UNIVERSITY
OF HULL

Identification Of Illegal Wildlife Trade On social media

Final report

Submitted for the BSc in

Computer science

April 2024

by

Joshua Salijeni

Word count: 11,112

Abstract

This article introduces an innovative method for detecting illegal trading of wildlife on social media platforms using transformer-based models, particularly DistilBERT. The project aimed to create a model capable of identifying instances of illegal trade involving elephant parts and products. After extensive experimentation and adjustments, a strong classification model was developed, achieving a test accuracy of 99.85%. The techniques used included transfer learning, tokenization, and data augmentation, demonstrating the effectiveness of transformer-based models for complex natural language processing tasks. Despite facing challenges such as limited data availability and time constraints, the project successfully met its goals, highlighting the potential of advanced machine learning techniques in combating illegal activities. The findings provide valuable insights into the practical application of transformer-based models in real-world situations and emphasize the importance of ongoing research in this area. Opportunities for future work include exploring additional datasets and improving model performance to enhance detection capabilities. Overall, this project illustrates the potential of utilizing cutting-edge machine learning technologies to address important societal issues.

Acknowledgements

I express my heartfelt gratitude to my mother and uncle for their unwavering support and inspiration throughout my academic journey. Their encouragement has been instrumental in pushing me to achieve my best work.

Additionally, I extend my sincere thanks to Professor Adil Khan, my project supervisor. His guidance and expertise in machine learning have been invaluable. His mentorship has shaped my understanding and fuelled my passion for cutting-edge technologies.

Lastly, I am grateful to all the dedicated lecturers at the University of Hull. Their knowledge-sharing and thought-provoking lectures have nurtured my curiosity, broadened my horizons, and enriched my learning experience.

Contents

1 Introduction

This project aims to combat illegal wildlife trade by analysing social media text data to facilitate the identification of instances of illegal wildlife trade on social media platforms. This identification task is done by leveraging the state-of-the-art transformer based large language models, machine learning techniques and algorithms. This project seeks to reveal the intricate networks involved in the illegal sale of wildlife online by carefully collecting data using specific keywords from multiple social media platforms and classifying it depending on whether it offers trade services or publishing a news article. This report will clarify the development process through each step, including a full specification and discussion of the background information required to grasp the complexities of illegal wildlife trade identification on social media. Furthermore, it provides a critical analysis of the project's implementation, including technical development stages and crucial decision-making processes. This review focuses on algorithmic models, their efficacy, and the rigorous testing techniques used to confirm their performance.

The purpose of this project is to contribute to the conservation of wildlife and assessing any presence of trade activities by using machine learning techniques and algorithms. This approach is efficient and cost effective compared to the traditional approaches used by law enforcement and other wildlife governing bodies which have some limitations to them.

The project definition document focused on the identification of illegal wildlife trade in a broad term. But due to the limitation set by twitter to access their API only by subscriptions, a limited amount of data was manually collected, this means that the data used during the project would not be of great quality and would lack diversity to it. For this reason, this project collected data related to elephant wildlife trade and this will be the main focus.

1.1 Background to the project

The rapid growth of the internet has undeniably transformed the way we live, work, and connect. However, with the massive expansion of the cyberspace, a dangerous underworld has formed in which privacy thrives unchallenged and legal borders are hidden. Due to the limitless connection the internet offers, the lack of effective surveillance has unintentionally given a sense of invisibility for wildlife traffickers, allowing them to operate with unprecedented freedom. Yu, X. and Jia, W (2015) conducted Initial research and found that, traffickers use images and information of illegal wildlife items to attract and engage with potential buyers on social media platforms online. The trade of wildlife is highly profitable to traffickers because of the beaty and rarity of the animals and some cultural practices happening in certain regions of the world. a report by Interpol, wildlife trafficking is the second leading underground-economy business, trailing only drug smuggling and barely ahead of illegal arms trade. Its projected annual worth is $6 billion. (Warchol, G.L., 2004). This is a big problem as it shows that for this huge amount of profit made, a substantial number of wildlife species are killed in the process. Mulero-Pázmány et al (2014) reported that, in 2010, a total average of 0.9% rhinoceros were murdered every day. In 2011, it grew up to 1.2%, this number escalated to 1.8% in 2012, leading to over 668 deaths throughout the year, and it is currently an alarming historical high of 2.2% per day in the first two months of the year 2013.

This wide availability of natural language on social media platforms opens a field of natural language processing which employs techniques that can help analyse and understand text from social media platforms to detect such illegal activities. Natural language processing (NLP) is a field that combines computer science and computational linguistics to convert spoken as well as written natural human languages into organized, reusable data. NLP can be used to identify the meaning of a piece of writing or to generate a human-like response by combining linguistic, statistical, and AI techniques (Fanni, S.C et al, 2023).

1.2 Evaluation of the trade and emerging challenges

primarily wildlife has been traded through traditional venues like city markets, trading centres, and stores. However, as with many other business activities, online commerce has affected the mechanics of how unlawful trade occurs (Salas-Picazo et al, 2023). As a result of rising global demand for wildlife and wildlife goods, wildlife trafficking, like other illicit markets, appears to be assisted by the use of social media and related private messaging platforms. Private messaging has a fast global reach, protects privacy, and is adaptable; if one group is removed, another can be started under a new guise (Wyatt T et al, 2022). This adaptability presents a formidable obstacle in efforts to curb the illegal wildlife trade. Bryukhova, S (2021) reported study results showing that there is a significant knowledge gap about internet illegal wildlife trade, leading to a lack of ongoing preventative and monitoring methods to address such a crucial conservation issue. Crimes against animals are seen as low-level crimes on the law enforcement agenda, thus investigations are often rare. As a result, there are little repercussions for individuals who commit wildlife trafficking, which makes it a high-profit and relatively low-risk illegal business (Lavorgna, A, et al, 2020).

Almost certainly, consumer wants and motives will have an impact on wildlife populations indirectly, influencing supply behaviour to change at the opposite end of the trade chain. Song Z et, al (2022) suggest that the illicit supply in the wildlife trade is mostly made up of poachers and illegal trafficking, both of which have a direct impact on natural wildlife populations.

1.3 Addressing the trade challenges through research

Despite these challenges, researchers have been working tirelessly to find ways of tackling this illegal trade activity on social media and the internet in general. Researchers have worked with social media data to train algorithms to detect these post made by sellers online. Stringham, O.C et al (2021) stated that automated data cleaning approaches, including machine learning and Natural Language Processing (NLP) methods, have been shown to improve the analysing of wildlife trade data obtained via the Internet. A potential yet to be explored approach is predicting and extracting relevant web listings based on their phrases, this may reduce the time required for manual data processing processes if the dataset contains a large number of unrelated advertisements.

Natural Language processing (NLP) has witnessed a significant advancement in the recent years, this is due to the emergence of large language models (LLMs). These LLMs are deep learning algorithms that excel at various NLP tasks by leveraging transformer models architectures and training on massive datasets. Transformer encoder-decoder models have gained popularity in natural language processing. The Transformer design effectively trains a deep stack of self-attention layers using residual connections and layer normalization. Positional encodings using linear functions give self-attention with sequence order information. Systematic improvements have been demonstrated across numerous applications compared to traditional multi-layer long short-term memory (LSTM) recurrent neural network models (Irie, K, et al, 2019). These models can used for many downstream tasks and require fine tuning to suit the task at hand. Witteveen, S. and Andrews, M. (2019) used LLMs in an approach that demonstrated to be capable of generating phrases for both sentences and paragraphs, eliminating the need to split them down into smaller bits. These models can be fine tuned to identify any text offering illegal wildlife trade on social media. Since these models are pre-trained on large datasets, it is easy for the models to understand context and the language used on social media post.

1.4 Ethical Issues

Twitter, Facebook, and Instagram were the main sources of the data collected for the project as this is the focus of the project. The data was collected by manually scrapping it using third party APIs. All the data collected remains anonymous, at the time of collection it was possible to trace back the original author of a post or tweet. After cleaning the dataset, all links and associations to the authors were removed and no trace backs were possible at all. All data handling and preparation procedures were ethically approved prior to implementation.

1.5 Aims and objectives.

Classification of Text Data: The main aim of the project is to classify text data collected from social media platforms through scrapping, Once the data is scrapped it is made anonymous so that users can not be traced back for security and ethical reasons. The data undergoes cleaning and augmentation making it suitable for deep learning classification models and a transformer model to train from the data and archive high accuracy. Python libraries such as Keras, TensorFlow and PyTorch are used to aid in the creation of these models. The data is classified as positive (1) for post offering trade and negative (0) for ones which are not, Ultimately, modelling techniques are employed to facilitate supervised learning, aiming to attain conclusive outcomes and establish a foundation for evaluating accuracy, sensitivity, and recall of the results.

Model Investigation: To enhance the project's capabilities in detecting illegal wildlife trade on social media, a pre-trained transformer-based model was used to leverage its state-of-the-art capabilities. The project used the transformer-based model to extract important data representation and then used this data representation to train on a smaller model such as a regression model. The results of these two approaches were compared. The process of using the representations from the transformer model to a small machine learning model is called transfer learning. Transfer learning utilizes a model that has been trained to a specific job by transferring its previous learnings to another model (Dongre, A., 2018).

Fine Tunning Transformer Model: The projects main objective was to fine tune the transformer model to a down streamed task of classifying tweets. The finetuning process of the model included providing the model with only two classes to classify, retraining the model's parameters on a new dataset while keeping pre-trained weights fixed and adjusting parameters such as learning rate, batch size and optimization algorithms.

using Distil BERT tokenizer on the data, model input data preparation and defining a classification head based on the number of target classes needed for prediction. Patel, S (2022) explained that BERT employs the word piece tokenization algorithm, which breaks inputs into smaller components called tokens.

1.6 Research question.

“Can Transformer based- models analyse and detect social media data referring to illegal wildlife trade?”

The aim of the project was to use transformer based deep learning models and modelling techniques for a down streamed tasks of detecting noisy social media data referring to illegal wildlife trade. These techniques will be explored on how noisy data can be cleaned and improve its quality, models will be trained on this data and then evaluated to understand the how using these techniques can aid a model understand context in social media data and make better predictions.

There are many ways which people can use a single word but with different context in the sentence, by using transformer-based models it is easy to implement this functionality. Data from social media has so much noise that if the model learns this noise, it would introduce overfitting and other issues, cleaning the data extensively ensures that this cannot happen and also ensure that ethical standards are being kept in place. Due to the amount of data available in this study, techniques to enhance the data quality after cleaning it will be employed to improve the model’s performance.

2 Literature review

This section aims to discuss the current understanding of how social media is being exploited for illegal wildlife trade, highlighting key themes, methodologies, and knowledge gaps that will inform the research objectives and approach of the present study. By critically examining the existing evidence, this review will establish a foundation for developing more effective technological and policy interventions to combat the digital dimensions of this global environmental crisis.

2.1 Transformer-based deep learning models

Effective NLP requires understanding sequential context in language. Recurrent neural networks (RNNs) recall sequential context in logically built cells. Nevertheless, due to the sequential design of these models, computation is costly, making them hard to scale to huge datasets. This makes the transformer architecture superior to the RNN's when it comes to computational efficiency and the ability to train on large datasets faster (Wang, C., Li, M. and Smola, A.J., 2019).

The transformer architecture was developed based on the multi-head attention in 2017 by Google researchers and proposed in a paper called "***Attention Is All You Need***". The main characteristics of the transformer architecture are the self attention mechanism, multi-head attention, encoder and decoder structure, feed forward neural networks and positional encoding. If we were to view a transformer for language translation as a basic, solid system, it would accept a sentence in a given language, such as English as input through the encoder, and produce its translated counterpart in the same language through the decoder.

Encoder: The encoder of a transformer typically consists of six equivalent layers that use multi-head attention, a two-layer feed-forward network, layer normalization, and an extra linkage. The multi-head attention mechanism calculates attention weights (Raganato, A. and Tiedemann, J., 2018).

Decoder: The decoder follows the same design as the encoder, with six matching layers of multi-head attention and feed-forward networks. There are two types of multi-head attention sub-layers, decoder-self-attention, and encoder-decoder-attention. The decoder's self-attention focuses on prior predictions, which are hidden by a single position. The additional multi-head attention focuses on the last encoder representations and the decoder representations (Raganato, A. and Tiedemann, J., 2018).

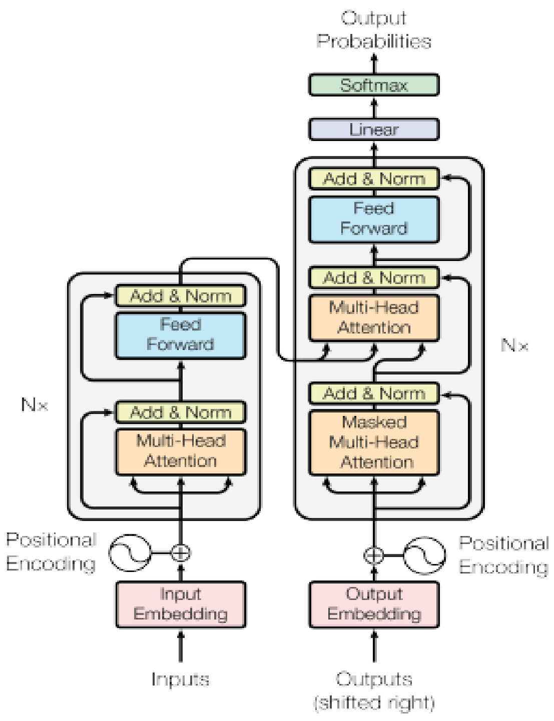


Figure 1 Figure 1: Transformer -model architecture. (Ashish Vaswani, et al, 2017)

Attention: The attention mechanism in a transformer model focuses on important parts of the input by paying close attention to key details. It combines these key details to form context from the input. An attention function maps the query and a set of key-value pairs to an output, whereby all elements are vectors. The result is calculated as a weighted sum of values, with each value allocated a weight based on the query's validity with the key (Ashish Vaswani, et al, 2017).

The attention is computed by taking the query and comparing it to each key to see how closely they match, based on this comparison, the function assigns a weight to each key-value pair based on how closely they match or relate. This was written down as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Figure 2: Attention function of a transformer, (Ashish Vaswani, et al, 2017)

As the amount of keys values and queries increase, the model can use a multi-head attention mechanism whereby multiple attention functions operate in parallel calculating the output value and finally concatenating it to form a final result. Ashish Vaswani, et al, (2017) stated that, rather than using just one attention function for d model-dimensional keys, values, and queries, they figured out how to linearly project them to dk, and dv dimensions, accordingly. This was written down as:

$$\begin{aligned}\mathrm{MultiHead}(Q, K, V) &= \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O \\ \text{where } \mathrm{head_i} &= \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)\end{aligned}$$

Figure 3: Multi-Head Attention function of a transformer, (Ashish Vaswani, et al, 2017)

Positional encoding: In a transformer-based design, the positional encoding is implemented because the model has no recurrence and no convolution, knowledge regarding the relative or precise position of the tokens in a sequence is provided (Ashish Vaswani, et al, 2017). This means that words can be specially marked by the model numerically to derive context based on input text based of the closeness of the numbers each word was marked with.

The Transformer model offers a significant advantage by eliminating the need to break down text into smaller parts, enabling a more natural and comprehensive understanding of the input text. Additionally, models built on this architecture can be finely tuned to identify specific patterns or content, such as text related to illegal wildlife trade on social media platforms. However, despite their effectiveness in numerous natural language processing (NLP) tasks, Transformers have limitations in efficiently modelling word-level sequential context. Unlike RNN-based models like Long Short-Term Memory (LSTM) networks or Gated Recurrent Units (GRUs), Transformers may not consistently capture the sequential relationships between words with the same level of effectiveness. This is partly due to their multi-head attention mechanism, which might not always adequately capture dependencies between words.

2.2 BERT and Distil BERT.

Transformer-based models have gained popularity when it come to NLP task in the recent years, they are many other models such as GPT (Generative Pre-Trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers). In this project the main model focused on is BERT due to its smaller size compared to GPT.

BERT was defined as a pre-trained deep bidirectional representation of words from unlabelled text by joining conditions from left and right context of a word in all layers. BERT is initially pre-trained on a large corpus of unlabelled text, such as Wikipedia and Book Corpus, to learn contextual representations of words and sentences. This pre-training process enables BERT to capture the patterns and semantics of language, which can be utilized for downstream tasks (Jacob Devlin et al, 2018). BERT uses the encoder part of the transformer architecture because of its ability to capture context in a bidirectional approach unlike GPT which uses the decoder part of the transformer architecture, it is not bidirectional. There two different type of BERT which are BERT base and BERT large, BERT base is being referred to in this project. Jacob Devlin et al (2018) stated that the architecture of BERT base consists of 12 transformer blocks, 768 hidden layers and 12 attention heads.

BERT's bidirectional capability to capture contextual information from both left and right directions in a sequence allows the model to fully comprehend the meaning of words in relation to their surrounding context. This makes BERT better suited to tasks requiring understanding context, such as text classification, question answering, and named entity recognition.

Despite BERT being small compared to GPT, it is still a big model for researcher and academics who have limited computational resources available to them making it difficult to train the model from scratch or fine-tune it on large datasets.

Researchers figured a way to compress these big models and make them smaller and fast enough while still retaining much of the bigger model's performance. BERT has its own compressed model which was achieved by a process called knowledge distillation and the final model is called

“DisitlBERT”. Sanh, V. et al (2019) defined knowledge distillation as a compression strategy which involves training a compact model (the student) to mimic the behaviour of the bigger model (the teacher) or ensemble of models. Distil BERT has the same overall architecture as the original BERT.

2.3 Application Of Transformer Based Models in Natural Language Processing

NLP emerged in the 1950s as a combination of artificial intelligence and linguistics. NLP was initially separate from text information retrieval (IR), which utilizes extremely robust statistics-based algorithms to gather and retrieve enormous amounts of text quickly (Nadkarni, P.M, et al, 2011). Computers lack the essential ability to grasp natural language in the same nuanced way as humans do, they cannot infer implied meanings or contextual details. This is the challenge that NLP endeavours to overcome.

One of the earliest forms of NLP to be attempted was the Turing test conducted by Turing in 1950. Turing proposed a test to test if a computer can exhibit intelligent behaviour which is indistinguishable from humans. The test involved a human judge engaging in a text-based conversation with both a human and a machine, if the judge could not consistently differentiate between the human and the machine based in the conversation, the machine is said to have passed the test. Unfortunately, this test does not allow for interim assessment of work along the route, despite setting an eventual target. In NLP, there is a rising worry about building sensitive evaluation models to track growth based on present performance rates (Allen, J.F., 2003).

In recent years new models have been developed which involves a one-time training strategy which is a significant barrier to developing a universal NLP solution. While collecting data and establishing a specialized model to solve a specific problem produces positive outcomes, it requires creating a new solution each time a new issue arises and adapting the model to different domains. To reduce this costly component, choosing a universal multitasking solver may be more advantageous. Gillioz, A., et al (2020) explained that Recurrent Neural Networks (RNNs) were widely employed to handle NLP difficulties. They have been used for a while in supervised NLP models for classification and regression. RNNs' performance is attributed to their Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures. These two components address the vanishing gradient issue by offering an easier solution for gradient backpropagation. It aids in calculating when the sentences are lengthy. However, these models are not great; the fundamentally recurring structure made them difficult to run parallel on numerous processes, and the handling of very long phrases is particularly challenging due to the vanishing gradient.

The limitations of these models prompted the introduction of the transformer architecture, which replaced the recurrent architecture with a self-attention mechanism. This mechanism excels in parallel processing, leading to improved overall performance.

2.3.1 Machine Translation

People from different places speak different languages and often a human translator is required to help overcome this barrier. By leveraging a transformer-based model like Distil BERT, computers can automatically do this task fast and efficiently without the need of human involvement. Neural machine translation (NMT) is a deep learning-based technology that uses an encoder to handle input text and a decoder to generate translated output text. According to scientist, the NMT model surpasses existing Statistical Machine Translation methods (Chen, Y. and Rong, P., 2020). The use of transformer-based models on this task fits best due to the self attention capability and the ability to work in parallel which other models can not easily accommodate. Okpor, M.D., (2014) elaborated that, human translators need extensive understanding of the source language's vocabulary, semantics, structure, phrases, and culture to fully understand and evaluate the text. Translators require extensive knowledge to convey meaning in the target language.

Based on this complexity it is machine translation poses challenging tasks to build accurate models that can translate language clearly with a perfect flow. The difference can be demonstrated by considering our ordinary interpretation of a sentence as basic as "He paddled down the river in a canoe". The ordinary computer interpreter, which only works with grammatical data, will be unable to determine the possibility that the clause "in a canoe" refers to "paddled" or "river.", (Wilks, Y., 2008).

The early days of MT is supposed to begin shortly after World War II, when the first computers were employed to decode codes. The field saw a significant shift in direction in the late 1980s, with the advent of a completely new approach to MT. Two major methods to MT have emerged: rule-based and statistics-based (Somers, Harold et al, 2012).

2.3.2 Text Classification

Natural language processing (NLP) relies heavily on text classification, which is rapidly evolving to improve communication. Several deep NLP models, including CNN-based, LSTM-based, and Transformer/attention-based, have been suggested developed to enhance text classification performance (Li, Z., Wang et al, 2022). Text classification usually involves having a set of classes to properly classify the wanted result and often taking a much more statistical approach to analyse the text data, this can be done by using unsupervised and supervised learning. Minaee, S et, al (2021) justified that, text classification can be accomplished with either human annotation or machine labelling. Automated text classification is crucial for industrial applications with large amounts of text data. Early text classification methods relied heavily on rules, therefore requiring the creation of sophisticated classification rules by professionals. The creation the upkeep of rules was costly and laborious. Since the 1990s, with the emergence of statistical machine learning, text categorization algorithms based on supervised machine learning have been extremely successful (Zong, C, et al, 2021).

Text classification involves extracting features from raw text input and predicting corresponding categories. Several text classification models have been proposed during the past few decades. In classical models, Naive Bayes is the initial model used for text classification. The paper proposes general classification models, including KNN, SVM, and Random Forest (RF), which are commonly used for text classification (*Li, Q., et, al, 2021*).

Traditional machine learning models face a challenge in maintaining context across lengthy sentences when performing text classification. In such scenarios, the attention mechanism proves effective in establishing correlations between words in textual data, making it a valuable asset in the development of deep learning models. Minaee, S et, al (2021) argued that one of RNNs' computational limitations is sequential text processing. While CNNs are less sequential than RNNs, the computing cost to grasp connections between the words in a sentence rises with the expanding length of the sentence, just like RNNs. BERT's ability to generate contextualized vectors of words marks a significant advancement in text classification and NLP technology. Researchers have found that text classification models based on BERT outperform generic models in several NLP tasks, including text classification (*Li, Q., et, al, 2021*).

2.3.3 Next Word Prediction

Predicting the next word focuses on anticipating the subsequent words based on the sequence of words entered by the user. For instance, smartphones can enhance typing speed by offering suggestions for the next word as users input text. As users type a word, the system presents relevant options for selection. The best example of when next word prediction has been used the most is in mobile phones where people are constantly communicating through text, this is important as it decreases the amount of time it takes the user to complete typing. Whenever someone writes a character or word, the computer system recommends the most frequently encountered letter or word. The user can then select the next word that appears. The steps described above will keep on until the text is complete (Hamarashid, H.K, et al, 2022).

Two methods exist for predicting the next word, the Statistical Language Model (SLM) and the Neural Language Model (NLM). SLM utilizes statistical methods to forecast the succeeding word in the sequence by analysing the words that came before it. SLM employs statistical techniques to predict the upcoming word in the sequence based on the words that preceded it. Traditional statistical language models (SLMs) may encounter challenges when dealing with extensive datasets. To address this limitation, neural language models (NLMs) have become increasingly prevalent. With NLMs, every word in the vocabulary is assigned a distributed word vector comprising various features. The collective probability function of word sequences can then be articulated in relation to the feature vectors representing the words within the sequence. NLMs are constructed using feed-forward neural networks, as well as advanced architectures like long short-term memory (LSTM), bidirectional LSTM (Bi-LSTM), and Bidirectional Encoder Representations from Transformers (BERT) (Chawla, M, et at, 2024).

Predictive text tailors its suggestions by analysing the text a user interacts with over time. It compiles a lexicon of words derived from the user's frequently inputted words and phrases. These terms are then evaluated based on the probability of their recurrence in future usage (Devi, P.S, et al, 2023). This brings on context limitations as the models may struggle with context because its only using the text phrases the use is using, without further context, the system may have difficulty discerning whether "overtime" refers to the passage of time or additional work hours beyond regular shifts. This ambiguity can lead to inaccurate predictions and reduce the effectiveness of next word prediction systems.

2.3.4 Question Answering

With the current state of the internet, people always have questions about many things and require validating this information. Search engines can provide a ranked list of documents and web pages but not provide the user with a straightforward answer, to address this problem question answering systems were developed. Question Answering (QA) involves gathering concise, relevant textual responses to natural language questions (Mervin, R., 2013).

An implementation of a question-answering system, typically in the form of a chatbot, has the capability to generate responses when presented with inquiries. One of the pioneering and highly effective implementations of such a chatbot is ALICE Bot, which was created using AIML (Sharma, Y. and Gupta, S., 2018).

Even though transformer models like BERT are trained for this kind of task, it becomes very difficult for the model to provide accurate answers to users as the data used to train the model is constantly changing. Accessing up-to-date data can be time consuming as this data will need to be added to the model each time it surfaces, it can also be challenging if data is not available to complete this task. Dimitrakis, E et al (2020) highlighted that a limitation of previous methods is that relying solely on a single source may not yield the most effective answer. Instead, an optimal solution may involve amalgamating information from various sources using diverse methods of representation. In such cases, the preferred approach is to create hybrid systems that blend multiple types of knowledge representation and utilize all available information, regardless of its nature.

2.4 Approaches for Detecting IWLT On Social Media

This section aims to highlight the methodologies employed for identifying instances of illegal wildlife trade (IWLT). Researchers and organizations are experimenting with novel methods to detect and combat the illegal trading of wildlife on social media platforms by employing Machine Learning. This proactive strategy emphasizes the necessity of leveraging modern technology, such as transformer-based models, to handle IWLT's difficulties in the digital domain.

2.4.1 Detecting IWLT on social media.

Stringham, O.C et, al (2021) conducted text classification on a website where people can post advertisements of their pet birds, the goal of this project was to verify if text classifier models can predict if a post is a relevant listing of a bird being sold or not. The researchers collected data from an Australian site and collected a total of 66, 704 listings which was manually cleaned and labelled. For each listing, the data was manually labelled in manner that if a user is requesting for a bird species it was classified as a “wanted” class, if a listing had no bird being traded it was classed as “junk”, if a trade was being made it was classed as "domestic poultry”. The data was then cleaned following common NLP cleaning procedures such as character and number removal, stop words removal and lower-case conversion. Stop words like “want”, “wants”, “wanting”, or “wanted” were left in the data to separate wanted advertisements from real listings.

The authors used supervised text classification models including, Logistic Regression, Multinomial Naïve Bayes, and Random Forest. The text data was processed by obtaining the “bag of words” and feeding this data to the model, and 10-fold cross validation was used to train the model and evaluate the performance of the models.

For the results the text classifiers showed exceptional performance for the 'domestic poultry' and 'junk' labels, with cross-validated average ROC AUC exceeding 99%, Precision-Recall AUC over 97%, and F1 score exceeding 95%. Although the models for the 'wanted' label performed moderately well, achieving ROC AUC greater than 98%, Precision-Recall AUC exceeding 88%, and F1 score surpassing 77%, they were not as proficient in predicting positive outcomes as negative outcomes. Specifically, the Logistic Regression classifier demonstrated a Specificity of 99% and Negative Predictive Value of 99%.

2.4.2 Challenges and limitations

The limitation that Stringham faced were that the models used are highly context dependant and the text data that was used was only based in Australia and will likely be useless for birds being traded in other countries. Another problem faced was the amount of common irrelevant advertisements found when collecting the data and given that some data used to train the models was not directly linked to wildlife research, these may find it difficult to classify other species of wildlife trade.

In general, some of the challenges encountered when detecting IWLT on social media include the emergence of many groups that offer the trade of illegal wildlife, and some sellers can be wrongly classified as illegal traders when they are completely operating legally and fully licenced to offer wildlife sales. Hernandez-Castro et, al (2015) explained that the illicit online trade of wildlife raises significant concerns due to its challenging detection and identification amidst the vast volume of lawful trade. Certain descriptors, such as "ivory," lack specificity as they refer to both tooth-derived material and a colour. Consequently, much of the current trade detection efforts by law enforcement agencies and non-governmental organizations (NGOs) rely on manual methods, which are extremely time-consuming.

2.4.3 Conclusion

From the inception of general machine learning models for text classification to the recent breakthroughs in transformer-based models, it is evident that many NLP tasks can now be performed with remarkable accuracy. The accessibility of these sophisticated models and tools empowers researchers to push the boundaries of experimentation, driving the technology forward. Such progress holds significant promise for applications in critical areas, including the detection of Illegal Wildlife Trade (IWLT), where precision and efficiency are paramount. As we continue to harness these advancements, the potential to make a substantial impact in combating ILWT and other pressing challenges becomes increasingly tangible.

3 Requirements

This stage will aim to explain the stages taken towards the process of data collection, data preparation, data analysis and data tokenization. These are the tools required to create this project laid out in a step-by-step approach to clearly outline a foundation for understanding the subsequence stages of the project and ensures the transparency and reproducibility of the work.

3.1 System requirements

This project presented the developer with a myriad of unfamiliar challenges, necessitating a steep learning curve to make significant strides towards its completion. Many of these project components fell outside the scope of the initial years of studying BSc Computer Science at the University of Hull, sparking concern about meeting project deadlines. To successfully navigate the project's requirements, the developer had to acquire essential techniques and a thorough understanding of transformer-based models. Pivotal modules that facilitated the developer's grasp of the project's fundamental concepts was Machine Learning and data analysis, drawn from both the university course curriculum and additional self-study during holidays. The skills cultivated through this module and self-directed learning enabled a deeper understanding of machine learning algorithms and their implementation. Moreover, the data analysis component underscored the significance of high-quality data in building effective machine learning models.

Through extensive research, the developer gained insights into the complexities inherent in the project and the foundational model underpinning its architecture. This whole approach of blending academic coursework with independent study and research, provided a comprehensive toolkit for tackling the project's challenges effectively.

The system aims to achieve high accuracy in predicting the classes of tweets without bias or mislabelling. It must seamlessly integrate with popular social media platforms, enabling it to classify text data from these sources. Real-time or near-real-time predictions are essential for timely identification of instances of illegal wildlife trade. These requirements collectively define the system's functionality and usability, ensuring its effectiveness in addressing the objectives of the project.

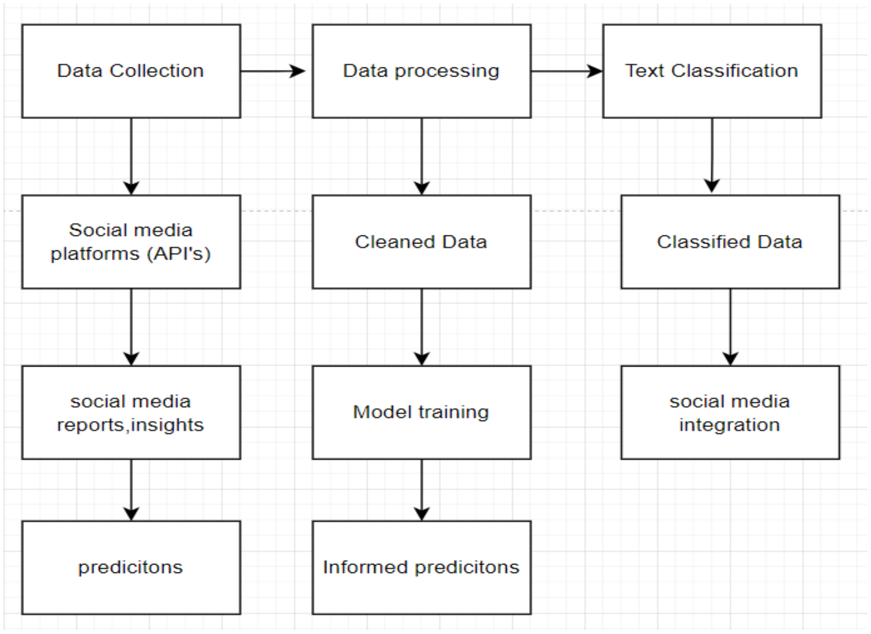


Figure 4 full system layout.

3.2 Functional requirements

This project leverages a multitude of Python libraries, significantly expediting various aspects such as data cleaning and intricate calculations to facilitate overall good results and great performance of the model The utilization of these libraries was instrumental in streamlining processes that would have otherwise posed immense complexity without their aid. Given the project's nature, there exists no standalone software solution parallel to the Jupyter Notebook environment. This choice stems from the project's intricacies and the need for real-time interaction and experimentation with the codebase. Additionally, the flexibility of the Jupyter environment allows seamless integration of various tools and libraries required to contribute towards the project's success.

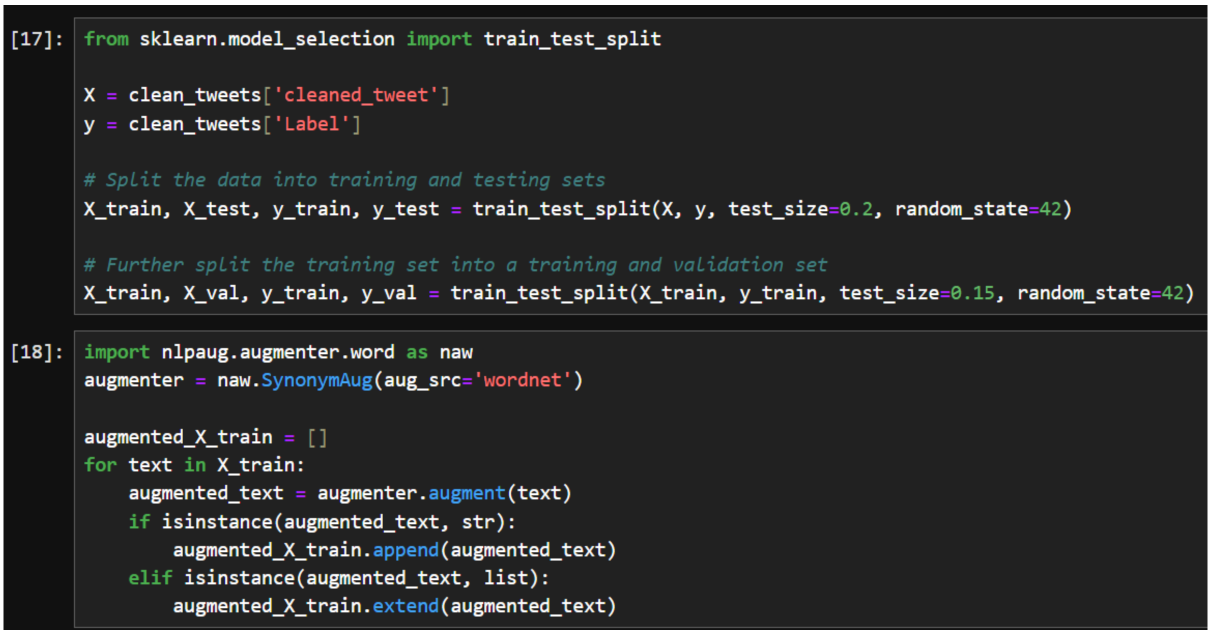
3.2.1 Python

Python serves as the primary language for this project owing to its prowess in building machine learning models with ease. The decision to opt for Python stemmed from its extensive ecosystem of deep learning and machine learning libraries, particularly those tailored for transformer-based models. These libraries streamline code development, offering clarity and facilitating deeper insights into project coding intricacies. Python's user-friendly syntax renders it accessible and comprehensible, contrasting with other languages, thus rendering it exceptionally well-suited for the task at hand. Libraries such Numpy, Pandas, and Pytorch played a crucial role in the development of the project. (Numpy – Numpy, 2024), (Pandas – Python Data Analysis Library, 2024), (Pytorch, 2024).

3.2.2 Jupyter Notebook

Given Python as the designated language for project development, an IDE was harnessed to bolster efficiency and streamline code execution, facilitating seamless testing and program construction. Jupyter Notebook, a web-based open-source application, emerged as the preferred platform, enabling users to generate and disseminate documents integrating live code and visual elements.

Distinguished by its block-based coding paradigm, Jupyter Notebooks simplify code organization and execution. This modular framework empowers users to execute discrete code segments selectively, facilitating iterative testing and real-time adjustments. Such flexibility fosters a dynamic development environment, where developers can refine and optimize code iteratively, ensuring robustness and efficacy throughout the project lifecycle.



```
[17]: from sklearn.model_selection import train_test_split

      X = clean_tweets['cleaned_tweet']
      y = clean_tweets['Label']

      # Split the data into training and testing sets
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

      # Further split the training set into a training and validation set
      X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15, random_state=42)

[18]: import nlpaug.augmenter.word as naw
      augmenter = naw.SynonymAug(aug_src='wordnet')

      augmented_X_train = []
      for text in X_train:
          augmented_text = augmenter.augment(text)
          if isinstance(augmented_text, str):
              augmented_X_train.append(augmented_text)
          elif isinstance(augmented_text, list):
              augmented_X_train.extend(augmented_text)
```

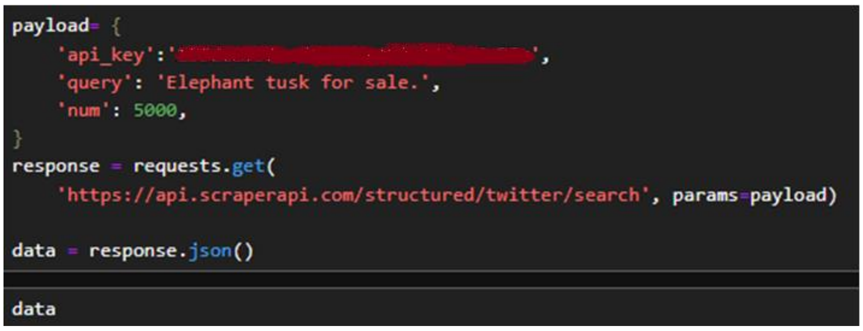
Figure 5 Jupyter notebook code example.

Figure 4 shows how it is possible to organise code blocks in Jupyter notebook. This layout the possibility to run code blocks in order and run them in reverse as well although running the code in reverse sometimes can produce errors. The environment allowed copying and pasting code blocks which enabled code reusability and debugging by making slight adjustments.

3.2.3 Scraper API and Apify

This project involved using noisy data from social media. The data required had to be of a large quantity and was to be data associated with certain hashtags or common words used in the ILWT domain. Elephant related data was the most common data to be found on social media platforms, for this reason hashtags and most commonly used words such as ivory and #ivorytrtade were used to collect gather the data.

Data was collected from multiple social media platforms such as twitter, Instagram and Facebook as these platforms people post millions of posts per day. To collect the data, two scraping tools were used namely “ScraperApi” and “Apify”. These platforms offer the use their API for scrapping data for free and with a paid subscription based on the user needs, a free subscription was used in both platforms which limited the amount of data that can be collected. Both platforms offered the ability to scrape data from social media platforms based on keywords and hashtags, scrapperAPI was integrated with python code to be able to extract the data while Apify had everything set in their platform dashboard ready to use.



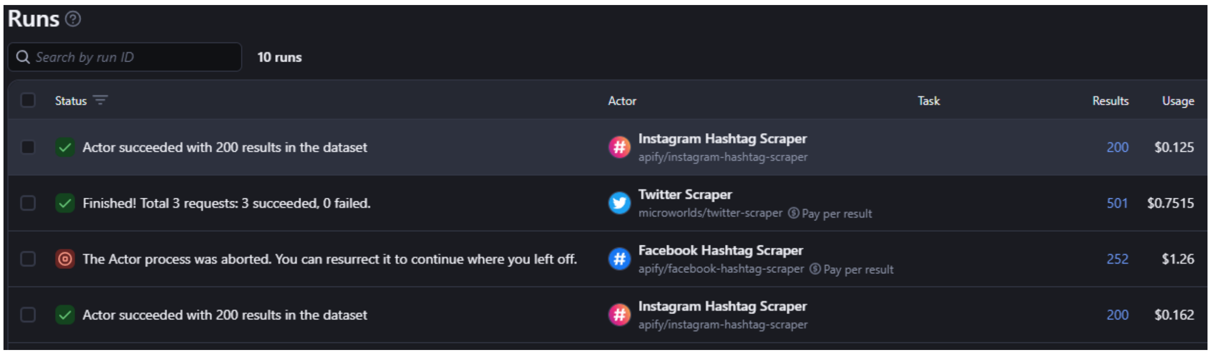
```
payload= {
    'api_key':'[redacted]',
    'query': 'Elephant tusk for sale.',
    'num': 5000,
}
response = requests.get(
    'https://api.scraperapi.com/structured/twitter/search', params=payload)

data = response.json()

data
```

Figure 6 Scraping data using the ScraperApi.

Displayed in Figure 5 is the Python code utilized for scraping data from twitter. The API imposed constraints, permitting only up to 100 Twitter posts per scrape and a monthly allowance of up to 1,000 posts under the free subscription tier. This presented challenges, as the processing speed was sluggish, and the issue of scraping duplicate posts necessitated cautious credit management, ultimately resulting in a diminished volume of collected data.



Runs

Search by run ID | 10 runs

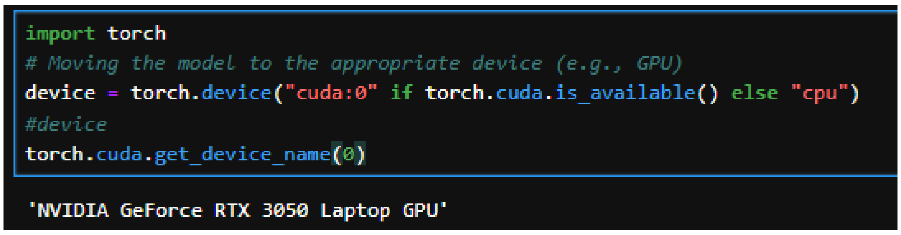
Status	Actor	Task	Results	Usage
Actor succeeded with 200 results in the dataset	Instagram Hashtag Scraper apify/instagram-hashtag-scraper		200	$0.125
Finished! Total 3 requests: 3 succeeded, 0 failed.	Twitter Scraper microworlds/twitter-scraper Pay per result		501	$0.7515
The Actor process was aborted. You can resurrect it to continue where you left off.	Facebook Hashtag Scraper apify/facebook-hashtag-scraper Pay per result		252	$1.26
Actor succeeded with 200 results in the dataset	Instagram Hashtag Scraper apify/instagram-hashtag-scraper		200	$0.162

Figure 7 Apify dashboard for scraping data from social media.

Depicted in Figure 6 is the dashboard interface from Apify, offering the capability to simultaneously scrape data from various social media platforms, a feature not available with ScraperAPI. While this platform permitted the collection of up to 1,000 posts from social media in a single instance, the absence of a premium subscription posed the risk of exceeding the allotted free credits when scraping 1,000 tweets per scrape. After using these platforms to collect data, up to 3,000 social media posts were collected.

3.2.4 Pytorch and CUDA

Using the Transformer-based model required some good computational power, Pytorch supports the use of CUDA NVIDIA graphical processing units (GPU's). Cuda Zone (n.d) defined CUDA as a parallel computing framework and programming approach to general-purpose computing using GPU's. The device which the project was developed on had a GPU that was used to speed the training of the models used, the GPU used was the laptop GeForce RTX 3050.



```
import torch
# Moving the model to the appropriate device (e.g., GPU)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
#device
torch.cuda.get_device_name(0)
```

```
'NVIDIA GeForce RTX 3050 Laptop GPU'
```

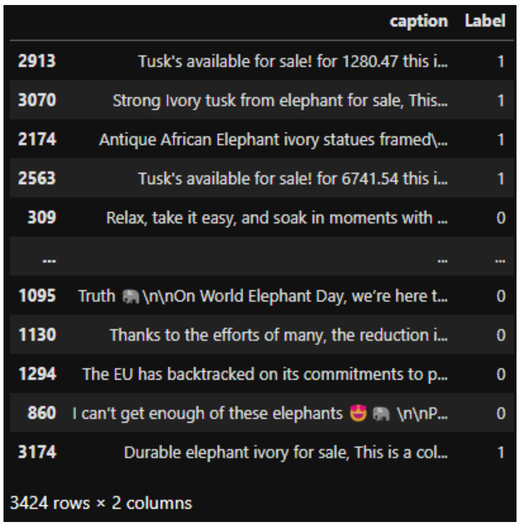
Figure 8 Accessing CUDA GPU using Pytorch.

3.2.5 Data Collection & Exploration

Training data is crucial for building any machine learning model, the project used data collected on social media platforms. The data was collected on post mainly about illegal trade of elephant parts. Keywords and phrases directly associated with illegal wildlife trade, focusing particularly on elephant-derived products like ivory and tusks were used to gather this data.

After the data was extracted from social media it was saved in an excel file and then explored further. The data underwent a process of a process of cleaning elements that were not necessary to be used during the modelling process. The data had multiple columns to it after each final scrape, these columns included things like the date the post was made, the link to the image if the post had one, a username column and many more. These columns were removed as they did not serve any purpose to the project. Regardless of python having a function to remove duplicate data, the data was manually cleaned of any duplicate post. This was done tediously to ensure that the data had less noise in it.

The data needed to be labelled to help the model classify tweets in a supervised manner. The data was manually labelled by identifying all the post that offered wildlife trade and those that are news articles and generic post. The label of 1 was given to the post offering sales and 0 was given to post that were just mentioning about the topic. This process had to be carefully executed as giving the wrong data a label that it wasn't supposed to have could lead in some incorrect classification of post. After this manual exploration and cleaning, the data was read into a Jupyter notebook where further cleaning and exploration was done.

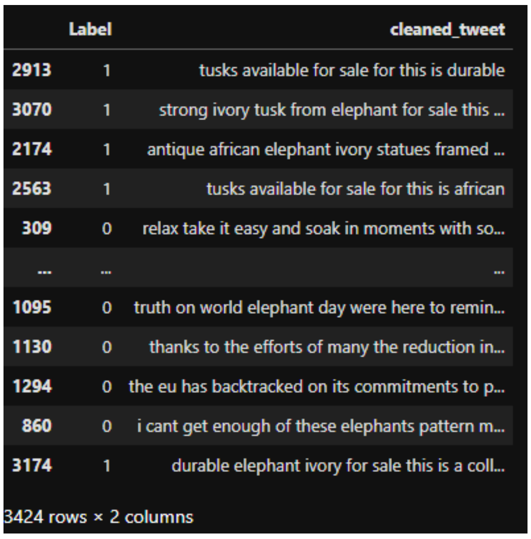


	caption	Label
2913	Tusk's available for sale! for 1280.47 this i...	1
3070	Strong Ivory tusk from elephant for sale, This...	1
2174	Antique African Elephant ivory statues framed\...	1
2563	Tusk's available for sale! for 6741.54 this i...	1
309	Relax, take it easy, and soak in moments with ...	0
...	...	...
1095	Truth 🐘 \n\nOn World Elephant Day, we're here t...	0
1130	Thanks to the efforts of many, the reduction i...	0
1294	The EU has backtracked on its commitments to p...	0
860	I can't get enough of these elephants 😍 🐘 \n\nP...	0
3174	Durable elephant ivory for sale, This is a col...	1

3424 rows × 2 columns

Figure 9 Uncleaned data shuffled.

Before any cleaning or exploration was done the data was shuffled to reduce any uniformity and increase randomness. Then it was checked for any available duplicates again this time using python code. A library commonly known for cleaning data called "re" was used to clean the data by removing website links, symbols, number and emojis in the data.



	Label	cleaned_tweet
2913	1	tusks available for sale for this is durable
3070	1	strong ivory tusk from elephant for sale this ...
2174	1	antique african elephant ivory statues framed ...
2563	1	tusks available for sale for this is african
309	0	relax take it easy and soak in moments with so...
...	...	...
1095	0	truth on world elephant day were here to remin...
1130	0	thanks to the efforts of many the reduction in...
1294	0	the eu has backtracked on its commitments to p...
860	0	i cant get enough of these elephants pattern m...
3174	1	durable elephant ivory for sale this is a coll...

3424 rows × 2 columns

Figure 10 Cleaned dataset.

3.2.6 Common words in the data

The analysis began by identifying the most frequently occurring words in the social media posts, which were extracted with the WordCloud library to create a visual representation of common terms. This involved saving the processed data into a variable, followed by employing the library's function to produce a word cloud. Subsequently, the word cloud was visualized using the matplotlib library, renowned for its ability to generate graphical representations.

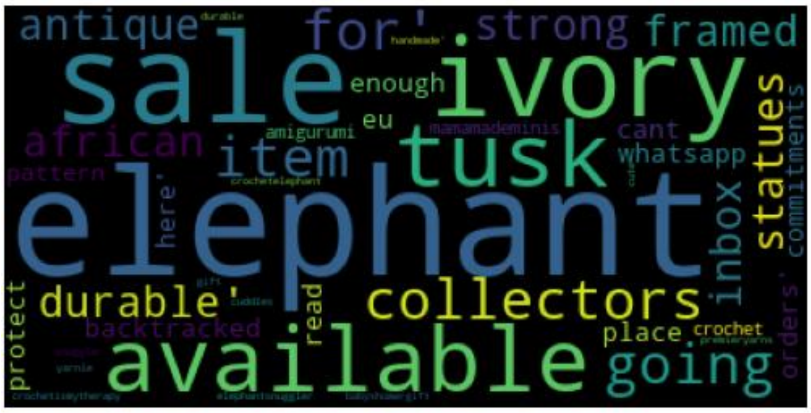


Figure 11 WordCloud of common words from the data.

Illustrated in figure is a WordCloud of common words, the bigger the text in the image the more the common it was in the data and visa versa.

3.2.7 Post character length distribution.

The dataset comprised various social media posts addressing wildlife trade, originating from diverse users with distinct perspectives. As part of the exploration process, one task involved scrutinizing the distribution of post lengths in terms of characters. This analysis is pivotal in grasping user behaviour, particularly due to the character constraints imposed by social media platforms. For instance, Twitter restricts tweets to 4,000 characters, Facebook allows up to 63,000 characters, and Instagram permits posts of up to 2,200 characters. By examining the distribution of post lengths, insights into how users adapt to these limitations can be gleaned. Twitter, for instance, imposes a concise format, fostering brevity and directness in communication. Conversely, platforms like Facebook afford more expansive discourse.

In the analysis, the tweets were segmented into 45 intervals to facilitate a nuanced understanding of their length distribution. Subsequently, a histogram was generated to visually represent this distribution. Notably, the histogram revealed that the majority of posts fell within shorter character ranges, aligning with the nature of social media communication. However, it was observed that some posts extended up to approximately 1,900 characters, indicating instances of more detailed or elaborate discourse within the dataset.

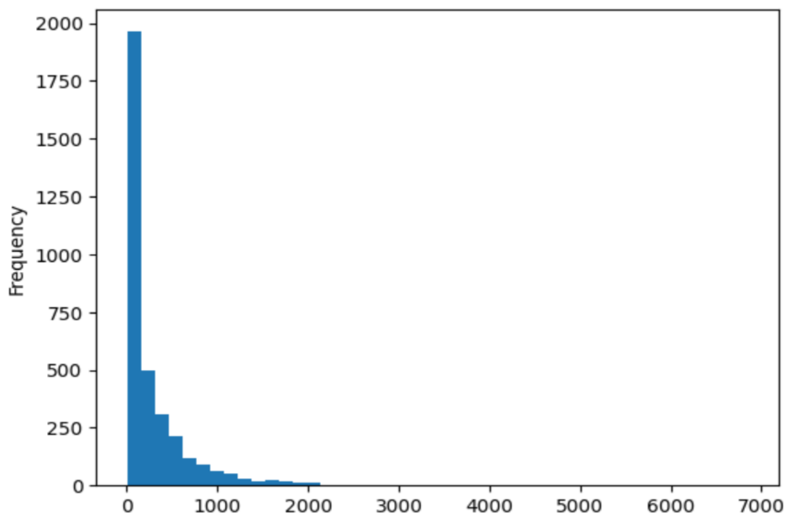


Figure 12 Post character distribution.

3.2.8 Words per post

After counting the character distribution, it was recommended to count the number of words each post contained. This is important because the model used requires that the text has a maximum number of up to 512 tokens. By plotting a box plot, each post in the data was split by words and then counting how many words each post contained. This analysis was conducted on the datasets based on the classes they belonged to. It was found that for the class which had generic posts about wildlife trade, users seemed to post text with more words in them compared to users who are actively offering wildlife services their post seemed to have less words to them.

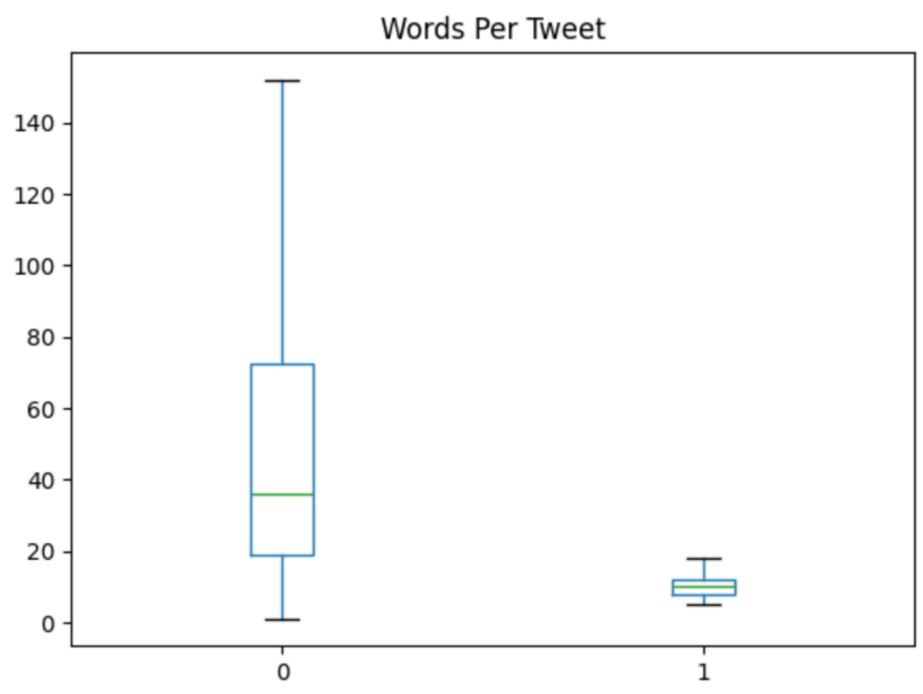


Figure 13 Words per post in each class.

3.2.9 Class distribution

Examining the class distribution of the dataset provided a straightforward means of gauging its balance or imbalance. A balanced dataset showcases approximately equal representation across each class, whereas an imbalanced one exhibits significant discrepancies in class proportions. Gaining insights into class distribution serves as a diagnostic tool for identifying potential biases that could emerge during training and prediction phases. Models trained on imbalanced datasets often exhibit a bias towards the majority class, leading to poor predictive performance for the minority class. During analysis, it became evident that the target class suffered from an imbalance. This inconsistency stemmed from the inherent difficulty in sourcing posts actively offering wildlife trade services due to various factors, such as the illegal nature of such activities, ethical considerations, and legal restrictions.

To overcome this the minority class was oversampled by creating a synthetic data and adding it to the main dataset. The data was made to closely match the language used by users offering the service of illegal wildlife trade from the already existing dataset.

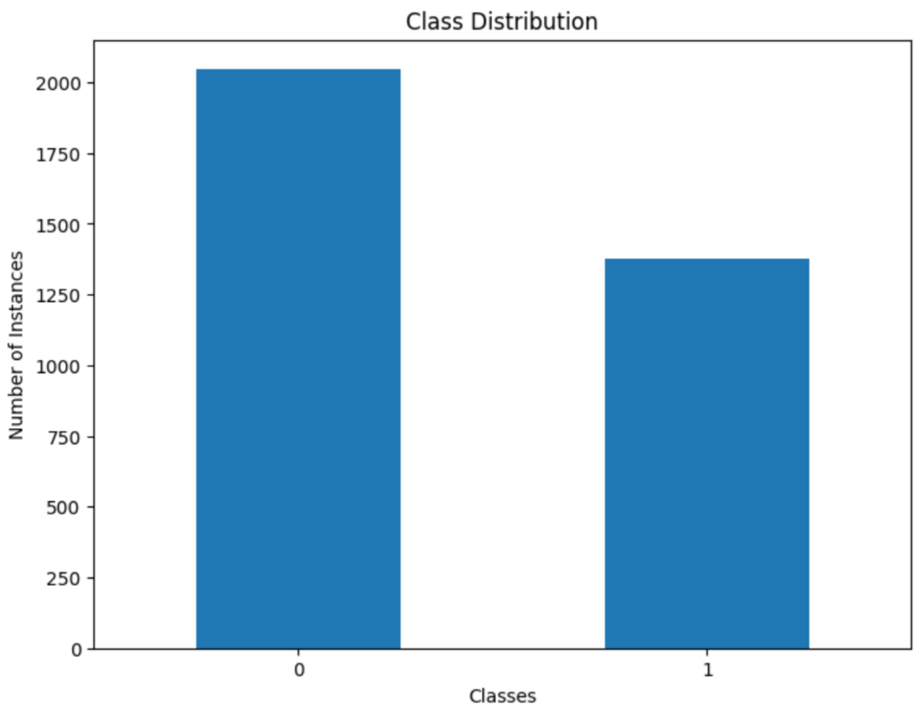
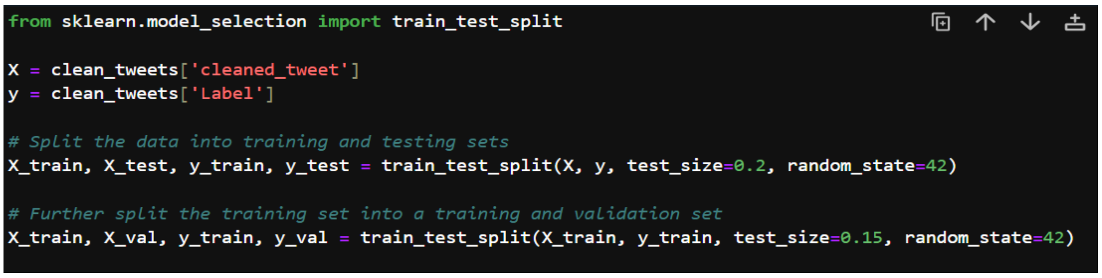


Figure 14 Class distribution after oversampling.

Figure 13 visually depicts the distribution of classes following the application of oversampling techniques. Despite efforts to rebalance the data through oversampling, a noticeable discrepancy persisted between the classes. Specifically, the class labelled as 0 continued to dominate the dataset, comprising approximately 60%, while the class labelled as 1 accounted for around 40% of the dataset. This uneven distribution underscores the ongoing challenge in achieving parity between the classes.

3.2.10 Data splitting

The dataset underwent segmentation using the "train_test_split" from the sklearn library, a tool commonly employed to divide data for training and testing purposes. This project's dataset was partitioned into three distinct segments. Initially, the dataset was separated into training data, followed by a subsequent division to create validation data, and finally, a portion was allocated for testing. The training data comprised approximately 68% of the dataset, while the testing data constituted around 20%, and the validation data was allotted roughly 12% of the total dataset. This meticulous partitioning strategy ensures that the model is trained, validated, and tested effectively, fostering robust performance evaluation and model refinement.



```
from sklearn.model_selection import train_test_split

X = clean_tweets['cleaned_tweet']
y = clean_tweets['Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Further split the training set into a training and validation set
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15, random_state=42)
```

Figure 15 Splitting dataset.

3.2.11 Data augmentation

To enhance the quality of the training data, the data underwent an augmentation process whereby the text data had synonym replaced with other words. The augmentation of the data was done by using a word embedding called word2vec. Word2vec is an unsupervised learning technique that creates a distributed representation of words and phrases in a complex linear space (Jansen, S., 2017). Word2Vec's basic principle is that the context in which a word appears can be used to determine its meaning. It applies the distributional hypothesis, which posits that words that appear in comparable settings have similar meanings. In the dataset only the training dataset was augmented to increase the diversity of the dataset and improve the generalization ability of the models, the choice to only augment the training data was made to avoid data leakage which can introduce artificial patterns that the model may learn to exploit, leading to inflated performance estimates and misleading evaluation results. In the dataset word2vec was used to replace 20% of the words in the data, this was possible by using the genism library that allows a pretrained version of word2vec called “word2vec-google-news-300” which was trained on a large corpus of Google News articles and contains word embeddings for a vast vocabulary of words. By augmenting the data, the problem of having imbalanced classes is addressed by introducing a different variation of the existing data.

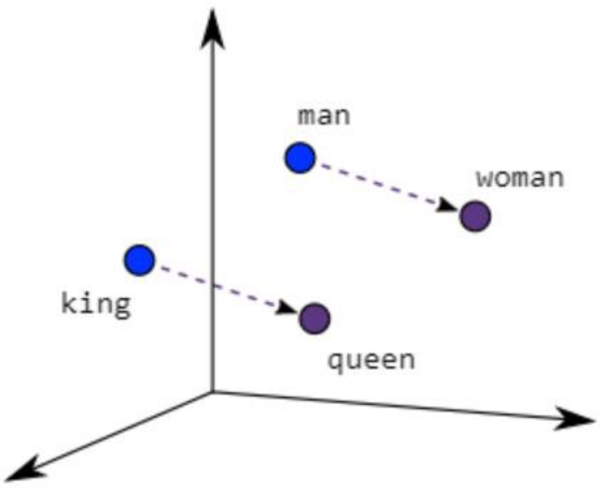
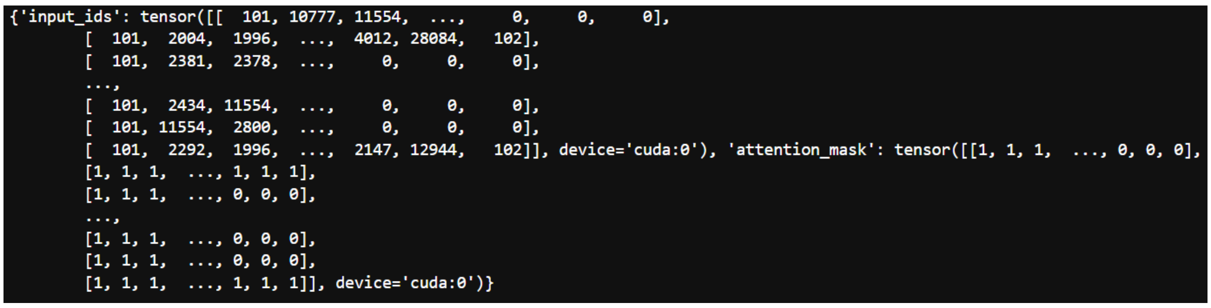


Figure 16 Dutta, M. (2024). word2vec representation

3.2.12 Data tokenization

The model distilBERT requires data to be represented in a format which it can easily understand, and this involves tokenization. The tokenization process involved the segmenting raw text into smaller units known as tokens, which represented words, phrases, or even entire sentences. This step is essential for natural language processing tasks, as it allows the model to understand and process the text at a more granular level. The project uses an auto tokenizer from the transformer library which is specifically trained to tokenize text data in a way that is compatible with the DistilBERT architecture. The tokenizer took text data and split it into tokens according to specified rules and each token generated was mapped to a corresponding integer. The specified rules for the tokenizer were set to be that each token must be truncated and padded, this ensured that all the input text data sequences were all the same length. The truncation process involved removing tokens from longer sequences while padding involved adding special tokens (typically zeros) to the end of shorter sequences. Unlike any non-transformer-based tokenizers, this tokenizer adds special tokens to the beginning and the end of all sequences. These special tokens are the classification tokens (CLS) and the separator tokens (SEP). Finally, the tokenizer generates input ID's which is the tokenized data itself and attention masks which is a binary tensor array that indicates which integer ID to attend to. For instance, if the input IDs are [101, 2001, 2023, 3185, 102, 0, 0, 0] with padding, the attention mask would be [1, 1, 1, 1, 1, 0, 0, 0], indicating that the first five tokens should be attended to, while the remaining tokens should be ignored. After the data was tokenized, the strings inside the data were padded.



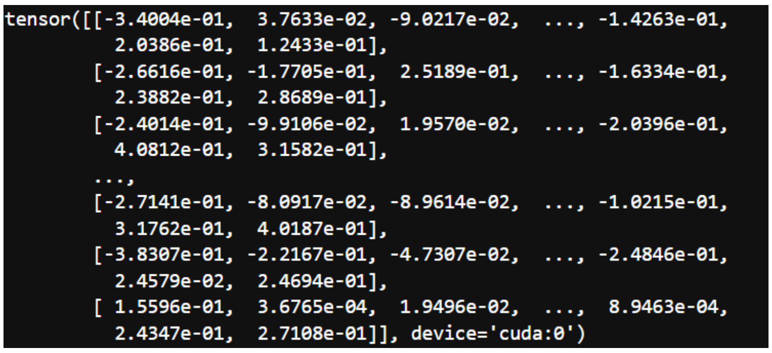
```
{'input_ids': tensor([[  101, 10777, 11554,  ...,     0,     0,     0],
        [  101,  2004,  1996,  ...,  4012, 28084,   102],
        [  101,  2381,  2378,  ...,     0,     0,     0],
        ...,
        [  101,  2434, 11554,  ...,     0,     0,     0],
        [  101, 11554,  2800,  ...,     0,     0,     0],
        [  101,  2292,  1996,  ...,  2147, 12944,   102]], device='cuda:0'), 'attention_mask': tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 1, 1, 1],
        [1, 1, 1,  ..., 0, 0, 0],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 1, 1, 1]], device='cuda:0')}
```

Figure 17 Tokenized data.

3.2.13 Feature extraction

In the process of feature extraction with DistilBERT, a pre-trained model was utilized to derive significant representations from the tokenized text data. These representations encapsulated the semantic nuances of the input text and served as crucial inputs for training a smaller downstream machine learning model dedicated to the classification task at hand. This endeavour was undertaken to facilitate a performance comparison between larger and smaller models.

DistilBERT adeptly extracted embeddings, also known as hidden states, from the input text. These embeddings were pivotal in training a random tree classifier model, which served as the smaller classification model. It's worth noting that these embeddings were exclusively derived from the training and validation tokenized data. This approach ensured that the downstream model was equipped with robust features extracted directly from the DistilBERT model, thereby enhancing its ability to discern and classify text inputs effectively.



```
tensor([[-3.4004e-01,  3.7633e-02, -9.0217e-02,  ..., -1.4263e-01,
          2.0386e-01,  1.2433e-01],
        [-2.6616e-01, -1.7705e-01,  2.5189e-01,  ..., -1.6334e-01,
          2.3882e-01,  2.8689e-01],
        [-2.4014e-01, -9.9106e-02,  1.9570e-02,  ..., -2.0396e-01,
          4.0812e-01,  3.1582e-01],
        ...,
        [-2.7141e-01, -8.0917e-02, -8.9614e-02,  ..., -1.0215e-01,
          3.1762e-01,  4.0187e-01],
        [-3.8307e-01, -2.2167e-01, -4.7307e-02,  ..., -2.4846e-01,
          2.4579e-02,  2.4694e-01],
        [ 1.5596e-01,  3.6765e-04,  1.9496e-02,  ...,  8.9463e-04,
          2.4347e-01,  2.7108e-01]], device='cuda:0')
```

Figure 18 Extracted hidden states from DistilBERT.

3.2.14 Converting Dataset to Pytorch tensors

Training DistilBERT requires the input data to be in pytorch tensor format to ensure compatibility and efficient computation. Pytorch tensor are multi-dimensional arrays that can be efficiently processed by neural network models, raw text data or tokenized data does not always work as input for these models. All data inputs, including input features, labels, and gradients, are represented as PyTorch tensors, simplifying the implementation, and ensuring uniformity across different components of the model. In a separate class functions were created to first convert the data and labels into numerical forma and then finally another method which converted the data to tensors.

Overall, requiring data in the PyTorch tensor format ensured that DistilBERT can take full advantage of the features and optimizations provided by the PyTorch framework, leading to efficient and effective training and inference processes.

4 Design & Implementation

This section delves into the design and development of the models employed in the project aimed at detecting illegal wildlife trade on social media platforms. The process involved meticulous adjustments and fine-tuning of hyperparameters to align with the chosen methodology effectively. Iterative refinement was carried out, with hyperparameters undergoing multiple adjustments to determine the optimal configurations for the models.

Furthermore, this section elucidates on the experimental methodologies adopted, shedding light on the diverse approaches explored to enhance model performance and efficacy. These experimental endeavours encompassed a spectrum of techniques aimed at refining model architectures, optimizing training procedures, and augmenting data inputs. Through systematic experimentation, the project endeavoured to uncover insights into the most effective strategies for combating illegal wildlife trade through machine learning and natural language processing techniques.

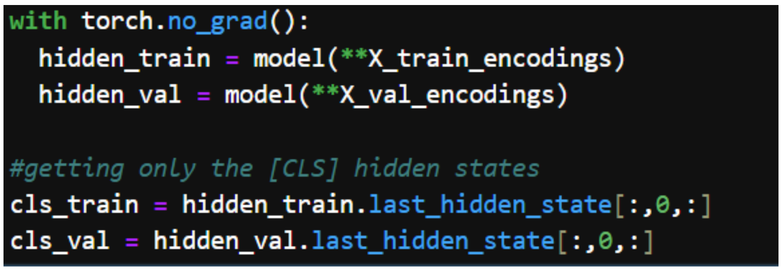
4.1.1 DistilBERT WordPiece tokenizer

To tokenize dataset an auto tokenizer was used from the transformer's library, by using the auto tokenizer the model can choose the appropriate tokenizer for distilBERT to use based on the given model's name or model path. DistilBERT uses a sub word-based tokenizer known as the wordPeice tokenizer, The WordPiece tokenizer starts with a small set of important tokens, which includes special tokens related to the model and the starting letters of words. When using the WordPiece tokenizer, it begins by breaking down words into smaller parts called sub words. These sub words are created by adding a prefix like "##" for BERT to each part of the word. As a result, each word is broken down into sub word units by adding the prefix to all the letters in the word. For instance, the word "word" is divided into sub word units. Sub word tokenization techniques operate on the premise that commonly occurring words should remain intact, while infrequently encountered words ought to be broken down into meaningful sub word units. This approach ensures that the tokenizer effectively captures the linguistic nuances present in the text, enabling more efficient handling of both common vocabulary and rare or specialized terms. By analysing the distribution of word frequencies, subworld tokenization algorithms dynamically adapt to the linguistic characteristics of the input data, facilitating accurate and contextually rich tokenization. The auto tokenizer by dsitilBERT can tokenize sequences with up to 512 characters but due to the heavy computational resources required when perform other task such as feature extraction, this was changed to tokenize only 32. Tokens exceeding this length were truncated, while shorter sequences will be padded.

4.1.2 Disabling gradient calculations

The gradient calculations of distilBERT are disabled during the process of feature extraction to reduce the computational power required and to reduce the amount of memory needed to process the data. During neural network training, gradients are used to update model parameters using techniques such as backpropagation and gradient descent. Calculating gradients adds computational cost, including forward and backward passes over the neural network. Disabling gradient calculations eliminates these superfluous computations, resulting in shorter inference times and lower computational costs.

Since the model being used to extract the features is already pre-trained and the gradient calculations were made already it makes more sense to freeze these parameters as they are and avoid updating them again.



```
with torch.no_grad():
  hidden_train = model(**X_train_encodings)
  hidden_val = model(**X_val_encodings)

#getting only the [CLS] hidden states
cls_train = hidden_train.last_hidden_state[:,0,:]
cls_val = hidden_val.last_hidden_state[:,0,:]
```

Figure 19 Disabling gradient calculations during feature extraction.

4.1.3 Transfer learning and Random Forest Classifier

The project leveraged transfer learning to compare the performance of a transformer model, specifically DistilBERT, with that of a smaller machine learning classification model. To achieve this, a random forest classifier was trained using the extracted hidden states features of the DistilBERT model as input. These hidden states represent high-dimensional vectors that encode various linguistic features such as syntax, semantics, and sentiment, making them valuable representations of the input text. During the feature extraction process, hidden states were extracted alongside their corresponding labels to construct training and validation data vectors. To ensure efficient computation and prevent unnecessary computations, the PyTorch context manager function was utilized to disable gradient calculation during forward passes. This approach also prevented the updating of DistilBERT parameters during feature extraction.

Specifically, only the special classification tokens (CLS) were extracted, as these tokens play a crucial role in determining the sentiment or classification of the input text. By focusing solely on these tokens, the feature extraction process remained streamlined and effective.

Furthermore, all inputs, including the extracted features and labels, were placed on the CPU. This decision was made to enhance compatibility during model training, as the random forest classifier itself was trained on the CPU. This approach was chosen due to the size of the model and the ease of debugging on the CPU compared to the GPU environment. Overall, this strategy ensured efficient feature extraction and model training while maintaining compatibility and facilitating debugging processes.

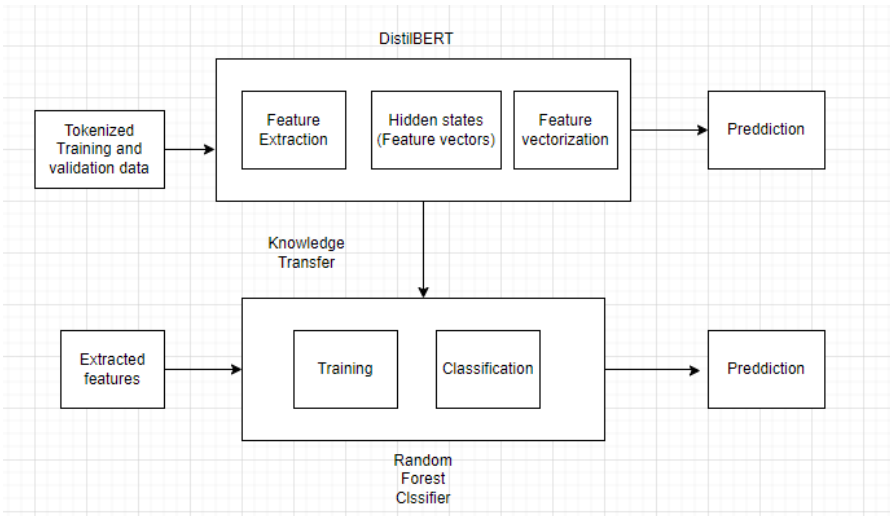


Figure 20 Transfer learning using DistilBERT.

The random forest classifier was trained using the aforementioned extracted feature datasets. Initially, the model was trained without any predefined parameters, allowing it to learn from the data autonomously. Remarkably, this unsupervised training approach yielded impressive results, with the model achieving an outstanding accuracy of 98.54% on the validation data set. Subsequently, the model underwent a second training phase, this time employing a grid search technique to optimize its parameters. Grid search involves systematically testing various combinations of hyperparameters within a predefined range to identify the optimal configuration. For this task, the grid search was conducted over a range of parameter values, including estimator values ranging from 10 to 30 and maximum tree depths ranging from 5 to 15. The model was trained using cross validation with 5 k-folds.

After thorough experimentation, the grid search identified the most favourable configuration for the model. Interestingly, it was determined that the model should have no maximum depth constraint and should utilize 20 estimators for optimal performance. This rigorous parameter tuning process ensured that the model was finely tuned to maximize its predictive accuracy and generalization capabilities. The model managed to achieve a validation accuracy of 98.78% which was slightly better than training the model without specific parameters.

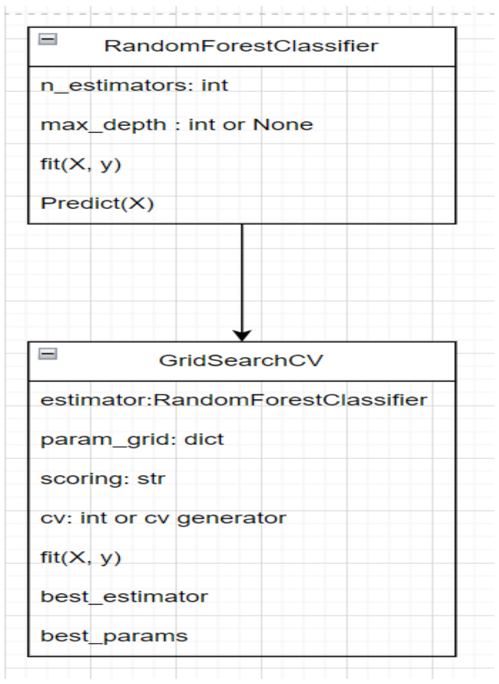


Figure 21 Random Forest classifier with Grid search classes.

The figure above shows that implementation of the random forest tree with grid search, the parameter "n_estimators" determines the number of decision trees to be included in the random forest. Each decision tree in the random forest operates independently and makes its own prediction Increasing the number of estimators typically leads to a more robust and accurate model, as it aggregates predictions from multiple trees, reducing the risk of overfitting. The "max_depth" specifies the maximum depth or levels allowed for each tree, the depth refers to the length of the longest path from the root node to a leaf node. Increasing the number of this parameters also increases the amount of time it would take to train the model.

4.1.4 Pytorch tensors for input

PyTorch tensors are the native data structure of PyTorch, a popular deep learning library. Transformer-based models in PyTorch are intended to function easily with PyTorch tensors, ensuring compatibility and simple incorporation into existing PyTorch workflows. The model is built to be able to run on both CPU and GPU by using Pytorch tensors hardware was leveraged for acceleration capabilities and parallel processing to perform computations quickly and efficiently, which is essential for training and inference with large-scale transformer models.

The project uses a custom class to convert text data to pytorch tensors, within this class they are methods that aid with converting the data to tensors, ensuring uniformity in data representation to be compatible with the model, and indexing the length of the data to ensure accurate iteration of the data during conversion. This class overall serves as a cornerstone in the project's data pipeline, bridging the gap between raw text data and the PyTorch ecosystem. Its seamless integration enables efficient training, validation, and testing of the model.

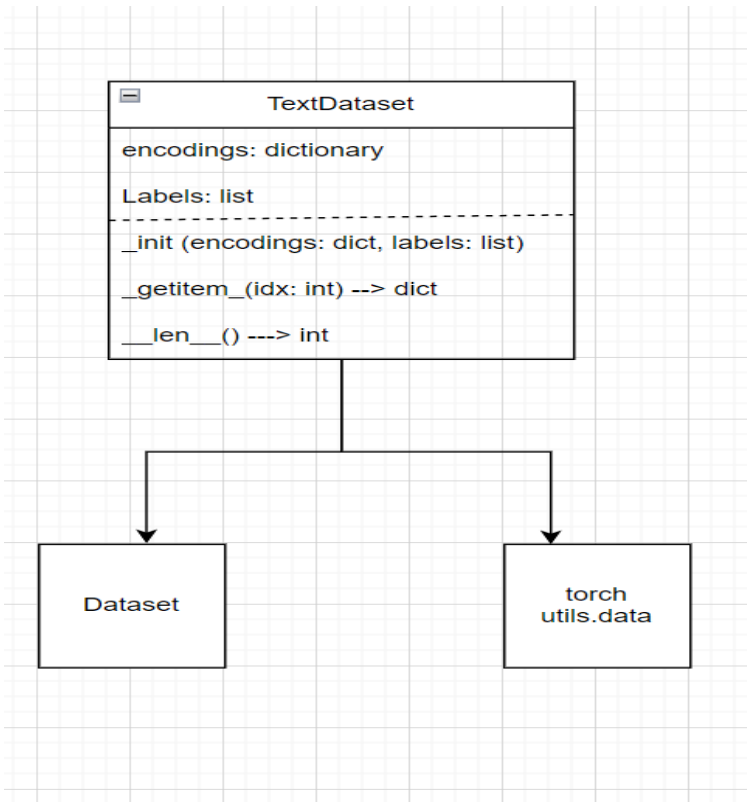


Figure 22 Class diagram for converting data to Pytorch tensors.

4.1.5 Fine tunning DistilBERT

DistilBERT was trained with the pytorch data and used for the tasks of classification with fine tunned parameters. The model is a pretrained model which was downloaded from hugging Face. Hugging Face is a website that builds tools for developing machine learning. The models pretrained parameters were loaded by using the from pretrained function from the transformer's library, this ensured that the model the model used the parameters it calculated during the process of pre training saving time and computational resources. Loading the pre trained parameters enabled the possibility of transfer learning whereby the model could be fine tunned on the specific task of identifying illegal wildlife trade. The design of the model follows the following steps, first of a method that computes metrics of the predictions made by the mode computes the accuracy of the prediction made by the model. Finally, the method calculated the F1 score of the model, the metrics help assess the performance of the model.

DistilBERT was fine tuned by defining training augments which included the output directory for saving the trained model, the number of training epochs set to 5, and a learning rate of 2e-5. Additionally, the batch size for training and evaluation was set to 64, with a weight decay of 0.01 to control overfitting. The evaluation strategy was configured to perform evaluation after each epoch, and the logging steps were determined based on the length of the training dataset and the batch size. The logging level was set to "error" to log only errors, and reporting was disabled. This comprehensive configuration ensured efficient training of the DistilBERT model while providing necessary logging and monitoring functionalities. An auto model for model for sequence classification was imported and 2 classes were passed to it which represented the two classes available in the dataset.

The learning rate controls the step size of the gradient descent optimization algorithm by determining how much the model parameters need to be adjusted during the training iteration. Choosing an appropriate learning rate was important as a higher learning rate would result in faster unstable training. The weight decay regulates parameters and penalize large weights in the model this helps prevent overfitting. Logging steps are based on the length of the training data and batch size, this provides insights into training progress and performance metrics. In the output of the model clutter is reduced by introducing logging level which controls the logging output to only log error messages. Finally, the last parameter set was the report to which specified where the training reports are sent it was set to none as training metrics were visualized during training.

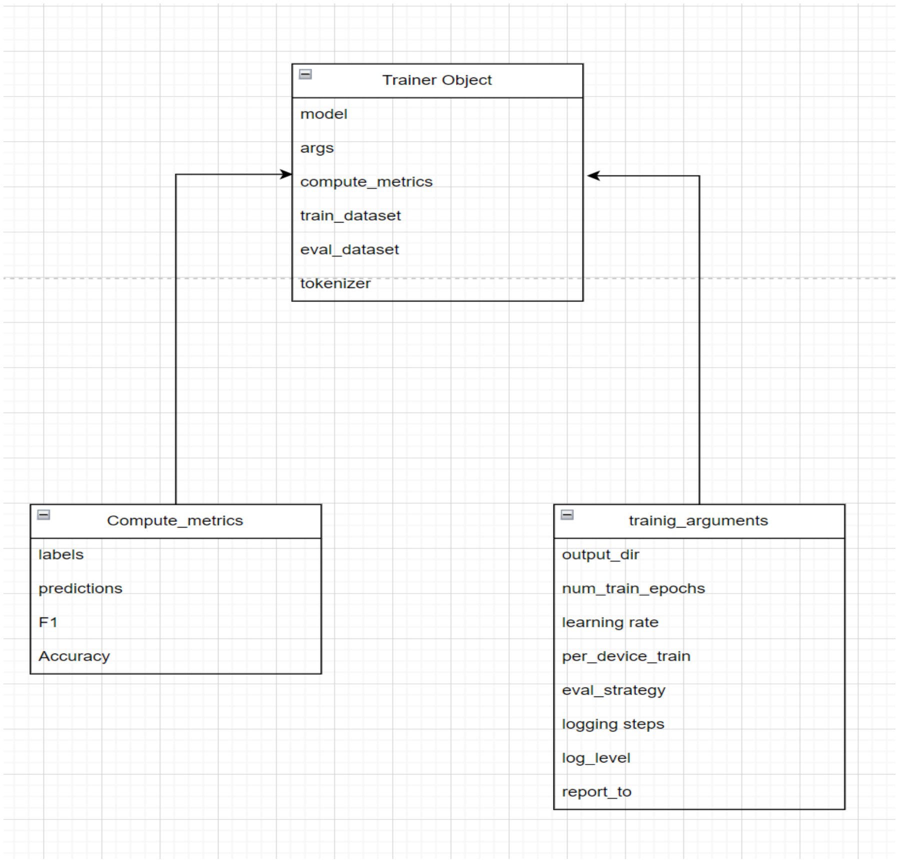
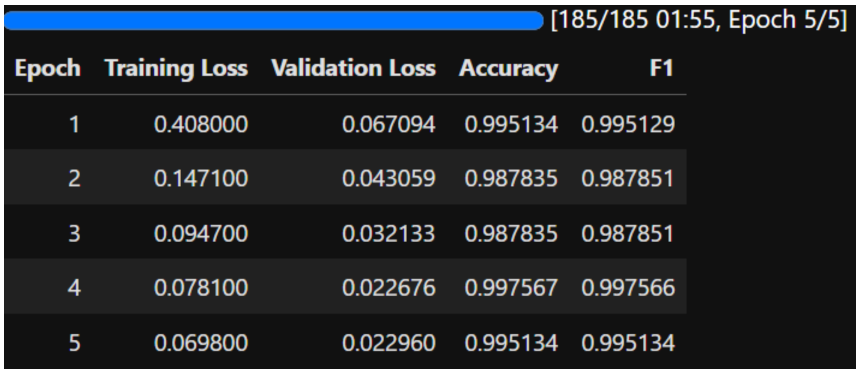


Figure 23 Fine tunning DistilBERT.

Figure 22 illustrates the relationship between objects and methods that make up the fine tunning process of distilBERT. After fine tunning the model and training it with the parameters provided, the model was able to achieve an accuracy of 99.51% on the validation dataset and an F1 score similar to the accuracy score. This demonstrated that fine tunning a transformer model will yield higher accuracy results compared to the regular machine learning approaches.



[185/185 01:55, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.408000	0.067094	0.995134	0.995129
2	0.147100	0.043059	0.987835	0.987851
3	0.094700	0.032133	0.987835	0.987851
4	0.078100	0.022676	0.997567	0.997566
5	0.069800	0.022960	0.995134	0.995134

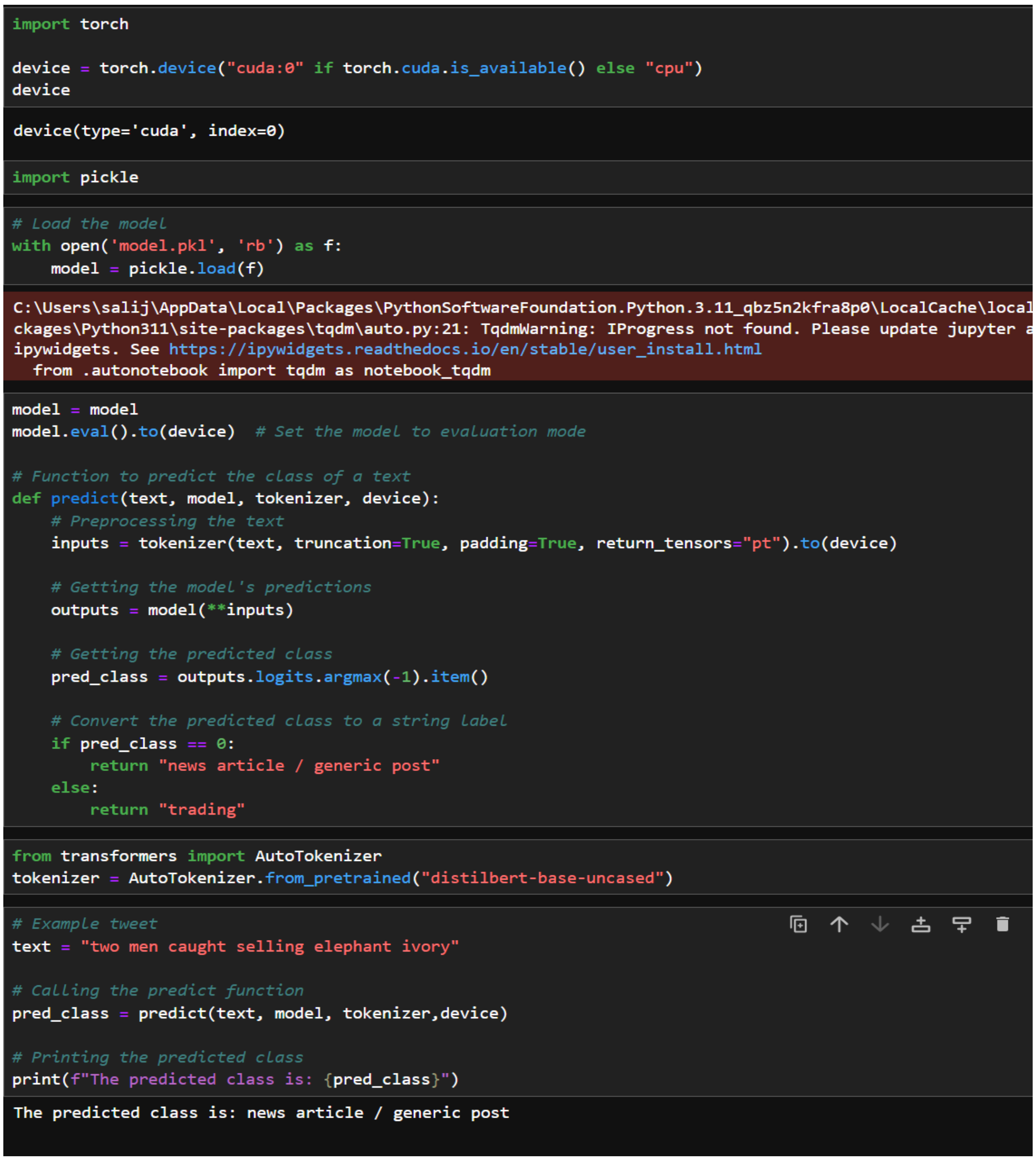
Figure 24 Metrics after training.

4.2 Experimental design

During the processes of hidden states extraction to train a smaller classification model, a pre trained auto model was used for the task and it was able to help the small model archive a high accuracy. This approach was to let the model itself determine what type of model would be suitable based on the model's name given, this model had no number of classes passed into it while extracting the features. The fine tunned transformer model itself also use an auto model but for sequence classification with 2 labels passed to it. The model slightly outperforms the first model with these changes made to it. This indicates that specifying the type of model to perform training is a good idea when more control over the model is required.

5 Testing

After training the model was trained and evaluated with the elephant data, the model was tested using unseen data from the testing dataset. The model was able to make predictions on the unseen data with an accuracy of 99.85%, F1 of 99.85 as well and a test loss of 1.3%. The model was deployed for further testing using the pickle module that serializes the model and preserves the structure of data structures and objects. The model was deployed to allow testing using raw text passed to the model directly whereby this text underwent processing, and the model was able to make predictions and classify the text given to it based on the classes available during training.



```
import torch

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
device
```

```
device(type='cuda', index=0)
```

```
import pickle
```

```
# Load the model
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)
```

```
C:\Users\salij\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local
ckages\Python311\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update jupyter a
ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

```
model = model
model.eval().to(device)  # Set the model to evaluation mode

# Function to predict the class of a text
def predict(text, model, tokenizer, device):
    # Preprocessing the text
    inputs = tokenizer(text, truncation=True, padding=True, return_tensors="pt").to(device)

    # Getting the model's predictions
    outputs = model(**inputs)

    # Getting the predicted class
    pred_class = outputs.logits.argmax(-1).item()

    # Convert the predicted class to a string label
    if pred_class == 0:
        return "news article / generic post"
    else:
        return "trading"
```

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
```

```
# Example tweet
text = "two men caught selling elephant ivory"

# Calling the predict function
pred_class = predict(text, model, tokenizer,device)

# Printing the predicted class
print(f"The predicted class is: {pred_class}")
```

```
The predicted class is: news article / generic post
```

Figure 25 Testing deployed model in a pkl file format.

6 Evaluation and discussion of results

Evaluation of the model was done to assess the possibilities of overfitting and the model being unable to predict certain labels in the data. Techniques such as plotting learning curves and confusion matrices were the main option of evaluating both the deployed and the small undeployed model. The confusion matrix of the small model indicated that they were no sign of overfitting as the precision of the model was 99.39%, recall of 97.02%, accuracy of 98.97 and specificity of 99.59%. This alone was not enough to determine whether the model was overfitting or not, a learning curve was plotted to diagnose the model further. The learning curve indicated that the model was performing well on the training data and similarly to the validation data.

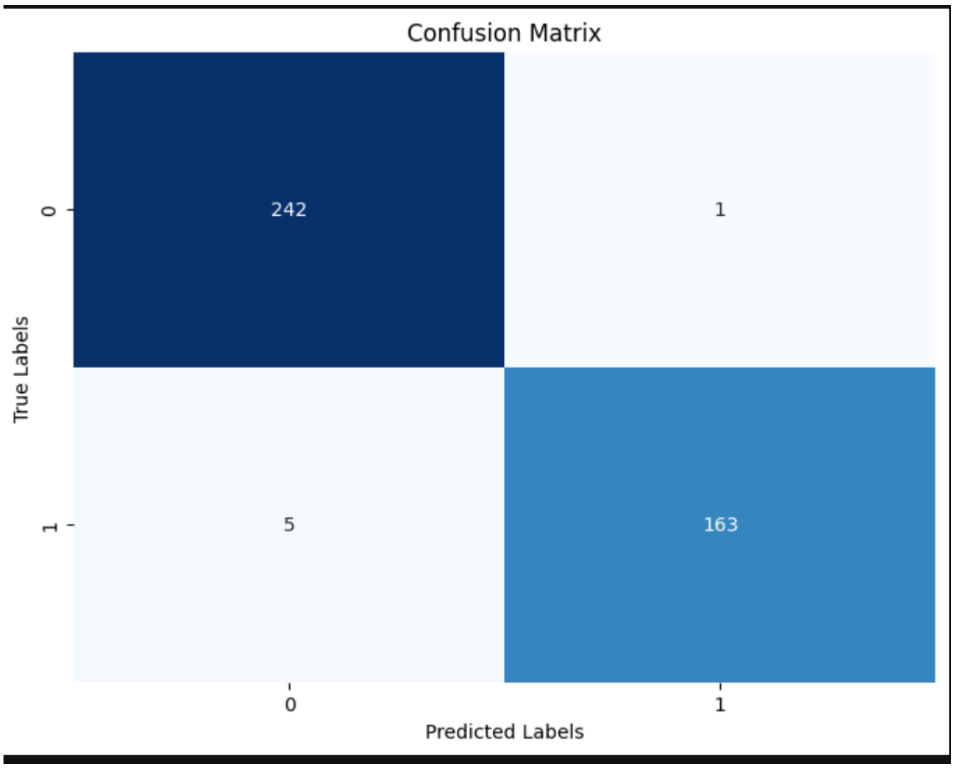


Figure 26 Random Forest classifier confusion matrix.

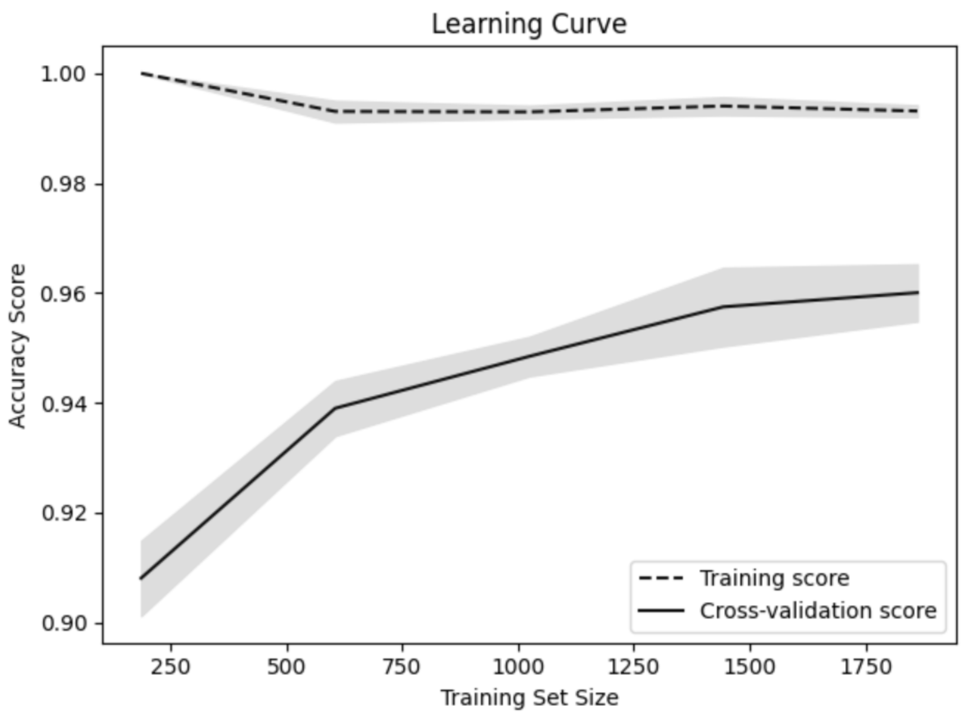


Figure 27Random Forest classifier learning curve.

These techniques were also applied to the deployed model, the deployed model’s confusion matrix had an accuracy of 92.07%, a precision of 63.43%, recall of 99.76% and a specificity of 90.80%. These metrics provided some insight into how the model was performing when given unseen. The model was finally evaluated by plotting the values of the training loss and the validation loss, these plots shown that the training loss was decrease as the number of epochs increased similarly to the validation loss. Overall, these results indicate that the model is not overfitting on the data, but they might be a possibility of bias towards a certain class.

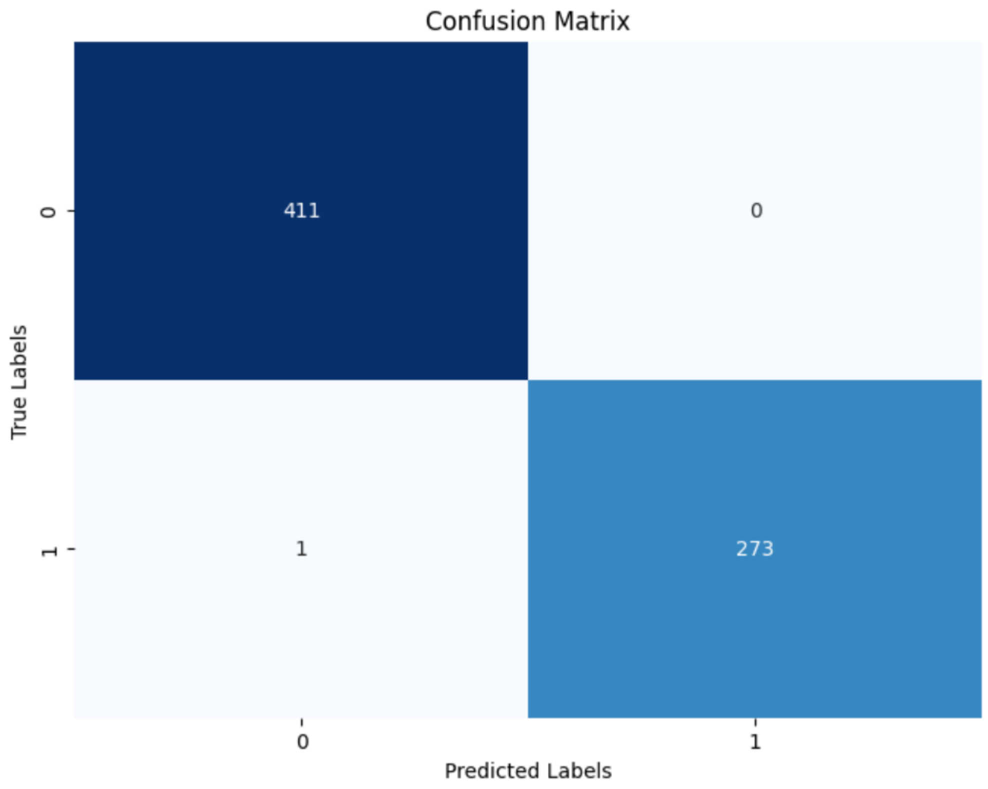


Figure 28 DistilBERT confusion matrix.

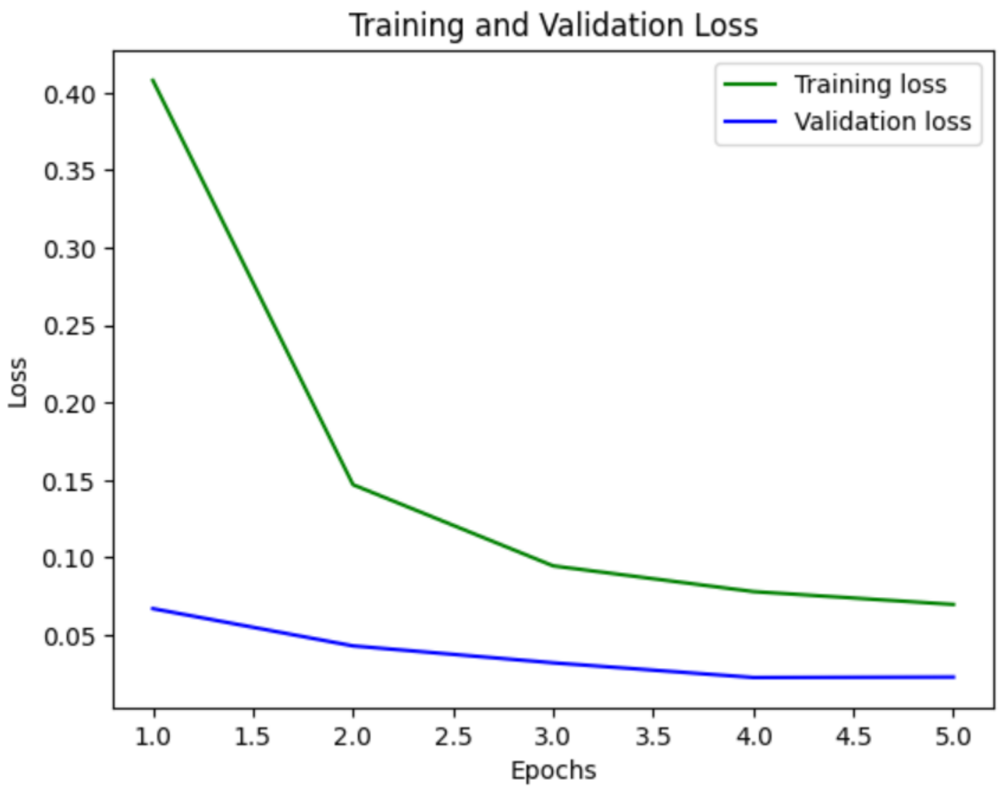


Figure 29 DistilBERT training and validation loss.

The model might be biased towards the class of 1 which indicates that there’s a possibility of trading activity happening, this is because the model predicts well the class of 0 which can be news articles or generic post. This can bring some unwanted behaviour to the model such as difficulty in predicting the label of 1 if the text is slightly changed.

6.1 Design constraints and future directions.

The design of this project had some constrains which were overcome to deliver the project with the main objectives complete. One of the first constrains that was came across was the availability of data if they were enough data to train the model with the project could have followed the main objective of identifying illegal wildlife trade in general rather than picking a subset of the domain data to train on. The second constrain that was faced was the time constrain where the time to complete the project was overlooked which led to extending the time requested for the project to be completed. Some technical challenges were faced due to the lack of understanding of the concepts underlying the conversion of input data to tensors, for a while this problem caused the project to make slow progress as it was difficult to understand and fix the issue.

There’s huge potential to the project if all objectives were met, the project could be trained with more diverse data from all possibilities of illegal wildlife trade rather than focusing on a specific species. In the future the fine tunning process can be implemented in a more rigorous approach and evaluate it further as a system like this can be beneficial for organisation fighting illegal wildlife trade on social media platforms. The approach of analysing only social media data can also be broadened to analyse data from other blogs and e-commerce sites as this would mean that more platforms are being monitored for this illegal activity.

7 Conclusion

The project was ultimately successful, albeit with some adjustments to the initial objectives and encountered limitations during development. A model capable of identifying illegal trade involving elephant parts and products was constructed, achieving a test accuracy of 99.85%. The methodologies employed throughout the project were largely novel, necessitating a learning curve during development. This introduced some risk of project completion due to the complexity of the techniques involved. Overcoming these challenges required conducting numerous experiments to address time constraints. The research question posed at the beginning whether transformer-based models like DistilBERT can identify illegal wildlife trade on social media was unequivocally answered in the affirmative. This conclusion underscores the significance of employing sophisticated machine learning models for tackling complex real-world problems.

The project's contributions extend beyond its immediate objectives. It offers insights into the intricate architecture of transformer-based models and demonstrates the feasibility of fine-tuning such models for downstream tasks, thereby enriching the broader field of natural language processing and machine learning. Reflecting on the project's methodology, while it proved to be appropriate, there were areas where time planning could have been improved. Despite this, the primary and secondary objectives were largely achieved, with any deviations attributed to unforeseen challenges and the iterative nature of the development process.

Moving forward, further work could focus on enhancing the project by exploring additional datasets and refining the model's performance. Additionally, efforts could be directed towards optimizing the fine-tuning process and investigating alternative transformer-based architectures to improve classification accuracy and robustness.

References

Yu, X. and Jia, W., 2015. Moving targets: tracking online sales of illegal wildlife products in China. TRAFFIC Briefing, 1009649, pp.1-10. https://www.trafficj.org/publication/15_briefing_China-monitoring-report.pdf

Warchol, G.L., 2004. The international illegal wildlife trafficking. *Criminal Justice Studies: a Journal of Crime, Law and Society*, *17*(1), pp.57-73. https://www.tandfonline.com/doi/full/10.1080/08884310420001679334

Mulero-Pázmány, M., Stolper, R., Van Essen, L.D., Negro, J.J. and Sassen, T., 2014. Remotely piloted aircraft systems as a rhinoceros anti-poaching tool in Africa. PloS one, 9(1), p.e83873. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0083873

Fanni, S.C., Febi, M., Aghakhanyan, G. and Neri, E., 2023. Natural language processing. In *Introduction to Artificial Intelligence* (pp. 87-99). Cham: Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-031-25928-9_5

Irie, K., Zeyer, A., Schlüter, R. and Ney, H., 2019. Language modeling with deep transformers. *arXiv preprint arXiv:1905.04226*. https://arxiv.org/pdf/1905.04226.pdf

Witteveen, S. and Andrews, M., 2019. Paraphrasing with large language models. *arXiv preprint arXiv:1911.09661*. https://arxiv.org/pdf/1911.09661.pdf

Dongre, A., 2018. *Transfer learning* (Doctoral dissertation, Savitribai Phule Pune University). https://d1wqtxts1xzle7.cloudfront.net/91287643/Seminar_Report_on_Transfer_Learning-libre.pdf?1663657960=&response-content-disposition=inline%3B+filename%3DSeminar_Report_on_Transfer_Learning.pdf&Expires=1715043264&Signature=IwnxqevkuXLSh8bPqsJt0jy2MeMQ9MqIq2SX1SZs3VNX56161BDQs0woSnAJXHFLP~myHRNphwxDHxyczGXQxVMvHlt33MxS8IF9fIEZmLKpTySfCJ8YpmbHZf5GpI4Rir6ywFVXoTfwPdotyYRT0o-sd7Pxz4nbguGXT4pteb1XXPAkDSil8P-cqdOWQ-nXwJ76Bwf3v0-uqH1~6EbLjP~-4tPzmY4dvrdwqaiBD9kSXaN1NzdMd7NMj6HYcFegFQKSkSjhMHOqvMERn5UsDyzd5DykpDGv-YNRmMvvYdWtJvcHczB8m~xIqncKsrWABl1iLzOTJqzWe7tgQAObvg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Patel, S., Shah, B. and Kaur, P., 2022. Leveraging user comments in tweets for rumor detection. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 2* (pp. 87-99). Springer Singapore. https://www.researchgate.net/profile/Prateek-Dutta/publication/354289002_Comparative_Analysis_for_Improving_Accuracy_of_Image_Classification_Using_Deep_Learning_Architectures/links/633bb414769781354eba78f8/Comparative-Analysis-for-Improving-Accuracy-of-Image-Classification-Using-Deep-Learning-Architectures.pdf#page=105

Rosen, G.E. and Smith, K.F., 2010. Summarizing the evidence on the international trade in illegal wildlife. *EcoHealth*, *7*, pp.24-32. https://link.springer.com/article/10.1007/s10393-010-0317-y

Salas-Picazo, R.I., Ramírez-Bravo, O.E., Meza-Padilla, I. and Camargo-Rivera, E.E., 2023. The role of social media groups on illegal wildlife trade in four Mexican states: A year-long assessment. *Global Ecology and Conservation*, *45*, p.e02539. https://www.sciencedirect.com/science/article/pii/S2351989423001749

Wyatt, T., Miralles, O., Massé, F., Lima, R., da Costa, T.V. and Giovanini, D., 2022. Wildlife trafficking via social media in Brazil. *Biological Conservation*, *265*, p.109420. https://www.sciencedirect.com/science/article/pii/S0006320721004729

Bryukhova, S., 2021. The dark side of e commerce: tracking illegal trade of pangolin species on social media. https://repositorio-aberto.up.pt/bitstream/10216/140384/2/540602.pdf

Lavorgna, A., Middleton, S.E., Pickering, B. and Neumann, G., 2020. FloraGuard: Tackling the online illegal trade in endangered plants through a cross-disciplinary ICT-enabled methodology. *Journal of*

Contemporary Criminal Justice, *36*(3), pp.428-450.
https://journals.sagepub.com/doi/full/10.1177/1043986220910297

Song, Z., Wang, Q., Miao, Z., Zhang, W. and Zhou, X., 2022. The dissemination of relevant information on wildlife utilization and its connection with the illegal trade in wildlife. *Journal of Forestry Research*, *33*(1), pp.357-367. https://link.springer.com/article/10.1007/s11676-021-01306-y

Stringham, O.C., Moncayo, S., Hill, K.G., Toomes, A., Mitchell, L., Ross, J.V. and Cassey, P., 2021. Text classification to streamline online wildlife trade analyses. *Plos one*, *16*(7), p.e0254007.
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0254007

Wang, C., Li, M. and Smola, A.J., 2019. Language models with transformers. *arXiv preprint arXiv:1904.09408*. https://arxiv.org/pdf/1904.09408.pdf

Raganato, A. and Tiedemann, J., 2018, November. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: analyzing and interpreting neural networks for NLP* (pp. 287-297). https://aclanthology.org/W18-5431.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. http://arxiv.org/abs/1810.04805.pdf

Sanh, V., Debut, L., Chaumond, J. and Wolf, T., 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
https://arxiv.org/pdf/1910.01108.pdf%3C/p%3E

Nadkarni, P.M., Ohno-Machado, L. and Chapman, W.W., 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), pp.544-551.
https://academic.oup.com/jamia/article/18/5/544/829676

Allen, J.F., 2003. Natural language processing. In *Encyclopedia of computer science* (pp. 1218-1222).

https://dl.acm.org/doi/pdf/10.5555/1074100.1074630

Gillioz, A., Casas, J., Mugellini, E. and Abou Khaled, O., 2020, September. Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)* (pp. 179-183). IEEE.
https://ieeexplore.ieee.org/abstract/document/9222960

Chen, Y. and Rong, P., 2020, December. Compressed-Transformer: Distilling Knowledge from Transformer for Neural Machine Translation. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval* (pp. 131-137).
https://dl.acm.org/doi/pdf/10.1145/3443279.3443302

Okpor, M.D., 2014. Machine translation approaches: issues and challenges. *International Journal of Computer Science Issues (IJCSI)*, *11*(5), p.159.
https://d1wqtxts1xzle7.cloudfront.net/41487988/IJCSI-_okpor-libre.pdf?1453581418=&response-content-disposition=inline%3B+filename%3DMachine_Translation_Approaches_Issues_an.pdf&Expires=1712858223&Signature=aJLczKrFhYxTkXgKQr9Ku5CTkGpqZ6UM7mP-zXC3zlRu-iOobJVgCRLEexaGhCug03aV0IIF5BwH1DsSmHuqT2M~Mvi3AcjSgL9JritSFIYuUAVBF-U3NidcQSP2k88Sq6uYpKMAOokzNQch8FknTvGIwljE5uHaJ1sHOUwfW23M60w1vx~b1C8Yw6u9rfqZWt5CWy6zWO917b6w~qKcngqYlRkmABaHotUAkJorW3IOpM4wJOv3d016N7Z1T2-

Q2~niBeDZuRjMF4u6v-JFvwz1YUN4fEeU-Adig-JOKPtptUNVOMYG-4AB5calCKjufR0a8pABB8~koqcTajM-rA__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Wilks, Y., 2008. *Machine translation: Its scope and limits*. Springer Science & Business Media.

https://books.google.co.uk/books?hl=en&lr=&id=82xFY5ls_-kC&oi=fnd&pg=PA1&dq=problems+with+machine+translation&ots=vziS6ETEns&sig=PwD_J8h9zW13Yld7H22qN335R7M&redir_esc=y#v=onepage&q=problems%20with%20machine%20translation&f=false

Somers, Harold, ' Machine Translation: History, Development, and Limitations', in Kirsten Malmkjær, and Kevin Windle (eds), *The Oxford Handbook of Translation Studies* (2011; online edn, Oxford Academic, 18 Sept. 2012), https://doi.org/10.1093/oxfordhb/9780199239306.013.0029,

Li, Z., Wang, X., Yang, W., Wu, J., Zhang, Z., Liu, Z., Sun, M., Zhang, H. and Liu, S., 2022. A Unified Understanding of Deep NLP Models for Text Classification. *arXiv e-prints*, pp.arXiv-2206. https://arxiv.org/pdf/2206.09355.pdf

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. and Gao, J., 2021. Deep learning--based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, *54*(3), pp.1-40. https://dl.acm.org/doi/pdf/10.1145/3439726

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S. and He, L., 2021. A Survey on Text Classification: From Traditional to Deep Learning12. ACM Transactions on Intelligent Systems and Technology, 37(4), Article 111. Available at: https://doi.org/10.1145/1122445.11224563.

Zong, C., Xia, R., Zhang, J. (2021). Text Classification. In: Text Data Mining. Springer, Singapore. https://doi.org/10.1007/978-981-16-0100-2_5

Hamarashid, H.K., Saeed, S.A. & Rashid, T.A. A comprehensive review and evaluation on text predictive and entertainment systems. Soft Comput (2022). https://doi.org/10.1007/s00500-021-06691-4

Chawla, M., Panda, S.N., Khullar, V., Garg, K.D. and Angurala, M., 2024. Deep learning based next word prediction aided assistive gaming technology for people with limited vocabulary. *Entertainment Computing*, p.100661. https://www.sciencedirect.com/science/article/pii/S1875952124000296

Devi, P.S., Tejaswini, C.S., Keerthana, M., Cheruvu, M. and Srinivas, M., 2023. Prediction of Next Words Using Sequence Generators and Deep Learning Techniques. *Intelligent Computing and Communication: Proceedings of 6th ICICC 2022*, *1447*, p.171. https://books.google.co.uk/books?hl=en&lr=&id=6jTYEAAAQBAJ&oi=fnd&pg=PA171&dq=problems+with+next+word+prediction+machine+learning&ots=HfwJKK05N5&sig=n-p22HFl4DCf2SE53jHzwBvUfl4&redir_esc=y#v=onepage&q=problems%20with%20next%20word%20prediction%20machine%20learning&f=false

Mervin, R., 2013. An overview of question answering system. *International Journal Of Research In Advance Technology In Engineering (IJRATE)*, *1*, pp.11-14. https://www.researchgate.net/profile/R-Mervin/publication/320978810_An_Overview_of_Question_Answering_System/links/5a055663458515eddb84f105/An-Overview-of-Question-Answering-System.pdf

Sharma, Y. and Gupta, S., 2018. Deep learning approaches for question answering system. *Procedia computer science*, *132*, pp.785-794. https://www.sciencedirect.com/science/article/pii/S1877050918308226

Dimitrakis, E., Sgontzos, K. & Tzitzikas, Y., 2020. A survey on question answering systems over linked data and documents. Journal of Intelligent Information Systems, 55, pp.233–259. A survey on question answering systems over linked data and documents | Journal of Intelligent Information Systems (springer.com)

Stringham, O.C., Moncayo, S., Hill, K.G., Toomes, A., Mitchell, L., Ross, J.V. and Cassey, P., 2021. Text classification to streamline online wildlife trade analyses. *Plos one*, *16*(7), p.e0254007. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0254007

Hernandez-Castro, J. and Roberts, D.L., 2015. Automatic detection of potentially illegal online sales of elephant ivory via data mining. *PeerJ Computer Science*, *1*, p.e10. https://peerj.com/articles/cs-10/

Numpy 2024 – Numpy. [Online] Available at https://numpy.org/

Pandas 2024 – Python Data Analysis Library. [Online] Available at: https://pandas.pydata.org/

Pytorch 2024 – Pytorch. [Online] Available at: https://pytorch.org/

Cuda Zone - Library of resources (n.d) NVIDIA Developer. Available at: https://developer.nvidia.com/cuda-zone#:~:text=CUDA%C2%AE%20is%20a%20parallel,harnessing%20the%20power%20of%20GPUs

Jansen, S., 2017. Word and phrase translation with word2vec. *arXiv preprint arXiv:1705.03127*. https://arxiv.org/pdf/1705.03127

Dutta, M. (2024) *Word2vec for word embeddings -A beginner's guide*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/