Norwegian market based on certain criteria. This thesis has tried to develop an automated solution for rating companies based on information extracted from their respective websites. This process involves scraping information from various websites and subsequently condensing the extracted text into a manageable size for machine learning models. Clustering was used to group the companies because many of them shared criteria. Then a multi-label classification technique was used to predict what companies fulfilled certain criteria.

CICERO is a center for climate research. They have published research, "Shades of Green", where companies are classified based on environmental risks and ambitions of companies. The arguments for each classification are published in reports. This dataset is scraped and utilized for fine-tuning and training transformer models, as well as utilizing transfer learning techniques on the biggest AI models.

1.4 Outline

In this thesis, the following chapters present a comprehensive exploration of the main points and structure of the research:

- Related Work
  - This chapter provides an overview of previous studies and works related to the topic of the thesis. It discusses similar research endeavors and establishes the theoretical background necessary to understand the thesis.
- Approach
  - This chapter details the methodology and approach employed in the thesis. It covers aspects such as data scraping techniques, implementation of models, and any other relevant procedures utilized during the research.
- Evaluation
  - This chapter presents the models used in the thesis, describes the dataset utilized and analyzes the results obtained from the experiments. It offers a comprehensive evaluation of the implemented models and their performance.
- Conclusion
  - This chapter summarizes the main findings of the research, discusses the implications of the results, and provides recommendations for future work. It concludes the thesis by highlighting the contributions made and their significance in the broader context of the research field.

Chapter 2

Related Work

Woloszyn et al. [4] states that from a computer science perspective, detecting greenwashing automatically is a problem that is yet to be solved. This is because it's an area of research that is relatively new and underexplored. However, detecting and fact-checking fake news is a similar problem, where tools have been made to detect false claims. This chapter will discuss some of the challenges to automatically detect greenwashing, approaches that already have been tried, and some theory needed to understand the thesis.

2.1 Corporate Greenwashing

Corporate greenwashing is a phenomenon where companies engage in misleading or deceptive environmental claims to improve their public image and attract customers who are concerned about environmental issues. It is a growing concern, as consumers increasingly prioritize environmentally sustainable products and services, and companies seek to capitalize on this trend.

terraChoice [5] analyzed 5,296 environmental claims made by 1,018 consumer products in the United States and Canada. The study found that more than 95% of the products made at least one false or misleading environmental claim. This suggests that greenwashing is a widespread problem and that consumers may be exposed to misleading information about the environmental impact of the products they buy.

Furlow [6] investigated the prevalence of greenwashing in the hotel industry. The study found that many hotels engage in greenwashing by making vague or misleading environmental claims and that this can lead to consumer skepticism and a loss of trust in green marketing.

2.1.1 Machine Learning for Corporate Greenwashing Detection

Machine learning has the potential to aid in detecting greenwashing, but there are limitations to its effectiveness in doing so. One challenge is the limited availability of data to train machine learning algorithms. As environmental claims can be subjective, determining what constitutes greenwashing is difficult, and labeling data accurately can be a challenge. Machine learning algorithms may not be able to detect subtle forms of greenwashing, such as over-generalization or vagueness, which can make it challenging to identify greenwashing effectively.

One notable limitation is the limited amount of research available in the field. Detecting greenwashing automatically involves two essential steps: first, identifying the claim and secondly verifying it. Unfortunately, this thesis did not discover any relevant research addressing the first step, which focuses on claim identification. However, a research paper was found that specifically tackles the latter challenge of claim verification.

Diggelmann et al. [7] presents a new dataset designed to support the fact-checking of real-world climate claims. The authors note that while climate change is a pressing issue, it can be difficult for non-experts to verify the accuracy of claims made by public figures and organizations.

To address this issue, the authors developed a dataset of real-world climate claims, sourced from publicly available documents and statements made by public figures and organizations. The dataset includes 9,323 unique claims, covering a range of topics related to climate change, such as the causes of climate change, the impacts of climate change, and potential solutions.

The authors also conducted a preliminary evaluation of the dataset, using a set of baseline models to assess the difficulty of fact-checking the claims. They found that the dataset poses significant challenges for fact-checking, due to the complexity of the language used and the diversity of the topics covered.

The authors note that the Climate-Fever dataset can be used to support the development of new fact-checking models and tools, which could help to improve the accuracy of information about climate change. By providing a large and diverse dataset of real-world climate claims, the authors hope to facilitate the development of new approaches to fact-checking that are better able to handle the complexity of climate-related information

To address the challenge of overcoming the first challenge of identifying green claims, this thesis will explore some research and theory explained below.

2.2 Multi-Label Text Classification Using Transformers

Transformers have become a popular tool for natural language processing (NLP) tasks, including multi-label classification. They work by using a self-attention mechanism to process input sequences, which allows them to learn complex relationships between words and capture long-term dependencies in the data. This makes them particularly effective for handling large datasets with complex input structures, such as text data.

A paper that has used transformers for multi-label classification is Liu et al. [8]. The authors achieved state-of-the-art performance on several benchmark datasets, including the Amazon-670K[1] and Wiki-500K[2] datasets, which contain text reviews labeled with multiple topics. On the Amazon-670K dataset, their model achieved a micro-F1 score of 87.18%, outperforming the previous state-of-the-art model by over 1 percentage point. On the Wiki-500K dataset, their model achieved a micro-F1 score of 95.48%, again outperforming the previous state-of-the-art model by over 1 percentage point.

In Bhavana R and Prabhu [9], the authors fine-tuned the BERT transformer model for multi-label text classification and achieved state-of-the-art performance on several benchmark datasets, including the Reuters-21578[3] and OHSUMED datasets[4]. On the Reuters-21578 dataset, their model achieved a micro-F1 score of 94.50%, outperforming the previous state-of-the-art model by over 1 percentage point. On the OHSUMED dataset, their model achieved a micro-F1 score of 57.53%, outperforming the previous state-of-the-art model by over 3 percentage points.

Overall, these papers demonstrate the effectiveness of transformers for multi-label classification tasks and the potential of deep learning approaches for natural language processing. The ability of transformers to learn complex relationships between words and capture long-term dependencies in the data makes them a powerful tool for handling large and complex datasets in NLP tasks.

2.3 Transfer Learning

Transfer learning is a technique in machine learning that involves leveraging the knowledge learned from one task to improve performance on a related task. It is particularly useful when there is a limited amount of labeled data available for the task at hand, as it allows for the model to learn from a larger dataset that is related to the target task. With

[1] https://www.kaggle.com/c/extreme-classification-amazon
[2] http://manikvarma.org/downloads/XC/XMLRepository.html
[3] https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection
[4] http://disi.unitn.it/moschitti/corpora.htm

the increasing availability of large-scale pre-trained models such as BERT, GPT-3, and Flan-T5, it has become easier to leverage these models as a starting point for various downstream tasks. These pre-trained models are trained on massive amounts of data and can capture a wide range of contextual information, which makes them highly effective for transfer learning. Brown et al. [10] has shown that large language models can perform competitively on these tasks, with way less data than is required by smaller models.

Transfer learning can be categorized into several different types, including zero-shot learning, one-shot learning, and few-shot learning. Each of these approaches involves different levels of prior knowledge and labeled data available for the target task. Examples with classes of dogs and cats will be utilized to illustrate the distinctions among zero-shot, one-shot, and few-shot learning.

2.3.1 Zero-Shot Learning

Zero-shot learning is a type of transfer learning where a model is trained to recognize objects or concepts that it has never seen before. This is done by leveraging the knowledge learned from a related task or dataset and using it to make predictions about novel objects or concepts. The model is given a set of classes to predict and then asked to classify a text based on only prior knowledge. An example prompt can be seen in Figure 2.1.

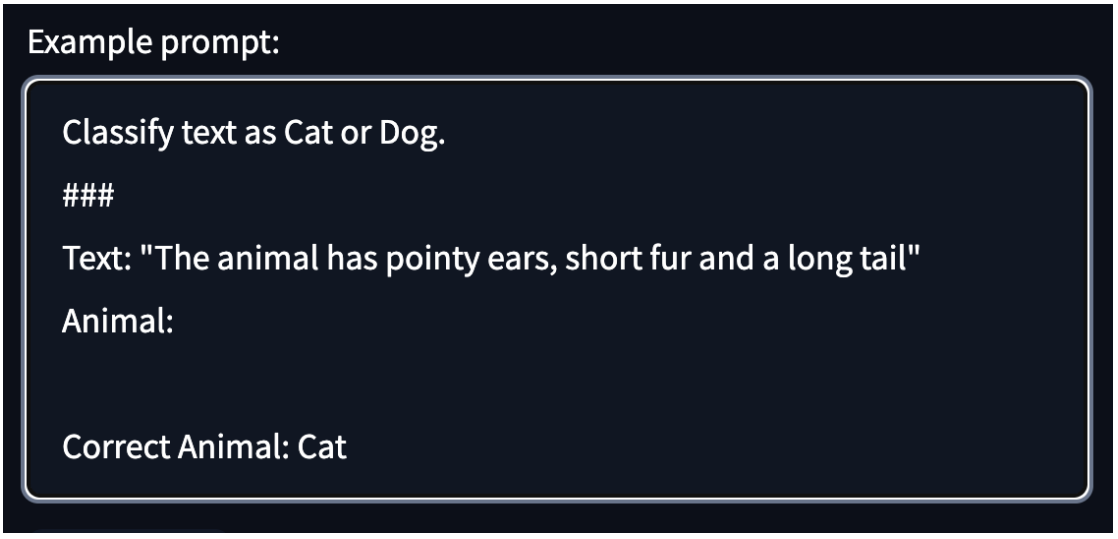

Figure 2.1: Example prompt of zero-shot learning

2.3.2 One-Shot Learning

One-shot learning is a type of transfer learning where a model is trained to recognize new objects or concepts with only one or a few examples. This is done by leveraging

the knowledge learned from a larger dataset of related objects or concepts and using it to make predictions about new examples with limited labeled data. The model is given a set of classes to predict as well as one example from each class. Therefore the model predicts on mostly prior knowledge but also utilizes some examples to make predictions. An example of a one-shot prompt can be seen in Figure 2.2.

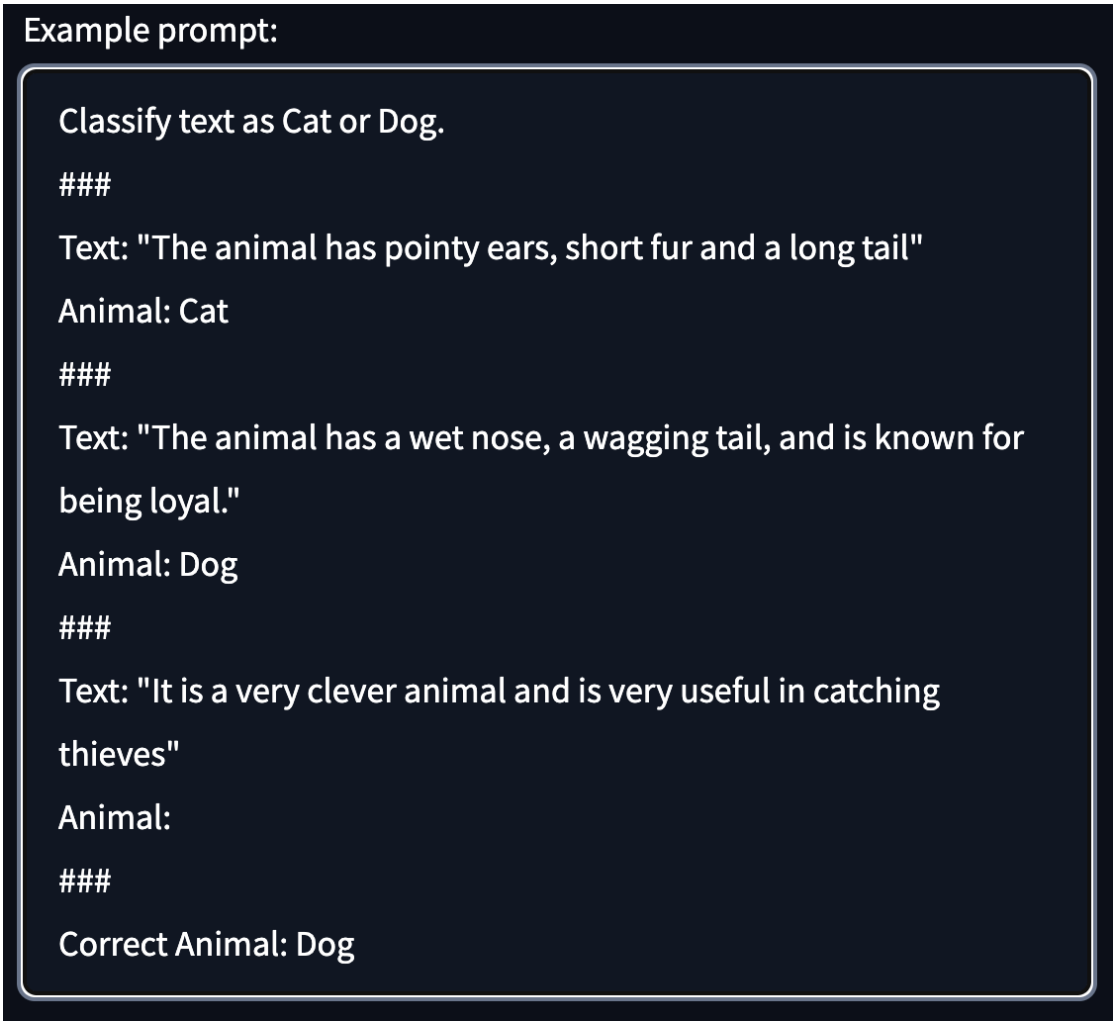

Figure 2.2: Example prompt of one-shot learning

2.3.3 Few-Shot Learning

Few-shot learning is a type of transfer learning where a model is trained to recognize new objects or concepts with a small number of examples. This is done by leveraging the knowledge learned from a larger dataset of related objects or concepts and using it to make predictions about new examples with limited labeled data. In this example, the model is given two examples per class and asked to predict what animal the last text describes Figure 2.3.

2.4 Fact checking

Fact-checking is the process of verifying the accuracy of claims or statements made by individuals or organizations. In recent years, the rise of fake news and misinformation has increased the importance of fact-checking, particularly in the context of social media and online news sources. Many organizations have emerged to help address this issue, including Factiverse.

2.4.1 Fake News Detection

Factiverse[5] is a fact-checking platform that uses artificial intelligence (AI) to analyze and verify claims made in news articles and social media posts. The company's system is designed to analyze large amounts of data and identify patterns and inconsistencies that may indicate false or misleading information.

One way that Factiverse has helped to address the issue of fake news is by developing a system that can identify and flag potentially misleading or false information in news articles and social media posts. The company's AI algorithms are designed to analyze the content of articles and posts, as well as the sources of the information, to determine the likelihood that the information is accurate.

In addition to using AI, Factiverse also relies on human fact-checkers to review and verify information. This combination of automated and manual fact-checking helps to ensure the accuracy of the information presented on the platform.

2.5 Green rating

2.5.1 Fauna

Fauna [6] is a company where users can earn points if they shop from a green brand. For a brand to be green, it must fulfill certain criteria based on the category of the brand. A fashion brand has different criteria than a food brand. Fashion has criteria like ”Sustainable raw materials” and ”Repair for free” while food has ”Local Produce” and ”Foodbox”. Each criterion is weighted a certain amount based on importance for the environment. A brand is rated from 0 to 100 by adding up the weights from each criterion that is fulfilled.

[5] https://www.factiverse.no/
[6] https://fauna.eco/

Fauna has a dataset of approximately 360 brands that operate in the Norwegian sector. Every brand where a belonging web page was found is manually rated by using information from the web page to decide if the green criteria are fulfilled. An example of an entry in the dataset can be seen in Figure 2.4.

2.5.2 CICERO

Center for International Climate Research (CICERO) has published a report series CICERO Shaded of Green [11]. The series focuses on assessing the environmental and climate impacts of various economic sectors and policies, as well as exploring potential solutions for sustainable development. The series have received several climate bond awards and external assessment provider awards from investors.

The series covers a wide range of topics, including climate change mitigation and adaptation, energy production and consumption, transportation, land use and forestry, and international climate policy. The reports are based on rigorous scientific research and analysis and aim to provide policymakers and stakeholders with evidence-based recommendations for achieving a more sustainable and resilient future.

Each assessment is given a rating based on environmental performance. There are five categories where dark green is best and brown is worse. Each rating and corresponding criteria can be seen in Figure 2.5. On CICERO's website, there is a list of public reviews. All assessments with rating "dark green, "medium green" and "light green" are listed with the report as a PDF.

2.6 Scraping

Web scraping (also known as data scraping or web harvesting) refers to the process of automatically extracting data from websites using software tools. This data can be used for a wide range of purposes, such as research, analysis, and modeling. Web scraping typically involves using a program to visit one or more web pages, extract the relevant information from the page, and save it to a local file or database. This information may include text, images, links, and other data that can be parsed and analyzed by data scientists. Web scraping is often used by data scientists to collect and analyze data from a variety of sources, including social media, news websites, e-commerce platforms, and more. By automating the process of data collection, web scraping can help data scientists to gather large amounts of data quickly and efficiently, allowing them to focus on analyzing and interpreting the data.

Factiverse has made a scraper called `ACDC`, which is written in `JavaScript` and `Rust`. The scraper collects information from an organization's web page. The various type of information collected are called signals. For instance, one signal called "Accounts" saves information about the organization's results such as operating income. In another signal, "Homepage", the URL of the organization's web page is saved as well as the HTML of the URL. All the signals for an organization are saved as an entry in a `MongoDB`[7] database.

[7] https://www.mongodb.com/

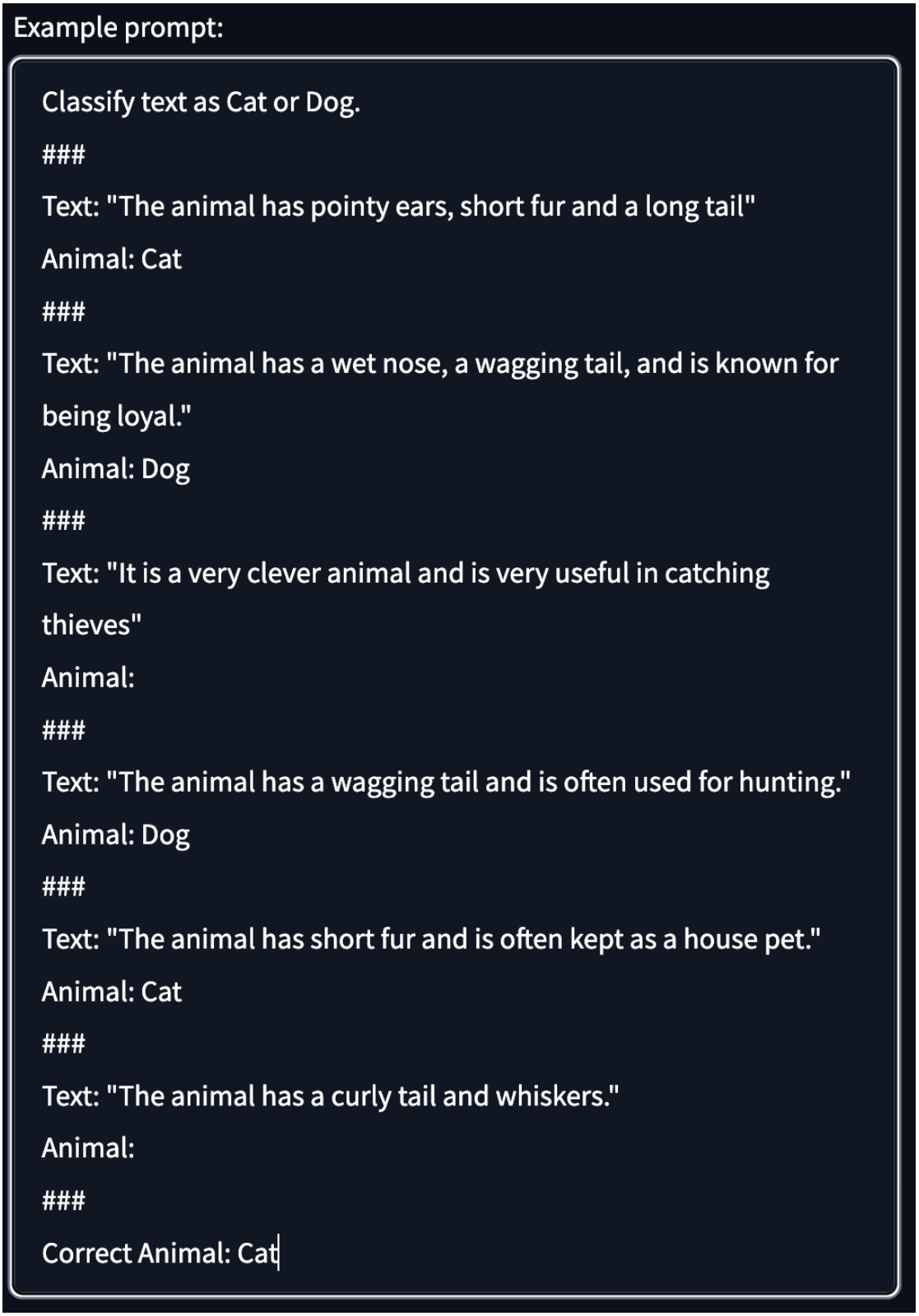

Figure 2.3: Example prompt of few-shot learning

A	B	C	D	E	F	G	H	I	J
id	**Title**	**URL**	**Is Partner**	**Green Score**	**Categories**	**Green Impact**			
3	Example Brand	ExampleBrand.com	t	100	fashion	Main criteria ☐ Rental, main service ☐ Second hand, main service ☐ Repair, main service ☑ Low impact products ☐ Do It Yourself Additional criteria ☑ Sustainability reporting ☐ CO2 emission reporting ☐ Miljøfyrtårn ☐ Rental, additional service ☑ Second hand, additional service ☐ Repair, additional service ☐ Repair for free ☐ Sustainability filtering Criteria FAQ			

Figure 2.4: Example of an entry in the Fauna dataset

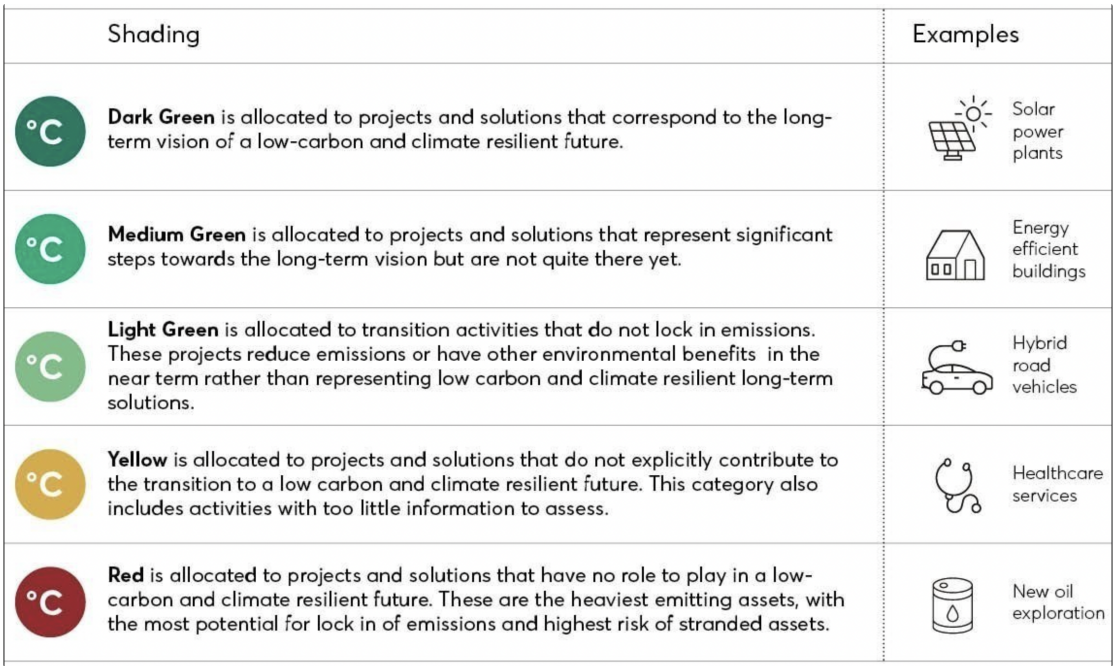

Figure 2.5: Ratings and criterias for CICERO Shades of Green

Chapter 3

Approach

3.1 Collecting Data from Organizations With a Custom Web Scraper

When researching corporate greenwashing, gathering data is fundamental. Most frequently, organizations put information on their web page. Therefore, building a web scraper that can collect this information can be a powerful tool to automatically research an organization. In section 2.6, Factiverse's `ACDC` scraper was introduced.

`ACDC` is written in JavaScript. This scraper will first be ported to Python. Python has better integration with data analysis tools that helps when analyzing the data that are being scraped. Python also has more support for web scraping libraries, such as `Beatifulsoup`[1] which is a great tool to parse scraped text. Also, other problems in this thesis will be written in Python. The scraper will have better integration with these problems if it's ported to Python.

One more downside of the scraper is it doesn't collect text from the web pages. Only the `HTML` of the homepage of each organization is saved. In this research, the text from every sub-page of the organizations will be needed since there is no way of knowing what sub-domain an organization can write about sustainability. After porting the basic `ACDC` scraper to Python, it will also be extended to collect, parse, and save all text from every sub-domain linked to the homepage of an organization.

[1] https://beautiful-soup-4.readthedocs.io/en/latest

3.1.1 Porting the Web Scraper to Python

As stated in section 2.6, the scraper collects different signals from different scripts. `Homepage` is the signal responsible for requesting the `HTML` from a URL and is the only relevant script for this thesis. This is the script that will be ported to Python.

The homepage is a Node.js program that scrapes company homepages and saves the HTML body to a MongoDB database. The script first queries the database to find organizations to scrape. It then iterates through all the organizations. It then searches `Yahoo` for the company's homepage URL and checks that the URL is valid and not blacklisted. If the URL is valid, the script fetches the page and saves it to the database. The script uses the `got` library to make `HTTP requests`, `cheerio` to parse the `HTML response`, and `ora` to display a progress spinner. The script also uses the `normalize-url` library to normalize URLs and the sleep function from a custom utils.js module to add delays to the scraping process. In the script, the `async` keyword is used to define asynchronous functions that perform non-blocking operations. These functions return a Promise that resolves with the function's return value or rejects with an error if one occurs. By using the `await` keyword inside another asynchronous function, the code can wait for the result of the asynchronous operation before continuing execution, without blocking the thread. This allows for more efficient use of resources and better handling of I/O-bound tasks, such as making HTTP requests or querying a database.

When porting to Python, the `googlesearch` module is substituted for Yahoo. Python's `request` module is changed for `got` to make the `HTTP request`. `BeatifulSoup` is used to parse this response. `halo` is used to display a spinner while Python's asynchronous frameworks `asyncio` is used instead of the `async` keyword. These are the most important changes regarding the modules. The rest of the code is translated directly from JavaScript to Python.

3.1.2 Expanding the Scraper

The scraper only saves the HTML body of the homepage of each company. Preferably, all text is needed from the web page for this thesis. To expand the scraper to retrieve all necessary text from each company's website, two sub-problems that need to be addressed are

- Developing a scraping algorithm that can effectively extract the identified text from an URL while also avoiding collecting irrelevant or duplicate information.

- To extract data from all sub-domains associated with a company's URL, an algorithm needs to be developed that can identify each sub-domain and scrape its contents.

Developing the Scraping Algorithm

To extract the text from a URL the HTML must first be requested. `fetch_html()` is responsible for this. The function takes a session object and a URL as input and returns the text of the HTML response received from the given URL. The reason for using a session is to make asynchronous requests, which can perform a lot when synchronous requests can cause a lot of waiting time. The function first tries to make an HTTP GET request to the URL using the given session object. If the request succeeds, the function returns the text of the HTML response. If the request fails due to a `UnicodeDecodeError` exception, which is when a request is done with a bad URL. The function tries to resolve the URL using `urllib.request` and makes a new HTTP GET request to the resolved URL. If this request succeeds, the function returns the text of the HTML response. If the function doesn't get a HTTP response it returns `None`.

`clean_HTML()` is the function responsible for parsing the HTML to natural text. The function takes an HTML response as input and extracts the natural text content from it. It uses the BeautifulSoup library to parse the HTML response and extract all the text inside.

To obtain the most accurate text, different techniques were evaluated. The first method involved extracting all text from a BeautifulSoup object using `get_text()`. Redundant text, such as text in website headers, was then filtered out based on tags. Text inside classes containing a "header" tag, which often included non-essential information such as "log in", "register", and links to other pages, was dropped. However, due to the lack of consensus on how to create an HTML file, it was discovered during the experiment that a "header" could also be a tag for the HTML body. This resulted in cases where all the text would be decomposed.

Another approach was to consider how natural text is typically stored. Based on this, we assumed that most text is stored in paragraphs (<p> tags). Therefore, we extracted all text inside <p> tags by calling `get_text()` on every result from `soup.find_all('p')`.

The function also filters the extracted text based on a blacklist of terms such as "online store", "log in", "sign in", and "register". These terms are removed from the extracted text.

Furthermore, the function uses a regular expression to remove special characters and numbers from the text. The regular expression pattern used is "[a-zA-Z]+", which matches all alphabetic characters and filters out any non-alphabetic characters.

Identify All Sub-Domains and Scrape

The asynchronous function `scrape_all_subpages()` which scrapes the main page of a company and all possible subdomains associated with it. The function takes two arguments - the URL of the main page of the company, and JSONText which is a `JSON` object used to save the scraped text.

The function first starts an asynchronous session using `aiohttp.ClientSession` to make multiple HTTP requests. It then fetches the HTML of the URL using the `fetch_html()` function, and cleans the HTML to extract the title and text using the `clean_HTML()` function. The extracted text is then saved in the JSONText object.

The function then uses `BeautifulSoup` to parse the HTML and find all links associated with the main page, meaning the first URL is in the sub-domain. It filters the links to remove any invalid strings using `is_valid_string()` and modifies the links to ensure they are absolute URLs. If no links are found, or the links are invalid, the function attempts to scrape the subdomains using `scrape_bad_htmls()` which uses `Selenium`[2] that is an open-source tool commonly used for automating web browsers. Selenium is used to handle pages where Javascript can be blocking. This is because sometimes Javascript loads content in e.g. `nav bars`. This takes a few seconds to execute and consequently blocks the HTTP request.

Once valid links are identified, the function makes multiple HTTP requests from `fetch_html()` to fetch the HTML of all subdomains using `asyncio.gather`. For each response, the title and text are extracted using the `clean_HTML()` and saved the text in the JSONText object based on the title of the page. If the page is an "About" page, the text is saved in the About key of the JSONText object. It is assumed that the page is an "About" page if the title variable contains the string "about" or the string "om oss" or if the title variable starts with the string "om".Every other text is saved to the "All" key which is a concatenation of all parsed text in every subdomain. If the title has not been seen before, it is added to the links key of the JSONText object. Finally, the scraped text is returned in the JSONText object.

[2] https://www.selenium.dev/

3.2 Automating Green Impact Identification Using Machine Learning

As discussed in subsection 2.5.1, Fauna has established a green rating system based on a set of criteria that green brands must fulfill to earn points. However, this process of evaluating each brand's green score is manually conducted by Fauna's experts, which can be time-consuming and resource-intensive. Therefore, this thesis aims to explore the possibility of automating the green scoring process using machine learning techniques, to provide a more efficient and accurate evaluation of the brands' sustainability performance.

3.2.1 Problem Definition

Each brand can fulfill multiple green criteria, from now on called labels. Since each brand can fulfill multiple labels simultaneously, this problem can be considered a multi-label classification problem.

All text from the website of each brand will be extracted using the custom web scraper. This text will be used together with the target labels to train a classification model that can predict if other companies have these labels or not.

3.2.2 Challenges

The labels for each brand's sustainability rating are manually checked, but the process by which they are determined and the specific source of the label fulfillment is not recorded or stored. This means that this information can be anywhere on the website of the company. Therefore all the text from the website must be checked. The generation of excess data leads to an increased volume of training data for the model, which in turn contributes to the complexity of the problem. With more text in the input, the task of predicting labels becomes increasingly difficult for the model. The dataset contains 360 brands, but a lot of the entries are empty. It is assumed that empty entries are companies without any web page. These will have to be removed from the dataset. Also, some companies use anti-scraping techniques to prevent automated scraping. Considering the difficulty of this problem, the dataset available is quite limited. This thesis will explore methods to expand the size of the dataset.

By nature, some labels will be more populated than others. This will cause a bias towards certain labels. In a multi-label classification problem, bias towards certain labels can have a significant impact on the performance of the model. When there is a lot of bias towards certain labels, the model may become over-reliant on those labels and may struggle

to accurately predict the other labels. This can result in an imbalanced distribution of predictions, with some labels being consistently over-predicted or underpredicted. Additionally, bias towards certain labels can lead to a lack of diversity in the predicted labels, potentially leading to missed opportunities for the model to make more nuanced and accurate predictions. To mitigate the effects of bias in multi-label classification, this thesis will consider the data and feature selection, as explore appropriate techniques such as oversampling or undersampling to balance the dataset.

With thirty labels in total, it would be a challenging task for any model to handle such a large number of labels. The dataset needs to be categorized and further models need to be trained accordingly. However, dividing the dataset into all eleven categories would result in an excessive number of models. Hence, it is crucial to find a balance between the number of models and the labels and explore alternative ways to achieve this.

3.2.3 Categorizing the Dataset

In Figure 3.1 the categories with belonging labels can be seen. After a quick look, it is obvious that it is possible to merge many of the categories based on shared labels. Clustering will be used to identify and group together categories with similar label compositions. By clustering the categories based on their labels, it becomes possible to identify which labels are common across multiple categories and which labels are unique to specific categories. K-means is a suitable clustering technique for this task. K-means is easy to implement and is designed to partition data into groups based on similarity. A binary vector with information on whether a category has a label or not is passed as training instances to the cluster. `sklearn.cluster.KMeans`[3] module is used to compute the clusters.

The most important parameter for this algorithm is the number of clusters to form. There are some techniques to determine this parameter. The most common is to choose a range of values. Iterate through all values K and calculate the silhouette coefficient for each point in the dataset, which measures how similar that point is to its cluster compared to other clusters. The value of K that maximizes the average silhouette coefficient is considered to be the optimal number of clusters. However, sometimes it is best to use prior knowledge or domain knowledge to decide. In this case, it can be seen in Figure 3.1 that there are some obvious groups. These are retail stores, food, and eco-lifestyle. Therefore it is decided to use 3 as the number of clusters.

Once the clusters have been determined, the selection of appropriate labels becomes necessary. Some labels are shared over the whole cluster, while others may not be so

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Criteria	Comment	Weight	Fashion	Food	Sports	Eateries	Tech	Energy	Home	Pets	Self Care	Family	Transport
Rental, main service	Rental as main service	20	x		x		x		x	x		x	x
Second hand, main service	Second hand sales as main service	20	x		x		x		x	x		x	x
Repair, main service	Repair as main service	20	x		x		x		x			x	x
Local Produce	produce	20		x		x							
Foodbox	Foodbox as their main concept	20		x									
Ecological	products	20		x		x							
service	main concept.	20		x									
No meat, main concept	concept	20		x		x							
Sustainable raw materials	environmentally friendly raw	20	x		x				x	x	x	x	
Do It Yourself	making or repairing your own	20	x		x		x		x			x	
Products for life	these companies put in	20	x		x		x		x	x		x	
Ghost kitchen	spare capacity	20				x							
Minimal waste	production, raw material	20	x		x		x		x	x	x	x	x
consumption	customers use more energy	20						x					
consumption	energy consumption	20						x					
Sustainable activities	sustainable use of nature and	20			x								
Public Transport	transportation	20											x
Sustainable transportation	sustainable transportation such	20											x
Sustainability reporting	Having an updated sustainability reporting including quantitative data for main metrics	1	x	x	x	x	x	x	x	x	x	x	x
CO2 emission reporting	Reporting CO2 emissions for scope 1, 2 and 3	3	x	x	x	x	x	x	x	x	x	x	x
Rental, additional service	Offering rental services as an addition/alternative to your traditional business model	5	x		x		x			x		x	
Second hand, additional service	Offering second hand as an addition/alternative to your traditional business model	5	x		x		x		x	x		x	
Delivery to doorstep	Delivering to customer as an alternative to customer all going to the store.	5		x									
Food waste reporting	Quantitative reporting of your food waste including goals and performance metrics	3		x		x							
Repair for free	Free repair of your post owned products	5	x		x		x		x	x		x	
Grønt Punkt	Member of Grønt Punkt	1	x	x	x	x	x		x	x	x	x	x
Miljøfyrtårn	Certified Miljøfyrtårn	1	x	x	x	x	x	x	x	x	x	x	x
Offer green energy	Offer renewable energy with origin guarantee	1						x					
energy	e.g. solar panels they will buy it	3						x					

Figure 3.1: Categories and labels in the dataset

important. This is also called feature selection. Feature selection is very important because too many features make it hard for the model to learn. Removing too many features may cause in loss of too much information. First, it was decided to only keep strictly shared labels. This means the labels that all categories in the cluster share are kept. Figure 3.2 shows all labels that were dropped using this method. As can be seen, some labels should probably be kept, such as 2,8,9 and 20. As a consequence, an algorithm was developed to measure weights for each label and keep the labels with weights above a certain threshold. The pseudo-code for this algorithm can be seen in Algorithm 3.1.

Algorithm 3.1 Find Common Criteria

Input: *df brands*, *list clusters*, *threshold*
  Initialize $shared\_criteria \leftarrow dict$
  Initialize $cluster\_dict \leftarrow dict$

  for *cluster in clusters* **do**
    for *category in cluster* **do**
      $cluster\_dict[category] \leftarrow cluster$
    end for
    $cluster\_df \leftarrow brands[cluster]$
    $nan\_percents \leftarrow \frac{\sum_{j=1}^{m} \mathbf{NaN}}{n\ columns},\ m\ =\ rows\ cluster\_df$
    $row\_scores \leftarrow (1 - nan\_percents) * df['weights']$
    $cluster\_df \leftarrow cluster\_df[row\_scores \geq threshold]$
  end for

  return *cluster_df*

	Fashion	Sports	Tech	Home	Pets	Family	weights
2	x	x	x	x	NaN	x	20
3	NaN	NaN	NaN	NaN	NaN	NaN	20
4	NaN	NaN	NaN	NaN	NaN	NaN	20
5	NaN	NaN	NaN	NaN	NaN	NaN	20
6	NaN	NaN	NaN	NaN	NaN	NaN	20
7	NaN	NaN	NaN	NaN	NaN	NaN	20
8	x	x	NaN	x	x	x	20
9	x	x	x	x	NaN	x	20
11	NaN	NaN	NaN	NaN	NaN	NaN	20
13	NaN	NaN	NaN	NaN	NaN	NaN	20
14	NaN	NaN	NaN	NaN	NaN	NaN	20
15	NaN	x	NaN	NaN	NaN	NaN	20
16	NaN	NaN	NaN	NaN	NaN	NaN	20
17	NaN	NaN	NaN	NaN	NaN	NaN	20
20	x	x	x	NaN	x	x	5
22	NaN	NaN	NaN	NaN	NaN	NaN	5
23	NaN	NaN	NaN	NaN	NaN	NaN	3
27	NaN	NaN	NaN	NaN	NaN	NaN	1
28	NaN	NaN	NaN	NaN	NaN	NaN	3

Figure 3.2: Dropped labels in the first method

3.2.4 Balance the Dataset

As discussed earlier, the distribution of labels can be uneven. In a multi-label classification task, an important task is to make sure the training and test data sets are representative of the overall population of labels. Let's say the label "Delivery to doorstep" is represented one time in the dataset and the train and test sets are divided randomly. If the label only appears in the test set, the model has not been trained on this label and can therefore not perform well on the new, unseen data. To address this issue, a technique called

stratification will be implemented. [12] has a `iterative_train_test_split` module utilized to ensure stratification in the train-test split. After testing this module it was discovered that when labels appeared one time the stratification did not work. This is because stratification is used to ensure labels are evenly distributed after the split. Consequently, it was determined to develop a function for oversampling.

In Pykes [13] describes oversampling as an approach where additional examples from the minority class are generated to balance the distribution of labels. There are different oversampling methods, such as random oversampling, which duplicates existing examples from the minority class, and synthetic oversampling, which generates new examples based on the existing examples. However, oversampling can also introduce some challenges. For example, it may increase the risk of overfitting, which occurs when the model is too closely fit to the training data and does not generalize well to new, unseen data. Oversampling can also introduce bias if the generated examples are too similar to the existing examples, or if the minority class is artificially over-represented in the training data. As discussed earlier, this model is very fragile to bias towards labels. Therefore, only labels that appear one time in the dataset are sampled an extra time to ensure that the stratification can work properly. `oversample_dataframe()` is implemented for this purpose. The pseudo-code for the function can be seen in Algorithm 3.2.

Algorithm 3.2 Undersample Dataframe

Input: *dataframe df*, *string label_column*
  label_counts ← *dict*
  rows_to_repeat ← *list*

  for *labels in df*[*label_column*] **do**
    for *label in labels* **do**
      if label in label_counts **then**
        label_counts[*label*] ← *value* + 1
      else
        label_counts[*label*] ← 1
      end if
    end for
  end for

  filter_labels ← *labels with* 1 *count*

  for *label in filter_labels* **do**
    rows_to_repeat ← *rows_to_repeat* + *df*[*df*[*label*] == *label*]
  end for

  combined_df ← *df* + *df*[*rows_to_repeat*]

  return *combined_df*

3.2.5 Summarizing Scraped Text

Scraping all the information from a corporate website can involve extracting a large amount of data, including text, images, videos, and other multimedia content. The text content alone can be substantial, especially if the website contains a lot of pages or blog posts. When dealing with long texts in the context of natural language processing, it is common to encounter a problem where the text is too long to be processed by neural networks or transformer models. These models typically have a fixed input size and can only handle a limited number of tokens, usually up to 512. As a result, it is necessary to preprocess the text and summarize it down to a fixed number of tokens.

Implementing the summarization of a long text using a neural network is a challenge and there is a big risk of a lot of details and information in the text can go missing. The goal of the summarization is to reduce the text to 512 tokens. How many words this is varies a lot, but a rough average, of 512 tokens is around 2500 words. First, each text is passed into a recursive function that summarizes the text if it's more than 2500 words.

The summarization method is implemented using a pre-trained BART (Bidirectional and Auto-Regressive Transformers) model, which is a type of transformer-based architecture that is commonly used for text generation tasks such as summarization [14]. To process the text, the code splits it into batches of tokens that correspond to the exact model maximum length (1024 tokens for BART), using a while loop. It generates a summary for each batch of tokens using the generate method of the BART model. Finally, the code decodes the summary outputs back into human-readable text format using the tokenizer's decode method and combines them into one string with one paragraph per summary batch using the join method. The resulting text is the summary of the input text that has been generated by the BART model.

One alternative approach that was considered was to use GPT-3 to summarize the texts. This approach is probably the better one as well. GPT-3 has the potential to capture more of the nuance and context of the original text while still providing a shorter version. However, summarizing very long texts with GPT-3 can cost a lot of money because it requires a lot of tokens and GPT-3 charges money per token.

3.2.6 Manually Checking the Texts

After summarizing the texts, it's easier to check the performance of the scraper as well as the summarizer. Therefore all the summed texts are checked manually. Simultaneously checking the texts, they are corrected using Grammarly. It is quickly discovered that the correctness of the texts is quite good, and the content is not. A lot of the texts contain

information about cookies, data, privacy, and JavaScript, despite efforts to avoid these topics.

Since this dataset is to be used for text classification, it was divided to delete all texts that only contain the topics above, as well as texts that don't mention anything about the environment at all. After the manual check, the dataset is reduced significantly from 188 samples to 23 samples.

3.2.7 A Multi-Label, Zero-Shot Learning Approach

Given the small size of this dataset, it is not appropriate to use it for training a machine learning model as there may not be enough data to generalize well. Hence, other techniques must be considered to extract meaningful information from the text. These techniques may include methods such as transfer learning, where a pre-trained model is fine-tuned on this dataset, or unsupervised learning techniques such as clustering and topic modeling. Yin et al. [15] suggested employing pre-trained Natural Language Inference (NLI) models as readily available zero-shot sequence classifiers. By utilizing this technique, it is possible to achieve high accuracy and efficiency without the need for extensive training on a specific dataset.

"Some pre-trained NLI models offer zero-shot learning capabilities, which can be easily accessed through the Huggingface zero-shot-classification pipeline. The models that support this feature can be found on this page[4]." These models support multi-label classification, which is the ability to assign multiple labels to a single instance. The models support this by setting `multi_label = True` in the pipeline. The multi-label, zero-shot learning algorithm can be seen in Algorithm 3.3.

3.3 CICERO Shaded of Green

The major challenge with the Fauna data is that all text or input has to be scraped from each company website. Developing the scraper is time-consuming since HTML's can be very different from each other and it is not guaranteed to end up with a dataset that will be suitable for training a machine learning model.

As introduced in subsection 2.5.2, the information about each company is stored in a PDF. The list of public reviews is not available to download so this dataset must also be scraped. However, most of the assessments stick to the same pattern so it's easier to scrape this dataset.

[4] https://huggingface.co/models?pipeline_tag=zero-shot-classification&sort=downloads

Algorithm 3.3 Zero-shot multi-label classification using BART-large-mnli model

Input: *Dataframe FaunaData*
f1_score ← *from sklearn.metrics*
model ← *Transformers.pipeline(model name)*
actual labels ← *dict*
predicted labels ← *dict*
candidate labels ← *dict*

for *row in FaunaData* **do**
cluster ← *row['cluster']*
text ← *row['text']*
actual labels ← *row['labels']*
candiate labels ← *candidate labels[cluster]*
predictions ← *model(text, candiate labels, multi_label = True)*
predicted labels[cluster] ← *predictions > 0.5*
acutal labels[cluster] ← *actual*
end for

for *cluster in clusters* **do**
F1 score ← *f1_score(predicted labels[cluster], acutal labels[cluster])*
end for

$$weighted\ F1\ score \leftarrow \frac{F1\ score}{number\ of\ clusters}$$

return *weighted F1 score*

This task aims to generate a dataset containing natural language text about the environmental aspects of companies, as well as their corresponding ratings. A machine learning model will be deployed to analyze this text data and forecast the rating of the companies. Since each company receives one of the multiple ratings, this is considered a multi-class classification problem. However, the list only possesses the green-rated assessments which is a drawback since the model will not learn about the companies without a focus on environmental issues.

3.3.1 Scraping the Dataset

The public list of reviews is called `CICERO Shades of Green - Second opinions`[5]. There are in total 363 assessments with 20 for each page. For each page, there is a link to each assessment and a button to the next page. Page layout can also be seen in Figure 3.3. The scraper starts at this page which has a base URL for all the pages. The blue button is blue until the last page, so the function is made, `crawl_page()` that calls each page recursively until the last page. The function is called with the base URL and the sub-link found by scraping the page and finding the 'href' tag to the class attribute

[5] https://pub.cicero.oslo.no/cicero-xmlui/handle/11250/2594071/browse?type=title

of 'next-page-link' which is the button. This 'href' contains the sub-link to the next page. This is found by parsing this page with `BeautifulSoup` and finding the class. In each function call, all assessments are scraped.

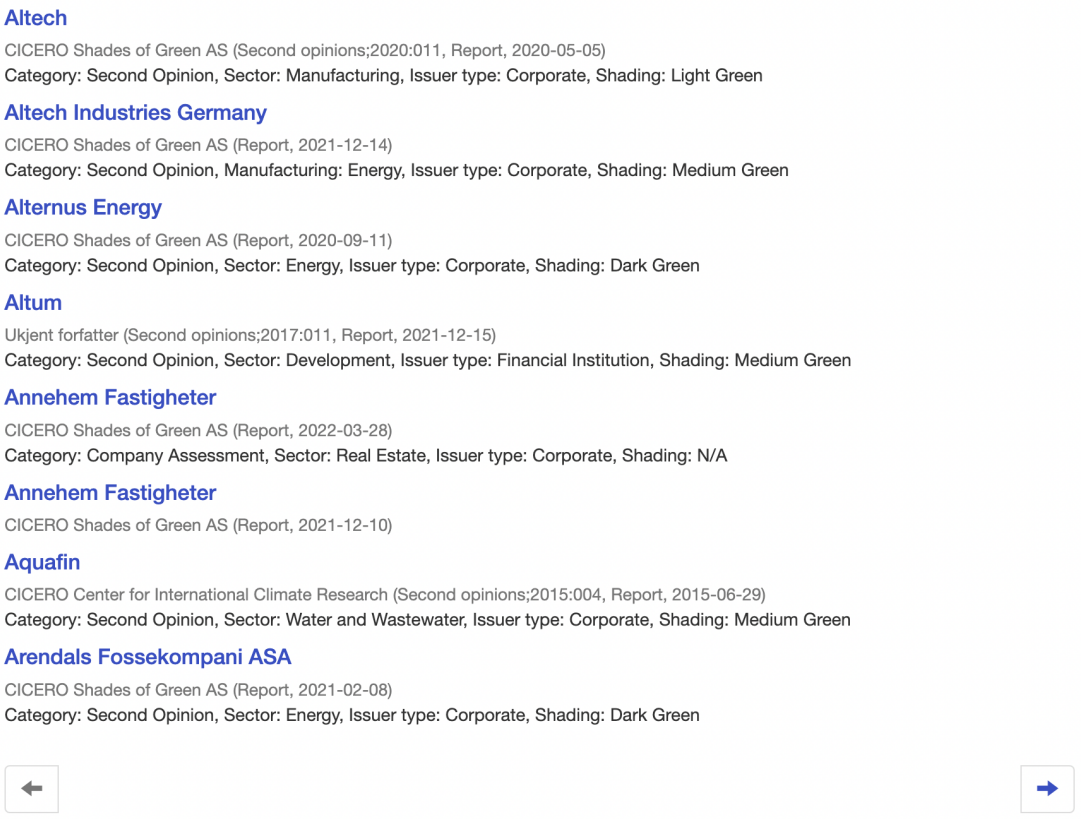

Figure 3.3: Page layout of the public list of review

To find all assessments the parsing of the page is used. Each page has an unordered list (ul) where the links to each assessment are stored. So all 'ul' tags are found on the page and then all list item (li) items with class name 'ds-artifact-item' are found. It's the list of items that contain the links to the assessments where the PDF is stored. The function iterates through all these list items and scrapes the page. An example of how each assessment page looks like can be seen in Figure 3.4. When the page is scraped the text under the header "Sammendrag" is parsed and saved. "Sammendrag" contains information about the rating and issuer type which will be relevant for this task.

The link to the PDF is found under the header "Åpne". To read and scrape the PDF, the Python library `PyPDF2`[6] is used. The content is requested using the same `request` library as before. The content is then passed into the PDF reader of `PyPDF2`. This outputs all the text in the PDF. Metadata and all text from the PDF are then stored in a dataframe.

[6] https://pypi.org/project/PyPDF2/

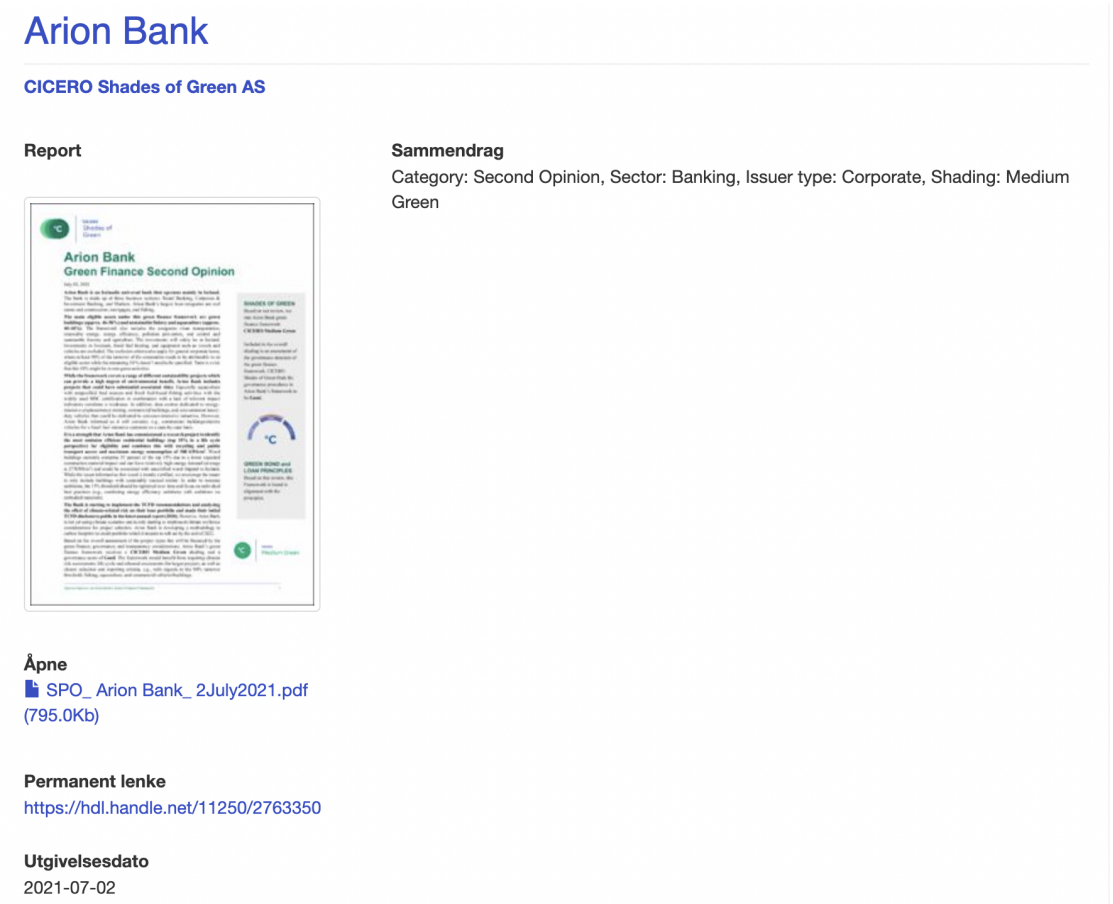

Figure 3.4: Page layout of each assessement

3.3.2 Extracting Relevant Text

The first step in the preprocessing of this model is to filter out relevant text from the PDFs. After a manual scan of the reports it was, it was discovered that most of the reports had the same structure. The reports that didn't will be discarded since they often did not contain any useful information. In Figure 3.5, the front page of the report can be seen. This is the text that will be filtered out and used for this task. In the figure, there are two red squares. The first square contains the date when the report was made, and the second contains the rating of the assessment. To extract the text between these two squares a Python regex will be used. This regex will find the text after any year, until any of the rating options. The available rating options can be represented as a single string that lists all the options separated by the 'or' operator. In this case, the string is written as `'Light Green|Medium Green|Dark Green'`. The regular expression is constructed using an f-string and includes the available shading options as a non-capturing group. The resulting string is as follows: `fr'\d4\s+(.*?)\s+(?:shading_options)'`. Additional preprocessing steps, such as removing any extraneous whitespace or newline characters, are performed on the input data before applying the regular expression. This ensures

that the input data is in a standardized format that can be correctly parsed by the regular expression. In the end, the last sentence is also removed because this is the sentence reasons why the company got its rating.

Arion Bank
Green Finance Second Opinion

July 02, 2021

Arion Bank is an Icelandic universal bank that operates mainly in Iceland. The bank is made up of three business sections: Retail Banking, Corporate & Investment Banking, and Markets. Arion Bank's largest loan categories are real estate and construction, mortgages, and fishing.

The main eligible assets under this green finance framework are green buildings (approx. 46-50%) and sustainable fishery and aquaculture (approx. 40-44%). The framework also includes the categories clean transportation, renewable energy, energy efficiency, pollution prevention, and control and sustainable forestry and agriculture. The investments will solely be in Iceland. Investments in livestock, fossil fuel heating, and equipment such as vessels and vehicles are excluded. The exclusion criteria also apply for general corporate loans, where at least 90% of the turnover of the corporation needs to be attributable to an eligible sector while the remaining 10 % doesn't need to be specified. There is a risk that this 10% might be in non-green activities.

While the framework covers a range of different sustainability projects which can provide a high degree of environmental benefit, Arion Bank includes projects that could have substantial associated risks. Especially aquaculture with unspecified feed sources and fossil fuel-based fishing activities with the widely used MSC certification in combination with a lack of relevant impact indicators constitute a weakness. In addition, data centres dedicated to energy-intensive cryptocurrency mining, commercial buildings, and zero-emission heavy-duty vehicles that could be dedicated to emission-intensive industries. However, Arion Bank informed us it will consider, e.g., commercial buildings/electric vehicles for a fossil fuel intensive customer on a case-by-case basis.

It is a strength that Arion Bank has commissioned a research project to identify the most emission efficient residential buildings (top 15% in a life cycle perspective) for eligibility and combines this with recycling and public transport access and maximum energy consumption of 300 kWh/m². Wood buildings currently comprise 35 percent of the top 15% due to a lower expected construction material impact and can have relatively high energy demand (average is 277kWh/m²) and could be associated with uncertified wood shipped to Iceland. While the issuer informed us that wood is mostly certified, we encourage the issuer to only include buildings with sustainably sourced timber. In order to increase ambitions, the 15% threshold should be tightened over time and focus on individual best practices (e.g., combining energy efficiency ambitions with ambitions on embodied materials).

The Bank is starting to implement the TCFD recommendations and analyzing the effect of climate-related risk on their loan portfolio and made their initial TCFD disclosures public in the latest annual report (2020). However, Arion Bank is not yet using climate scenarios and is only starting to implement climate resilience considerations for project selection. Arion Bank is developing a methodology to carbon footprint its credit portfolio which it intends to roll out by the end of 2022.

Based on the overall assessment of the project types that will be financed by the green finance, governance, and transparency considerations, Arion Bank's green finance framework receives a **CICERO Medium Green** shading and a governance score of **Good**. The framework would benefit from requiring climate risk assessments, life cycle and rebound assessments for larger projects, as well as clearer selection and reporting criteria, e.g., with regards to the 90% turnover threshold, fishing, aquaculture, and commercial vehicles/buildings.

SHADES OF GREEN

Based on our review, we rate Arion Bank green finance framework **CICERO Medium Green**

Included in the overall shading is an assessment of the governance structure of the green finance framework. CICERO Shades of Green finds the governance procedures in Arion Bank's framework to be **Good**.

GREEN BOND and LOAN PRINCIPLES

Based on this review, this Framework is found in alignment with the principles.

°C CICERO Medium Green

Figure 3.5: Front page of the reports

3.3.3 Balance the Dataset

To gain further insight into the population distribution of the dataset, a histogram was created to visualize the frequency of each class. This histogram can be viewed in Figure 3.6. As can be seen, there is a small representation of the light green class. To fix this, undersampling will be used as discussed in . First, the dataset is divided into one train and one train, and one validation dataset. Then, some of the light green samples are transferred from the train set to the validation set to balance out the validation dataset. This is because the validation dataset must contain original samples, while the training dataset can be balanced later using other techniques.

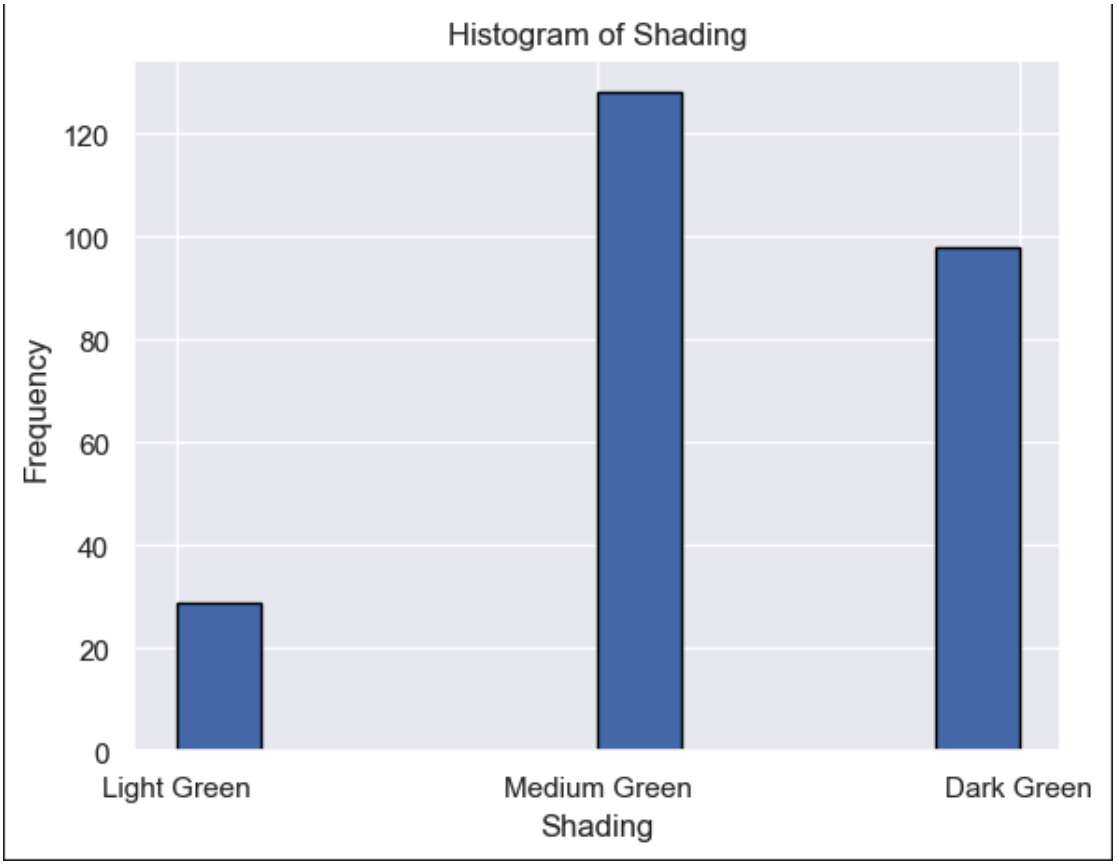

Figure 3.6: Population distribution of each rating (shading) in the CICERO dataset

3.3.4 Data Augmentation and Undersampling

Undersample the Traning Dataset

In subsection 3.3.3, the method of undersampling was implemented by duplicating samples from the classes that were only represented one time in the dataset. However, in the following section, an augmentation technique will be utilized instead. This section

will utilize the `OpenAi API`[7] to generate additional samples. In Algorithm 3.4, the algorithm for generating additional samples is shown. The `OpenAi API` is called with `text-davinci-002`[8] as engine. Next, all samples labeled as "Light Green" are iterated through. The model is given a prompt requesting it to generate text that is similar to the text associated with the given sample. The response from the model is then added to a new dataframe. At the end of the loop, the new dataframe is returned and concatenated with the training dataset.

Algorithm 3.4 Generate new samples

Input: $dataframe\ df, string\ api_key$
$new\_samples_df \leftarrow dataframe$
$model \leftarrow OpenAI(api\_key)$

for $text, label\ in\ df[label == '' LightGreen''] $ **do**
$prompt \leftarrow '' Create\ a\ similar\ text\ to\ text''$
$response \leftarrow model(prompt)$
$new\_row \leftarrow response, label$
$new\_samples\_df \leftarrow new\_samples\_df + new\_row$
end for

return $new\_samples_df$

Using Data Augmentation to Expand the Training Dataset

The thesis aims to explore the effectiveness of generating data using OpenAI text generation in improving the performance of models utilized on the dataset. The procedure depicted in Algorithm 3.4 is employed once again, but this time the entire training dataset is iterated through and re-sampled by generating new text. As a result, the size of the training dataset is doubled, and the augmented dataset is saved as `train_augmented`. After this, the same procedure is repeated with the `train_augmented` dataset to create a new dataset with even more augmented samples.

3.3.5 Correcting Texts

The scraping algorithm used `PyPDF2`[9] library to read the texts from the PDFs. One challenge with this library, and other PDF reading libraries is that the accuracy of the extracted text is often poor. One example is that words are divided, e.g. "word" can end up as "w" and "ord", "wor" and "d" or "wo" and "rd". There are many reasons why this occurs, but inconsistent formatting and embedded fonts are probably the most common

[7] https://platform.openai.com/docs/api-reference/introduction
[8] https://platform.openai.com/docs/models/model-endpoint-compatibility
[9] https://pypi.org/project/PyPDF2/

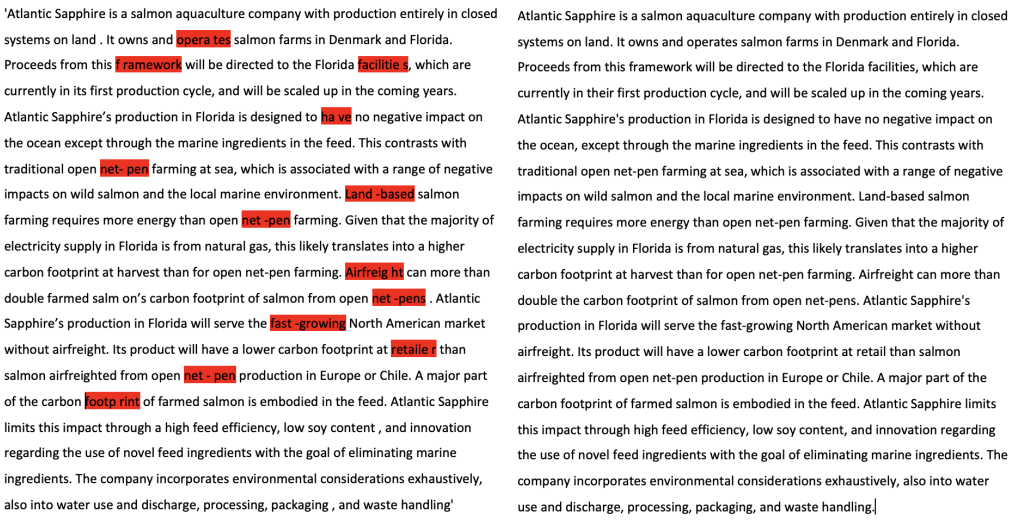

Figure 3.7: Scraped and corrected text utilizing OpenAI's text generation model.

culprits. One example of the scraped texts can be seen on the left side of Figure 3.7. In addition to the marked words, punctuation, and commas are almost always misaligned to the right.

Since NLP models can be sensitive to inaccurate language, the texts must be corrected. In Python, there are many ways of doing this. The Python library `Pyspellchecker` is a popular library for spell checking. However, when using this library each word is corrected by iterating through every word in the text. Since the biggest problem is divided words, this is not an effective approach. OpenAI's approach to correcting text is more effective because it is based on advanced machine learning algorithms that are capable of understanding natural language and identifying errors in the text. Therefore, a similar approach as in section 3.3.4 will be used. This time, each text in the dataset is iterated through. The prompt now will be "Correct this text", followed by the text. The prompt is fed into the model, and an example response is displayed on the right in Figure 3.7. The language in the corrected text is nearly flawless, indicating that this approach to text correction is highly effective.

3.3.6 Supervised Training of the Dataset

The model built for training this dataset mainly uses `PyTorch` and the `Hugging Face Transformers` libraries. The `scikit-learn` library is used to compute evaluation metrics and `Weights  Biases` is used for logging. The model is trained on a `Azure GPU` server. The GPU has a 15 GB memory.

Choosing Model

The transformer model called `Roberta-Large` from the Transformers library is implemented for training on this dataset. Roberta-large is a large-scale neural language model developed by Facebook AI Research (FAIR) in 2019. It is based on the transformer architecture and was pre-trained on a large corpus of text data, including both supervised and unsupervised learning tasks. Roberta-large has several characteristics that make it a good model for text classification on long texts. Roberta-large has 355 million parameters, making it one of the largest language models available. There are other bigger models, but since there is a memory cap of 15GB, this is one of the biggest models that can be used for training on the GPU. Roberta-large was trained on a diverse range of data sources and tasks, which helps it to generalize well to different types of text data. Roberta-large can also be fine-tuned on specific text classification tasks, allowing it to adapt to the specific characteristics of the target dataset. This is particularly useful for long texts such as the texts in this dataset, which often require more specialized models due to length and complexity.

Implementing the Model

The pre-trained `RobertaModel`[10] is imported from `Transformers`. The `RobertaModel` is followed by three fully connected layers: the pre-classifier, the classifier, and the activation function layer. The pre-classifier layer transforms the 1024 dimensional output from RobertaModel into a specified `hidden_layer_size`. The dropout layer is then applied to reduce overfitting with a dropout decided by the `dropout` parameter. The classifier layer performs the final classification, with the number of output classes. Finally, the `activation_func` parameter determines which activation function to use for the model's non-linearity. The forward method takes in input ids and attention mask parameters and returns the output of the classifier layer.

In this project, the values for hyperparameters such as `hidden_layer_size`, `dropout`, and `activation_func` are important factors that determine the performance of the model. These hyperparameters need to be optimized to obtain the best possible results. However, the details of how this optimization process is carried out will be explained later.

The `CustomDataset` class defines a PyTorch dataset for text classification tasks. The class takes as input a list of texts and a corresponding list of labels, and a `Roberta-large tokenizer` to use for tokenizing the texts. The `__getitem__` method tokenizes the text

[10] https://huggingface.co/roberta-large

and returns a dictionary containing the input_ids, attention_mask, and label tensors. The dataset made from this class is then used to create a `PyTorch DataLoader` object. The purpose of using a `Dataloader` is to facilitate the model training process by loading the data into the model in batches. Loading the entire dataset into memory at once may not be feasible due to memory limitations. When using `Dataloader`, each batch of data is loaded and preprocessed on-the-fly while the model trains on the previous batch. This helps to conserve memory and increase efficiency during training.

Since there is still an imbalance in the dataset, class weights are calculated before training the model. The class weights are calculated using Equation 3.1. The class weights are then passed into the `CrossEntropyLoss`[11] loss function. The loss function uses reduction='mean' as default, which means that loss will be normalized by the sum of the corresponding weights for each element.

$$Class\ weight = \frac{maximum\ count\ of\ any\ class}{number\ of\ samples\ in\ the\ class} \tag{3.1}$$

Fine-tuning the Model

To fine-tune the model, hyperparameter tuning is conducted. `Weights and Biases (Wandb)`[12] is used to logging. One of the features of `Wandb` is sweeps, which are used to automate the process of hyperparameter tuning. A sweep involves defining a configuration space for hyperparameters along with a search method, such as random, grid, or Bayesian. In the case of this model, the random search method is employed. This entails randomly selecting a configuration from the parameter range in a single sweep. This means that the "method" is random. Wandb will run multiple experiments with different combinations of hyperparameters in that space. The sweep results are then logged to Wandb, allowing for comparison of the performance of different hyperparameter configurations and selecting the best one for the model. The validation dataset is utilized to evaluate the model after every training epoch during the sweeps. Evaluation loss and accuracy are then logged for each epoch. After the sweep, the parameters from the run with the lowest evaluation loss are picked to train the model. The parameters used in the hyperparameter tuning sweep can be seen in Table 3.1.

Training the model

When training on the best parameters found from the sweep, the validation dataset is not used to evaluate during training. This is because when the model is tested at the

[11] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html
[12] https://wandb.ai/site

Hyperparameter	Values
Optimizer	adam, sgd
Activation Function	ReLU, sigmoid, tanh
Dropout	0.1, 0.2, 0.3
Epochs	50, 80, 100
Learning Rate	min=1e-08, max=1e-05
Batch Size	4
Hidden Layer Size	768, 512, 256
Weight Decay (L2 regularization)	min=1e-05, max=1e-03

Table 3.1: Hyperparameter Tuning Parameters

end, it should be evaluated on data that it has not seen before, and using the validation dataset for evaluation during training could potentially lead to overfitting the validation dataset. Therefore, the validation dataset is only used to select the best hyperparameters during the sweep and is not used for evaluation during the final training process.

Training Models with AutoTrain

AutoTrain[13] is a cloud-based platform for automated machine learning that enables users to easily train, test, and deploy machine learning models. The platform has pre-trained models and datasets available for users to fine-tune their datasets.

One of the main advantages of using AutoTrain is its ease of use. The platform provides a simple interface for users to upload their data and fine-tune pre-trained models. This eliminates the need for users to have expertise in machine learning and data science to achieve high accuracy. Additionally, AutoTrain provides various optimization techniques to improve the performance of the models, such as automatic hyperparameter tuning and ensembling.

To train a model in AutoTrain, a new project must be created. The user gives the project a name, selects the classification task, and selects a model. After the project is created, the user must upload a training dataset and a validation dataset. Then the platform trains multiple models and the best one can be selected based on the desired metric. The model can be uploaded with `Transformers.AutoModelForSequenceClassification(model-modelID)`.

Testing the Model

To test the model, the testing data and the trained model are passed into the training function. The `model.eval()` method is called to put the PyTorch model into evaluation

[13] https://huggingface.co/autotrain

mode before testing it on the test dataset. For each batch of data, the function computes the model's predictions using the `torch.argmax` method to get the index of the highest predicted score for each input. The true labels and predicted labels are stored in separate lists. After the loop, true and predicted labels are passed into `sklearn.metrics classification_report` which calculates precision, recall, and F1-score for each class, as well as an overall accuracy score. This report is then logged to Weights and Biases.

3.3.7 Transfer Learning on the Cicero Dataset

As discussed in section 2.3, transfer learning is a powerful technique in the context of deep learning, especially when the available training data is limited. Due to the limited availability of training data in the Cicero dataset, this thesis will include experiments on transfer learning techniques to improve classification performance. Zero, one and few-shot learning algorithms will be implemented using different models and compared to each other.

Certain models used in this study have a maximum sequence length of 512, which is roughly the length of each text instance in the Cicero dataset. These models are only suitable for zero-shot learning, which means they can classify texts without being trained on them. However, some other models can handle up to 2048 tokens, enabling them to perform one-shot learning, which involves learning from three example texts (one for each class) and then using a single text for classification. `GPT-3` and `GPT-3.5` can handle approximately 4000 tokens which are enough for two-shot learning. Six texts, two for each class are used for learning and one for classification.

Zero-Shot Implementation

How to implement zero-shot learning varies from model to model. The `bart-large-mnli` model explained in subsection 3.2.7 is compatible with zero-shot learning which makes it easy. Some models must be implemented using the model and a tokenizer. `Flan-T5`[14] is such a model. First, the tokenizer processes the zero-shot prompt to create input IDs. The input IDs are numerical values used to represent the words in the prompt. Each ID is mapped to a token, where a token is a sequence of characters that represents a single, meaningful unit of text. Then, the model takes these input IDs as input and produces an output which is one of the classes. Finally, the tokenizer decodes the output generated by the model. When using zero-shot on the GPT models, the API for the models is called with the prompt and the response is the prediction of the model.

[14] https://huggingface.co/google/flan-t5-xl

Common for all the zero-shot models is how the testing is conducted. To begin, since there is no training involved, the testing and validation datasets are combined. Next, the code iterates through each sample and extracts the text and true label. A prompt is created based on the text by constructing a string that reads: "Classify as Light Green, Medium Green, or Dark Green: [text]". The prompt is passed to the function responsible for predicting each model. The actual class and the predicted class are then saved. In the end, all actual and predicted values are saved in a text file and the F1 score is calculated.

One and Two-shot Implementation

The implementation for One ant Two-shot learning is similar since only the API models have a long enough sequence length for the prompt to fit. The only difference between One and Two-shot is that in one shot there is one example per class and in two-shot there is two. This is shown in subsection 2.3.2.

The models are tested similarly to section 3.3.7. The difference is that in each iteration, example texts are extracted to generate the prompt. For one-shot, one random text for each class is selected from the dataframe using `sample`[15]. The only condition is that none of the randomly selected texts are the same as the text that is being tested. For two-shots, two random texts for each class are selected. The algorithm is shown in 3.5.

Algorithm 3.5 One-shot implementation

Input: $dataframe df, string\ api\_key$
$model \leftarrow OpenAI(api\_key)$
$actual\ labels \leftarrow list$
$predicted\ labels \leftarrow list$

for $text, actual\ label\ in\ df$ **do**
$text1,\ text2,\ text3 \leftarrow df.isin([Light, Medium\ Green, Dark\ Green]).sample(3)[''text'']$

$prompt \leftarrow''\ Classify as Light Green, Medium Green or Dark Green : text1, Class : Light Green text2, Class : Medium Green text3, Class : Dark Green text, Class :''$
$prediction \leftarrow model(prompt)$
$actual\ labels \leftarrow actual\ labelslabel$
$predicted\ labels \leftarrow predicted\ labels\ +\ prediction$
end for

$F1\ score \leftarrow sklearn.metrics(actual\ labels, predicted\ labels)$

return $F1\ score$

[15] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html

Chapter 4

Experimental Evaluation

4.1 Experimental Setup

4.1.1 Models

All of the neural networks used in this thesis are large models, designed to handle complex tasks in natural language processing. Some of these models, such as GPT-3, are among the largest and most powerful models available today, while others are smaller and more specialized for specific tasks. A specification of key information of each model used in this thesis can be seen in subsection 4.1.1.

Model Specification			
Model	**Company**	**Size**	**Max Sequence Length**
Flan-T5-XL	Google	3 billion parameters	512 tokens
GPT-Neo 2.7B	EleutherAI	2.7 billion parameters	2048 tokens
GPT-3	OpenAI	175 billion parameters	4000 tokens
GPT-3.5	OpenAI	175 billion parameters	4000 tokens
bart-large-mnli	Facebook	406 million parameters	1024 tokens
mDeBERTa-v3-base-mnli-xnli roberta-large	Microsoft Research Asia Facebook	432 million parameters 355 million parameters	512 tokens 512 tokens
distilroberta-base-climate-f	Facebook	66 million parameters	512 tokens

4.1.2 Trained Models

As described in Section 3.3.6, certain trained models underwent hyperparameter tuning. This involved conducting 20 runs where random hyperparameters were selected, and the training process was evaluated. From each fine-tuned model, the configuration from the run with the lowest evaluation loss was chosen. Additionally, the same models were trained using `AutoTrain`, a method that automatically selects the optimal parameters. A comparison was then made between these approaches, and the model yielding the highest F1-score was selected as the best.

4.1.3 Metrics

In this thesis, the Weighted F1 score is used as an evaluation metric for comparing the performance of different models on multi-class classification problems.

In multi-class classification problems, the Weighted F1 score is a suitable evaluation metric because it takes into account the class imbalance that may exist in the dataset [16]. The equation for the F1 score can be seen in Equation 4.1. Weighted F1 score

takes the mean of the F1 score of each class concerning the support for each class. The support for each class is the number of samples. This gives a more accurate evaluation of the model's performance across all classes, rather than just the most frequent ones.

$$F1\ Score = \frac{True\ Positives}{True\ Positives\ +\ \frac{1}{2}(False\ Positives\ +\ True\ Negatives)} \tag{4.1}$$

Given that all the problems tackled in this thesis are multi-class classification tasks, the use of the Weighted F1 score provides a consistent evaluation metric that allows for a fair comparison of the performance of different models. This is important because it allows for objective evaluation of the effectiveness of the proposed solutions and helps to identify the best-performing models for each specific problem.

The F1 score, which is the harmonic mean of precision and recall, provides a combined measure of both metrics. However, it can be beneficial to analyze precision and recall individually, especially when they diverge from each other.

Precision represents the proportion of true positive predictions among all positive predictions, indicating the model's ability to minimize false positives. A high precision value suggests a low number of false positives, meaning the model is accurately identifying positive instances.

On the other hand, recall represents the proportion of true positive predictions among all actual positive instances, indicating the model's ability to minimize false negatives. A high recall value implies a low number of false negatives, signifying that the model effectively captures most positive instances.

Examining precision and recall separately can provide insights into specific aspects of the model's performance. If precision is high while recall is low, it suggests that the model is cautious in making positive predictions and tends to miss some true positive instances. Conversely, if the recall is high while precision is low, it indicates that the model is capturing many positive instances but may also produce a higher number of false positives [17]. In some models, it can be useful to focus on one of these metrics, depending on the use case.

4.1.4 Datasets

Fauna Dataset

As explained in subsection 3.2.6, the Fauna dataset is reduced to 21 samples. This small sample size reduces the statistical power of the testing. Therefore, no certain conclusions

will be made by the results of the models used to test this dataset. However, the results can give some indications of the model's performance. As explained, because the dataset contains so many classes, it's split into three clusters. Some statistics on the dataset can be seen in Table 4.1. The shared labels in each cluster that the model will predict can be seen in Table 4.2.

Fauna Dataset		
Cluster	**Num Labels**	**Num samples**
1	6	12
2	6	4
3	4	5

Table 4.1: Key Statistics on the Fauna Dataset

Cluster	**Labels**
Cluster 1	Sustainable raw materials, Second hand, main service, Sustainability reporting, CO2 emission reporting, Repair for free
Cluster 2	Ghost kitchen, Foodbox, Sustainability reporting, CO2 emission reporting, Food waste reporting, Miljøfyrtårn
Cluster 3	Sustainable raw materials, Sustainability reporting, CO2 emission reporting, Miljøfyrtårn

Table 4.2: Shared labels in each cluster

The clusters in this study are formed based on the labels provided by Fauna, which are displayed in Figure 3.1. The resulting clusters are visualized in Table 4.3,. Generally, the clusters demonstrate a meaningful grouping of companies. However, it is observed that the self-care category appears somewhat randomly combined with energy and transport within the clusters.

Fauna Clusters					
Cluster 1	fashion	tech	home	family	sports
Cluster 2	food	eateries			
Cluster 3	self-care	energy	transport		

Table 4.3: Fauna clusters grouped by labels

Cicero Dataset

The Cicero dataset is a bit bigger. For the fine-tuned models, the validation data contains 95 samples. Regarding the statistical power of the results, this is considered a small amount. But 95 samples can be a good starting point for testing a machine learning model. For the transfer-learning models, the test data contains 248 samples. This is the concatenation of the normal training data and the validation data. With this amount, it's more likely to obtain a representative sample of the population and reduce the impact of random variability on the results. For statistics on the dataset, see Table 4.4.

Cicero Dataset				
Dataset	**Num samples**	**Light Green**	**Medium Green**	**Dark Green**
Train Normal	153	38	58	56
Train Augmented	300	75	116	109
Validation	95	15	39	41

Table 4.4: Key Statistics on the Cicero Dataset

4.2 Experimental Results

4.2.1 Multi-Label, Zero-Shot Learning on Fauna Dataset

The zero-shot experiment was tested on two models. As explained in subsection 3.2.7, certain models are compatible with this experiment. `bart-large-mnli` was tested because it's the most popular and best-rated model. The model is a `Multi-Genre Natural Language Inference (mnli)`. `mDeBERTa-v3-base-mnli-xnli` is also tested because it was the second best-rated mnli model. This model is also a `Cross-lingual Natural Language Inference (xnli)` model, which means that it's trained across multiple languages. The dataset consists of Norwegian and English language so this can be of importance. The results can be seen in section 4.3.

Model ID	Metrics											
	Cluster 1			Cluster 2			Cluster 3			Weighted avg.		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
	Fine-Tuned											
bart-large-mnli	0.53	0.7	0.51	0.63	0.22	0.2	0.47	0.57	0.42	0.54	0.50	0.38
mDeBERTa-v3-base-mnli-xnli	0.3	0.3	0.27	0.67	0.33	0.37	0.42	0.57	0.48	0.46	0.40	0.37

Table 4.5: Metrics for Multi-Label Zero-Shot Learning on Fauna Data

As discussed in section 4.1.4, the experiment's limited sample size prevents any conclusion. However, the results can indicate a few things. First, predicting classes across different languages is feasible. All of the labels in the dataset are English and the texts is a mix of English and Norwegian. This suggests that the `Transformer` models possess the capability to generalize and understand patterns in multilingual data. The best model had an overall F1 score of 0.38, suggesting that there is room for improvement. The models performed better on cluster 1 and cluster 3, where the labels are mostly 'Sustainable raw materials', 'Sustainability reporting', and 'CO2 emission reporting'. These labels are probably easier to map than food labels in cluster 2 that are more specific like 'Foodbox' and "No meat, main concept". Therefore, achieving an effective multi-label classification model is partially fulfilled.

In this thesis, two objectives are addressing the dataset. Collecting the data with a scraper, and prepossessing it so it's suitable for machine learning models. After the dataset is reduced to 21 samples, these objectives have to be classified as partly met, and the reason is a mix of both. The first obstacle is when extracting data from websites there many of whom are blocked from scraping. This should be respected. Respecting these limitations is essential to maintain ethical practices and comply with legal requirements. The second significant issue encountered was the presence of extensive textual data extracted, which consisted of a substantial amount of random information. When summarizing these texts down to a suitable size, it turned out it was difficult to filter out what information is important to keep and not. Despite the implementation of filters specifically designed to block random information, such as ”cookies” and ”personal data,” there were instances where this irrelevant data managed to bypass the filters.

Despite the drawbacks associated with this study, such as limited sample size and challenges in data extraction, valuable insights and findings were still obtained. This study has shown that it's possible to classify corporate websites in green labels with a pre-trained model and a zero-shot technique to a certain degree. What this means practically is that if this method can be improved, particularly in the data extraction step, are significant. Automating the process of rating these companies could result in substantial time and cost savings. By overcoming the challenges associated with data extraction and refining the methodology, the rating process could be streamlined and made more efficient.

In this thesis, other preprocessing steps were also implemented, such as oversampling. This was because the plan was to use implement a model that could be trained on the dataset as well. When the dataset become so limited, this wasn't possible anymore. However, these are important steps that can be used in the future if a method for increasing the size of the dataset can be found.

4.3 Multi-Class Classification on Cicero Dataset

Results from all experiments on the Cicero data can be seen in Table 4.6. As can be seen fine-tuned models are tested on different dataset, while the transfer-learning models are tested on a concatination of the normal traning and validation data .

4.3.1 Trained Models on Cicero Data

This experiment involved two models that were trained using both fine-tuning and auto-training techniques. `RoBERTa-large` is used because it's a big model pre-trained on

Model ID	Metrics											
	Light Green			Medium Green			Dark Green			Weighted avg.		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
Fine-Tuned												
Roberta-Large, Augmented	0.24	0.27	0.25	0.58	0.54	0.56	0.79	0.80	0.80	0.62	0.61	0.61
Roberta-Large, Normal	0.62	0.53	0.57	0.63	0.79	0.71	0.88	0.71	0.78	0.74	0.72	0.72
distilroberta-base-climate-f, Normal	0.5	0.2	0.26	0.62	0.74	0.67	0.76	0.78	0.77	0.66	0.67	0.66
Zero-Shot												
Bart-large-mnli	0.029	0.45	0.35	0.41	0.67	0.51	0.75	0.03	0.06	0.52	0.37	0.3
Flan-T5-xl	0.24	0.43	0.31	0.41	0.18	0.25	0.46	0.54	0.50	0.40	0.37	0.36
GPT-neo-2.7B	0.0	0.0	0.0	0.40	0.96	0.57	0.56	0.09	0.16	0.38	0.41	0.29
GPT-3	0.33	0.02	0.04	0.41	0.61	0.49	0.46	0.47	0.47	0.41	0.43	0.38
GPT-3.5	0.28	0.25	0.26	0.40	0.52	0.45	0.55	0.42	0.48	0.43	0.42	0.42
One-Shot												
GPT-neo-2.7B	0.0	0.0	0.0	0.4	1.0	0.57	1.0	0.02	0.04	0.55	0.41	0.24
GPT-3	0.24	0.21	0.22	0.44	0.73	0.55	0.57	0.24	0.34	0.45	0.43	0.40
GPT-3.5	0.41	0.42	0.41	0.37	0.48	0.42	0.62	0.43	0.51	0.48	0.45	0.45
Two-Shot												
GPT-3	0.24	0.17	0.20	0.42	0.70	0.53	0.57	0.29	0.38	0.44	0.43	0.40
GPT-3.5	0.34	0.65	0.45	0.46	0.33	0.39	0.67	0.52	0.58	0.52	0.48	0.48

Table 4.6: Metrics for Fine-Tuned and Trained Models

the massive amount of text data. On the other hand, `distilroberta-base- climate-f` is used because the model is additionally pre-trained on climate-related research text [18]. `Distilroberta` is a smaller and faster version of `Roberta`, so it's interesting to see how the size of `Roberta` compares to a model that's fine-tuned for climate-related research text.

The best model is the `Roberta-large` trained with `AutoTrain`, on the normal dataset. First of all, an F1-score of 72% can be considered quite good. This is because all of the texts that are classified are quite long and close to the max sequence length of the models. Usually, these models perform significantly better on shorter texts. Secondly, for both the models, `AutoTrain` performed better than the fine-tuned models, implying that `AutoTrain` is one of the most powerful training methodologies available. Another notable finding is that the larger model, `Roberta-large`, outperformed the smaller model, `Climatebert`, which is specifically trained on climate-related texts. The last observation is that the model trained on augmented data performed worse compared

to the original data. This suggests that the augmentation techniques employed did not effectively enhance the model's ability to generalize and make accurate predictions.

4.3.2 Transfer Learning Models (Zero-Shot, One-Shot, and Few-Shot) on the Cicero Dataset

In this experiment, five models are evaluated. Specifically, `bart-large-mnli` was pre-trained for this task and will be compared with the largest models available. The other models range from 2.7 billion parameters up to 175 billion, which are the biggest models available today. Due to limitations in the maximum sequence length of certain models, not all experiments are performed on every model. Some models have a maximum sequence length that is not sufficient to accommodate the required input size.

As explained in section 2.3, Brown et al. [10] showed that extremely large language models can perform competitively on few-shot, classification tasks with less data than is required from other models. In this study, it was observed that the best model, a two-shot prompt with GPT-3.5, exhibited an F1 score that was 24 percent lower than that of the best-fine-tuned model. This implies a certain degree of competitiveness. The study also showed that n-shot learning models have a parallel relationship, where the model with the highest value of n consistently outperformed the models with lower values of n. This is also the case in this study, at least for GPT-3 and GPT-3.5. This implies that these models can accurately classify a given text into a green class, with just a few examples as training data.

Zero-Shot Models

Figure 4.6 shows the predictions from each model from the zero-shot prompt. As can be seen, there is a big mix of how the models are biased. `gpt-neo-2.7B` predicts `Medium Green` almost all the time. This bias indicates that the model's predictions may not be reliable or trustworthy, as it appears to favor a specific class without considering the diversity of inputs or providing accurate representations of other potential outcomes. `GPT-3` and `bart-large-mnli` looks to have a little less bias, while `GPT-3.5` and `flan-T5-xl` look like the best model from solely looking at the distribution.

One-Shot Models

Figure 4.10 shows the distribution from the models tested on the one-shot prompt. As can be seen once again, `GPT-Neo` is biased in its predictions, indicating that the model

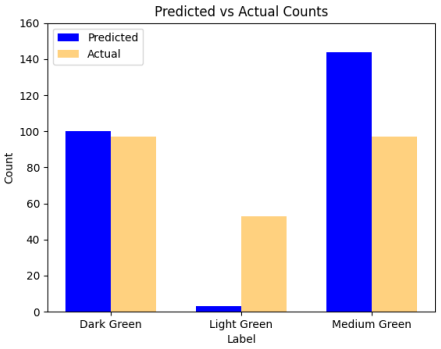

Figure 4.1: GPT-3

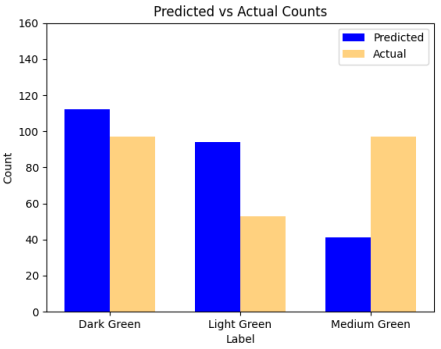

Figure 4.2: GPT-3.5

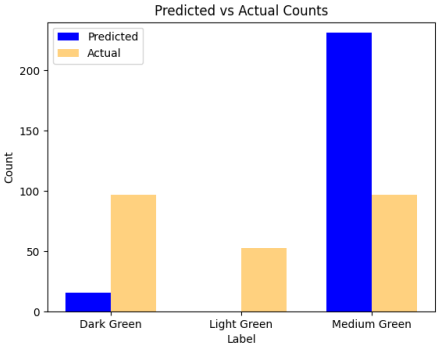

Figure 4.3: gpt-neo-2.7B

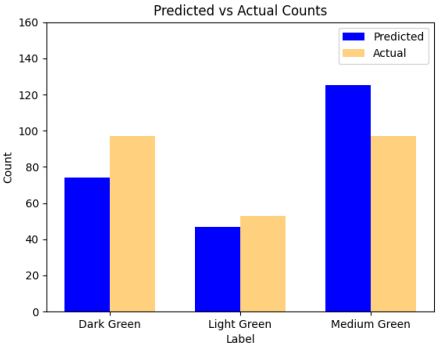

Figure 4.4: flan-T5-xl

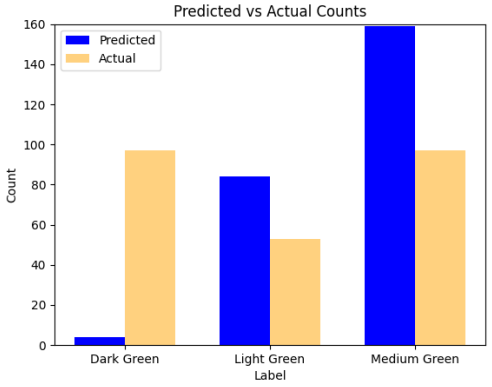

Figure 4.5: bart-large-mnli

Figure 4.6: Distribution of Predicted vs Actual for Zero-Shot Learning

is not suitable for transfer learning. This time, `GPT-3` looks a bit better, now that it's managing to predict `Light Green` sometimes. Once again, `GPT-3.5` stands out as the preferred choice based on the distribution, which is now even more balanced across the options. Two times, the output from this model was "N/A (not related to green bonds)", which indicates that the model makes predictions based on actual information present in the texts it analyzes.

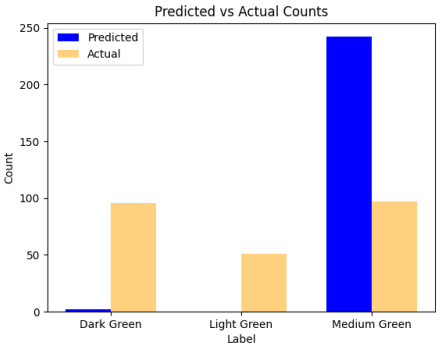

Figure 4.7: GPT-Neo

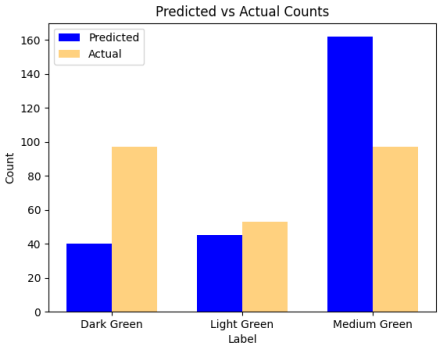

Figure 4.8: GPT-3

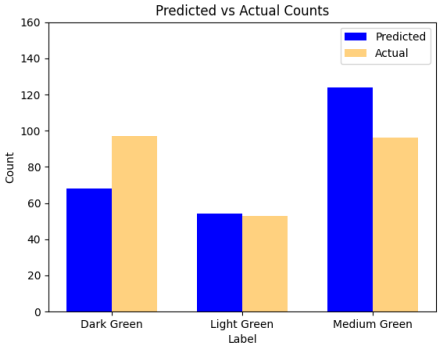

Figure 4.9: GPT-3.5

Figure 4.10: Distribution of Predicted vs Actual for One-Shot Learning

Two-Shot Models

Figure 4.13 shows the distribution from the models tested on the two-shot prompt. The distribution looks similar to the one-shot prompt, but the results are a few percent better. This model also generated, ”N/A (not related to green bonds)” which means that it is not completely random when the model does not predict.

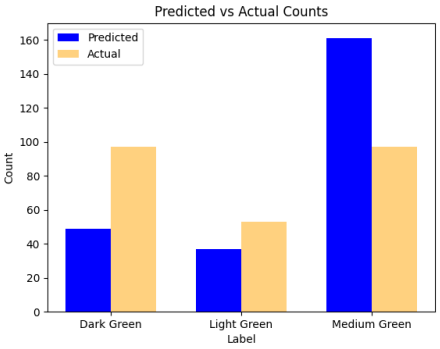

Figure 4.11: GPT-3

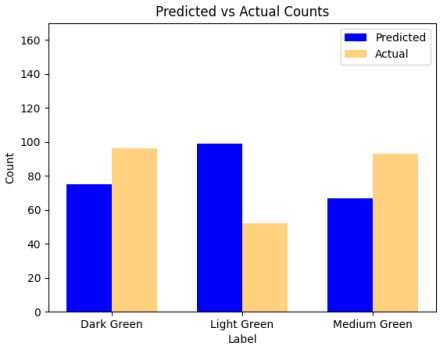

Figure 4.12: GPT-35

Figure 4.13: Distribution of Predicted vs Actual for Few-Shot Learning

4.3.3 Overall Implications of Results on Cicero Data

Trained models classified with an F1-score of 72%. This can be considered quite good considering all texts are long and approximately the max sequence length for the models. The best transfer-learning models, the two-shot models, performed significantly worse than the trained models, suggesting that training large models with limited training data remains an effective approach.

The fact that recall and precision are the same across fine-tuned models suggests that the models are performing consistently well across all classes. This indicates that the fine-tuning process has effectively adapted the pre-trained model to the specific task and data.

On the other hand, the variation in recall and precision observed in the transfer learning models implies that the performance differs across different classes. This can have a few implications. Firstly, in the fine-tuned models, it was possible to calculate class weights, and this was not possible in the transfer-learning models. Secondly, the transfer learning models may struggle with classes that have different data distributions compared to the pre-training data. In such cases, the models may not generalize as well to these specific classes, resulting in lower recall and precision. The model may also struggle at recognizing certain patterns or features that are highly indicative for some classes, and can struggle to find patterns in the other classes.

The practical issue with these results is that this thesis aimed to find an automatic approach to finding green claims or classifying companies as green/not green. The Cicero data has origin from reports that are manually generated by a research company. However, it shows that if it's possible to make companies share reports on a common communication channel or such, it is possible to automatically classify them.

Chapter 5

Conclusions

5.1 Summary

This thesis has developed web scrapers to scrape two types of datasets. The first scraper was implemented to scrape all information from the website of each company in the Fauna dataset. This was done by recursively scraping each sub-page extracted from the home page of every website. To extract data from the Cicero dataset, a second scraper was developed. The dataset consisted of entries listed as links within an unordered HTML list. By visiting each link, the scraper successfully extracted the report text contained within them.

Both datasets were then prepossessed to ensure compatibility with machine-learning models. A significant portion of the text scraped from the Fauna dataset was excessively long, primarily due to the abundance of information available on the websites. These texts were summarized with a Transformer model to ensure they were shorter than 512 tokens, which is the max sequence length for most of the Transformer models capable of text classification. Subsequently, each text within the dataset underwent manual correction, and those texts that were deemed irrelevant to sustainability were discarded from further analysis. Texts in the Cicero dataset were augmented to research if expanding the dataset could enhance the performance of the machine learning models. This was done by asking ChatGPT to write a similar text to each text in the dataset.

For the Fauna dataset, a zero-shot learning model was implemented for a multi-label classification task. The task was to classify each scraped text into the "green criteria" Fauna had already classified the company too. The zero-shot learning technique was used due to the small size of the dataset. Both fine-tuned models and transfer-learning models such as zero, one, and two-shot models were implemented to classify the Cicero dataset

into light green, medium green, or dark green. Both of the multi-class classification tasks were implemented on multiple models to research what kind of models are most likely to effectively accomplish the classification task.

5.2 Objectives

The first research question, RQ1, asked what datasets currently exist for greenwashing detection and what is needed. In this thesis, two datasets have served as benchmarks for collecting comprehensive information on companies. Both these datasets have their limits. The Cicero dataset categorizes the extent of ”greenness”. The results from experiments show that large language models manages to distinct less sustainable companies from the most sustainable companies. This doesn't help directly when detecting greenwashing, since no green claims can be found from this. The information are stored in reports that are manually generated, which removes the aspect of automatically finding these claims. The second dataset, can potentially be more efficient to detecting greenwashing since a model can be trained to distinct green labels of corporate products. From the experiments on this dataset, it was discovered that when a company website is manually labeled, it is very hard to train on these labels because a website can contain very much information. So if this dataset is too used to detect greenwashing, a better scraper must be implemented, as well as better methods of filtering key information from the websites.

After working on this thesis, there is no doubt to what is needed to detect greenwashing regarding datasets. A good method must be implemented to annotate green claims in texts that are published by companies. By doing this, more specific training data can be made, where green claims are actually connected to some information about the company or the products of the company.

The second research question, RQ2, asks if greenwashing detection be modelled as a multi-class and multi-label classification task. Based on the results in this thesis, the answer is no. RQ2 was addressed by employing Transformer models and leveraging the latest AI models for both multi-class and multi-label classification tasks. The study incorporated multiple models and techniques to explore the effectiveness of each approach in achieving the research objectives. The first classification task, the multi-label classification on the Fauna dataset, was tested on insufficient data. The other classification task showed that the large AI models are good enough to classify texts on very small amount of training data. Therefore, if a technique for annotating superior data can be employed, utilizing these large AI models as a multi-class or multi-label approach can be utilized to identify greenwashing.

During the analysis phase, RQ3 was addressed to determine the most suitable models with the best performance. The study incorporated multiple models and techniques to explore the effectiveness of each approach in achieving the research objectives. By using the power of Transformer models and utilizing the newest and largest AI models available, the study aimed to enhance the classification accuracy and performance across both multi-class and multi-label classification scenarios. Various models and techniques were implemented and evaluated to compare their effectiveness and determine which approaches yielded the most favorable outcomes.

As discussed, the multi-label models were tested on a limited dataset, which made it difficult to conclude the models. From the models tested on the Cicero dataset, the best-trained model performed with an F1 score of 72% which is considered a promising result in this context.

For the Fauna dataset, a zero-shot learning model was implemented for a multi-label classification task. The task was to classify each scraped text into the "green criteria" Fauna had already classified the company too. The zero-shot learning technique was used due to the small size of the dataset. Both fine-tuned models and transfer-learning models such as zero, one, and two-shot models were implemented to classify the Cicero dataset into light green, medium green, or dark green. Both of the multi-class classification tasks were implemented on multiple models to research what kind of models are most likely to effectively accomplish the classification task.

5.3 Future Directions

The application of machine learning for corporate greenwashing detection is still in its early stages, and there is significant potential for further research in this area. There is some potential for future research in the detection of greenwashing using machine learning models and techniques, alongside strategies to counter potential attacks on the algorithm. One crucial aspect that requires attention to bridge the existing gaps in this thesis is the enhancement of the web scraper. Considering that web pages are likely to contain a significant number of green claims, this particular area demands extensive effort and dedication. Additionally, there is a need to conduct further investigation into structuring and summarizing the vast amount of information obtained from web scraping, ensuring that it can be effectively utilized by ML models without compromising any relevant details about the environment.

Additionally, research can be done to create better-labeled datasets to improve the relevance of the classification. Classifying companies to a degree of green practices, like

in the Cicero data, is not too relevant when fighting greenwashing. Creating labels that are more specific to green claims is going to be important when detecting greenwashing.

Investigating real-time monitoring mechanisms can be valuable in identifying greenwashing practices as they occur. Developing algorithms that continuously analyze and evaluate corporate communications and actions can provide timely insights and assist in detecting misleading environmental claims.

Overall, continued research in this area is critical to prevent companies from deceiving consumers and contributing to environmental degradation.

Appendix A

Sample Appendix Contents

A.1 Code

All code implemented for this thesis can be found at https://github.com/factiverse/acdc/tree/feature/python_rework.

- Path to the script that scrapes the Fauna dataset is:

  `acdc-data-master/src/signals/homepage.acquire.py`

- Path to the script that scrapes the Cicero dataset is:

  `acdc-data-master/Cicero/scrape_cicero.py`

- All prepossessing scripts, ML-models and datasets for the Fauna dataset can be found in `acdc-data-master/Fauna`
  - `Fauna.xslx` is used as dataset for zero-shot experiment.
  - `Fauna_summed.xslx` is dataset prior to manually correcting the dataset.
  - zero-shot experiments are scripts with prefix: zero_shot_multi_label
  - `fetch_data.py` is to make a dataframe of the data from the database, as well as summarizing the texts.
  - scripts like `data_augment.py` and `impact_classifier2.py` are deprecated due to the small dataset.
- All prepossessing scripts, ML-models and datasets for the Cicero dataset can be found in `acdc-data-master/Cicero`
  - `train.csv` is Normal training data. `train_augmented.csv` is augmented training data. `val.csv` is testing data. For transfer-learning experiments, `train.csv` and `val.csv` are concatenated.

- `augment_data_gpt.py` is used to make similar texts with GPT API
- `autoTrain.py` is used to make classification reports from AutoTrain models
- zero-shot experiments are scripts with prefix: zero_shot
- one-shot experiments are scripts with prefix: one_shot
- two-shot experiments are scripts with prefix: few_shot
- fine-tuned models are `multi_class_climateRoberta.py`, `multi_class_Roberta.py`

Bibliography

[1] Accenture. More than half of consumers would pay more for sustainable products designed to be reused or recycled, accenture survey finds, 2019. URL https://www.businesswire.com/news/home/20190604005649/en/More-than-Half-of-Consumers-Would-Pay-More-for-Sustainable-Products-Designed-to-
Accessed: 2023-01-18.

[2] Birgitte Naderer, Desirée Schmuch, and Jörg Matthes. Greenwashing: Disinformation through green advertising. In *Commercial Communication in the Digital Age*, pages 105–120, 2017. Accessed: 2023-01-16.

[3] Greenwatch. Greenwatch ai, 2022. URL https://greenwatch.ai/greenwatch-methodology/. Accessed: 2023-01-16.

[4] Vinicius Woloszyn, Joseph Kobti, and Vera Schmitt. Towards automatic green claim detection. pages 28–34, 12 2021.

[5] terraChoice. Terrachoice: The sins of greenwashing - home and family edition 2010, Oct 2011. URL https://twosidesna.org/US/terrachoice-the-sins-of-greenwashing-home-and-family-edition-2010/.

[6] Nancy E. Furlow. Greenwashing in the new millennium - na-businesspress.com. URL http://www.na-businesspress.com/JABE/jabe106/FurlowWeb.pdf.

[7] Thomas Diggelmann, Jordan Boyd-Graber, Jannis Bulian, Massimiliano Ciaramita, and Markus Leippold. Climate-fever: A dataset for verification of real-world climate claims, Dec 2020. URL https://arxiv.org/abs/2012.00614v1.

[8] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1441. URL https://aclanthology.org/P19-1441.

[9] Bhamare Bhavana R and Jeyanthi Prabhu. A multilabel classifier for text classification and enhanced bert system. pages 167–176. Association for Computational Linguistics, 2021. doi: https://doi.org/10.18280/ria.350209. URL https://www.iieta.org/journals/ria/paper/10.18280/ria.350209.

[10] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and et al. Language models are few-shot learners, Jul 2020. URL https://arxiv.org/abs/2005.14165.

[11] Company assessments. URL https://cicero.green/company-assessments.

[12] P. Szymański and T. Kajdanowicz. A scikit-based Python environment for performing multi-label classification. *ArXiv e-prints*, February 2017.

[13] Kurtis Pykes. Oversampling and undersampling, Sep 2020. URL https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf.

[14] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL http://arxiv.org/abs/1910.13461.

[15] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach, Aug 2019. URL https://arxiv.org/abs/1909.00161.

[16] Kenneth Leung. Micro, macro amp; weighted averages of f1 score, clearly explained, Sep 2022. URL https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f.

[17] Bex T. Comprehensive guide on multiclass classification metrics, Apr 2023. URL https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd.

[18] Nicolas Webersinke, Mathias Kraus, Julia Bingler, and Markus Leippold. ClimateBERT: A Pretrained Language Model for Climate-Related Text. In *Proceedings of AAAI 2022 Fall Symposium: The Role of AI in Responding to Climate Challenges*, 2022. doi: https://doi.org/10.48550/arXiv.2212.13631.