CS

COC251

F127167

Analysing Social Media Responses to Cyber Attacks Using Large Language Models

by

Sean Cashin

*Supervisor: Magda Z*ajaczkowska

Department of Computer Science
Loughborough University

May 2024

Abstract

Cyber-attacks are an ever-present feature of modern-day life, from large organisations being at risk to individuals. Large Language Models are increasingly being developed and used in society as their popularity increases. This project investigated how these tools can be used to perform sentiment analysis of how people react to cyber-attacks on social media, with the aim of contributing to understanding of how the public reacts to these attacks.

By using the Reddit API, posts and comments involving discussion surrounding cyber-attacks were collected. After pre-processing, the sentiment analysis was conducted using an LLM. Platform-specific features were also used to gain further insight. The results suggested that most responses to the cyber-attacks were negative in the earlier days of the attack but tended towards more neutral responses as time passed. In addition to this, most cyber-attacks followed similar patterns of responses. The effect of developments in events happening around the attack was also investigated.

Upon completion of the project, objective completion was evaluated. In addition to this, project contributions and ideas for future work were discussed.

Acknowledgements
A thank you to Dr. Magda Zajaczkowska for her guidance and advice throughout the project.

Contents

1 List of Figures

2 Introduction

2.1 Aims

The aim of this project is to understand how social media users have responded to cyber-attacks. This will be done by performing sentiment analysis. By analysing data gathered from social media posts, an idea of the changes in people's feelings and sentiments over time will be created. Large language models will be used for the sentiment analysis.

Before conducting the practical steps, research will be undertaken into relevant literature to inform decisions made throughout the project. These decisions will be related to data collection, data pre-processing, and the tools and approach taken to the sentiment analysis.

If successful, this project will contribute to literature about sentiment analysis by adding to the discussion around how large language models can be utilised to perform this task. In addition to this, this project will improve understanding of how people respond emotionally to cyber-attacks. These contributions can help inform decision making when choosing how to conduct similar analyses.

2.2 Objectives

1. Determine project aims and objectives.
2. Conduct research and write a literature review for key aspects of the project.
3. Using knowledge of sentiment analysis, decide what is being looked for in the data to be collected.
4. Create criteria for the specific cyber-attacks to collect data about and select the attacks.
5. Collect data using appropriate methods.
6. Preprocess and store data appropriately.
7. Utilise Large Language Models to perform sentiment analysis.
8. Represent findings visually.
9. Report on findings and evaluate the results.
10. Evaluate the project.

3 Literature Review

3.1 Introduction

This chapter will involve a review of the available literature to provide understanding of the background of relevant topics. Discussions will also involve how the relevant literature will influence the approach to this project. The inclusion of this chapter meets one of the objectives lined out in section 2.2. Each section of this literature review will involve a background on a topic, then a discussion on how the project will be influenced.

3.2 Cyber Attacks

3.2.1 Background

Cyber-attacks are “intentional efforts to steal, expose, alter, disable, or destroy data, applications or other assets through unauthorized access to a network, computer system or digital device“ (IBM, n.d.). Threats from cyber-attacks include viruses, trojans, worms, and human-based risks. These risks, for example, can involve an employee exploiting weaknesses in an organisation (Stacey, et al., 2021). There are a variety of reasons attacks can occur, such as personal gain or as a political act. Cyber-attacks can disrupt or destroy businesses targeted, with an average cost of USD 4.35 million per attack. Therefore, cyber-attacks have become a serious threat in modern day life (Shu, et al., 2018), and an individual attack can affect a great deal of people. Personal data can be made publicly available, and as such there is a great deal of responsibility placed on those collecting this data to protect it properly.

3.2.2 Project implications

This project aims to add to literature concerning the emotional aspects of cybersecurity (Stacey, et al., 2021) by using large language models to perform analyses like sentiment analysis. When deciding criteria for gathering data on cyber-attacks, this project will prioritize factors such as the scale of the attack, as well as consider other factors that may lead to the generation of lots of conversation. An example of this would be cyber-attacks with a political element.

3.3 Data Collection and Social Media

The process of data collection from online sources can rely heavily on various technologies for data collection. One popular method involves using Application Programming Interfaces (APIs), which are used to access structured data from web sources. An API (IBM, n.d.) provides rules for different applications to communicate, enabling data transfer between systems, allowing companies to share data externally. By using APIs, researchers and analysts can efficiently gather specific datasets, increasing the depth and breadth of information available for web mining techniques.

There are several types of API, they are as follows:

- Open APIs: Open-source APIs that can be accessed with HTTP protocols. These involve defined API endpoints, and request and response formats.

- Partner APIs: These APIs are available to those with credentials for access. Developers typically access these through a public API developer portal. An onboarding process is normally needed for access to login credentials for partner APIs.
- Composite APIs: A combination of multiple APIs. These allow programmers to access several endpoints in a single call.

In addition, many APIs have something known as a wrapper associated with them. This is code that simplifies interactions with an API by providing the API's functionality in a more user-friendly interface, such as in a particular programming language (University of Texas, 2021). Wrappers can make the process of performing multiple API calls faster and easier.

Social media are platforms that use web-based technologies to create highly interactive platforms for individuals to share and discuss user-generated content (Kietzmann, et al., 2011). Social media has seen great increases in usage over the years and this increase looks to continue, with a projected six billion users in 2027 (Dixon, 2023). This extensive usage has led to social media being an essential aspect of modern-day communication. For those looking to collect data, these platforms can a great deal of information.

In addition, conversations are a foundational aspect of social media (Kietzmann, et al., 2011). Websites such as Twitter (also known as X) are known for the exchanging of shorter messages, while social media like blogs or forums can focus on longer form conversations. Reddit is another popular social media platform that is conversation focused, due to it consisting of many different forums (subreddits).

Many social media websites, such as the ones mentioned earlier, have APIs readily available for usage. These are typically partner-style APIs, with an account needed for accessing them. Due to the great deal of conversational data available from high usage, as well as the potential of APIs as an information-gathering resource, this leaves social media as a valuable option for collecting conversation-based data.

However, to prevent abuse of APIs, social media websites sometimes have rate-limits on their APIs. Rate limiting is the use of putting a cap on how often a user can repeat an action in a set time. Often rate-limits are put in place with the intention of stopping malicious activity. (Cloudflare.com, n.d.)

3.3.1 Project implications

For this project, there will be a social media-API driven approach to the data collection process. Due to the conversation-based nature of social media like Reddit and Twitter, both were considered as the source of data collected. These websites are both immensely popular and have the potential to provide access to lots of relevant data. In addition, both websites provide ways to focus on a certain topic for users, with hashtags for Twitter and Subreddits for Reddit, leading to lots of focused discussion. This means there is an opportunity for collection of a large amount of data.

However, due to the restrictive nature of the rate-limits of Twitter (requiring payment for higher tiers of access), it was removed from consideration. Twitter heavily limits the number of tweets an unpaid user can access in a 30-day period. In addition, data collected would be reduced further in pre-processing. For example, tweets classified as "off-topic" would be removed. Furthermore, as data collection would need to be split up evenly between the

attacks chosen, this would further reduce the amount of data that can be collected via Twitter. These factors led to Twitter not being used in the data collection process.

The Reddit API does have rate-limits, but the level of restriction placed upon a user was considered not too restrictive for this project. Access to a suitably large quantity of data is still possible. The Python wrapper Praw was also selected for usage. The documentation available for Praw is informative, and the wrapper itself provides the functionality needed to collect data.

3.4 Data Storage

3.4.1 Background

A database is an organised collection of information, normally stored electronically. Control of a database is provided by a database management system (DBMS) (Oracle , n.d.). Designing a database (Dawson, 2012) requires an understanding of entity-relationship analysis and data normalisation.

Entity-relationship analysis is the process of defining the information relating to an "entity" as well as its relationship with all other entities. Entity Relationship Modelling is a process that assists in the understanding of the complexities of the data in a system. They help to organise data into a logical and structured form. In addition, they should result in an efficient form of storage, so data is neither unnecessarily duplicated nor omitted. (Dawson, 2012)

Data normalisation is the use of techniques to check that an optimal data structure has been achieved (Dawson, 2012). SQL (structured query language) is a high-level language that manages interactions and queries with databases. Normalisation has several "normal forms" for a database to meet. Examples of these are as follows:

- First Normal Form (1NF) Requirements (Dawson, 2012):
    - Remove repeating groups by putting each attribute in a separate column.
    - Create a primary key for each record. A primary key is an identifying entry for a row. All primary keys are unique.
    - Ensure that each field holds no multivalued attributes, each cell in the table must hold a single value.
- Second Normal Form (2NF) Requirements:
    - Meet all the requirements of 1NF.
    - Remove partial dependencies by removing columns that are not fully dependent on the primary key.
- Third Normal Form (3NF) Requirements:
    - Meet all requirements of 2NF.
    - Remove transitive dependencies by taking out columns dependent on non-primary-key columns.

In addition to primary keys, there are also foreign keys. A foreign key is a key taken from another table that shows how an entry is connected to another table. (Dawson, 2012)

3.4.2 Project Implications

When designing the database to store the data collected, the concepts of entity-relationship modelling as well as data normalisation will be applied. If done correctly, this should stop

any issues that may have potentially arisen from improper database design. In addition, SQLite will be used for the database interactions. Another form of data storage, CSV (comma separated values) storage, was considered for this project. These tables (Florida State University, 2016) consist of values separated by commas. Using SQL for databases over CSV allows for structured data as well as query capabilities, to extract specific data according to set out requirements. SQL also allows the use of built in data constraints to ensure data integrity. Overall, were decided to be more appropriate for this project SQL, the main factor in this decision was concerns over proper database design. The software SQLiteStudio (Salawa, n.d.) is a freely available software that supports interacting with databases in an easy-to-use interface.

3.5 Natural Language Processing

3.5.1 Background

Natural Language Processing (NLP) as described by (Bandyopadhyay, et al., 2013) involves the use of computational techniques for the automation of human-like processing of natural language content. Applications of NLP include Machine Learning, Information Retrieval, and Sentiment Analysis. Many NLP applications define an individual word, or token, as a string of letters between spaces. For example, the sentence “The game starts at half 5” would have tokens “The”, “game”, “starts”, “at”, “half”, and “5”. In NLP, sentence meaning is determined by the sum of the meanings of each word in the sentence. There has been a rapid growth in both academic and industry areas in NLP in recent years (Bandyopadhyay, et al., 2013) and predictions suggest it will grow further.

NLP originates from artificial intelligence, linguistics, formal languages, and computation (Ghosh & Gunning, 2019). Initially, rule-based systems were used, but advancements in NLP led to using machine learning and deep learning techniques. NLP differs from regular text analytics in that NLP involves the understanding, interpretation, and manipulation of human language, while text analytics focuses more on extracting meaningful information from text.

Out of NLP, Applied Natural Language Processing (ANLP) (McCarthy & Boonthum-Denecke, 2012) has formed. This differs slightly from NLP, while NLP is focused on refining language processing approaches and testing them against well-defined tasks and datasets, ANLP focuses on applying NLP approaches and concepts to new problems. ANLP researchers assess the effectiveness of approaches in new contexts and find areas for improvement. Sentiment Analysis and Large Language Models can both be viewed as examples of ANLP.

Data preprocessing is the manipulation of data before analysing it. The intention of this process is converting the data to a state of higher data quality (Yerashenia, et al., 2020). Proper pre-processing can result in better outputs from models used on the data. For the preprocessing of text-based data, NLP methods can be applied.

One of these methods is tokenization. As described earlier, tokenization is the process of breaking down a sentence into its individual words, or tokens. An example, (Ghosh & Gunning, 2019) involves the sentence "I am reading a book.". Tokenization would yield the tokens "I", "am", "reading", "a", "book", and ".". Each extracted unit is called a unigram, but tokenization can also produce bigrams (two tokens at a time) or trigrams (three), based on the needs of the analysis. Therefore, N-grams refer to sequences of ‘n’ items in a text. Using N-grams, more nuanced language analysis can be performed. For example, (Ghosh & Gunning, 2019), the trigrams of “The sky is blue” would be “The sky is” and “sky is blue”.

Phrases can be challenging to deal with in tokenization. One way to handle them, for example, would be “United States” being treated as a single entity rather than separate tokens to preserve meaning. Therefore, instances like compound words are often handled separately as the individual words lose context when taken separately. An example of this is attaching ## at the start or end of a token, to show it is part of a larger phrase. So, “United States” would be “United##” “##States”. Tokenization is crucial to NLP as it helps analysing and processing natural language by breaking it down into manageable units.

Text cleaning is the process of removing symbols in text that do not contribute much to sentence meaning. These can be emoticons like “:)”, or symbols like “//” (Ghosh & Gunning, 2019). Regular expressions are used in text cleaning, these are a set of characters that represent a pattern. In Python, the re library is used for searching for these patterns.

Stop Word Removal can also be conducted during preprocessing. Stop words are words that appear frequently in a language, such as “a” and “the”. They serve as structural elements in sentences but carry minimal significance in meaning, so are removed (Ghosh & Gunning, 2019). The elimination of these words means that the analysis can focus on more important words, and the amount of noise in the data is reduced. This means processing of text data is more efficient, in addition, the relevance of significant terms is heightened in tasks like information retrieval. Stop words can vary, due to the different contexts an analysis may have. For example, the word “not” may be removed from the list of stop words due to inverting meanings. “I do not hate trees” with “not” removed has the opposite meaning.

Text Normalization is the process of converting different variations of words or phrases into a standard or base form (Ghosh & Gunning, 2019). For example, “United Kingdom” and “UK” convey the same meaning, despite their differences. Normalization ensures consistent representation. Words with similar meanings but different forms are treated the same, this can lead to more effective natural language processing by reducing variation-induced noise. Some text normalisation techniques are explored below, all are relevant to pre-processing.

Spelling Correction is another normalisation task. Traditionally, this is a time-consuming process, but there are tools available to automate this process, like the Python libraries TextBlob or autocorrect. Without spelling correction, there is a risk of losing out on required information. In searches, incorrect spelling might lead to missed or inaccurate search results, affecting retrieval of relevant information. Meaning can be lost in a sentence if a key word is spelt incorrectly. (Ghosh & Gunning, 2019)

Stemming is a technique used to reduce words to their base or root forms, known as stems. The aim of stemming is to unify variations of words that convey the same meaning. A word can be transformed into various forms depending on how it is used in a sentence. For example, “product” can become “production” when referring to the process or “products” when referring to the plural. Words like these get converted back to their base forms as the same meaning is conveyed. So, in the example given, as the stem would be “product”, these variations are all converted to “product”. This increases consistency in text analysis and processing. (Ghosh & Gunning, 2019)

As stemming can lead to inaccurate results, lemmatization is used to overcome these issues. For example, “battling” getting reduced to “battl”, which is meaningless. The process of lemmatization involves an added check, looking through a dictionary a word’s base form. However, this added check slows down the process. When using NLP techniques,

researchers need to consider if the speed reduction from using lemmatization over stemming is worth the increase in accuracy.

3.5.2 Project implications

The NLP techniques discussed above will be implemented as part of the pre-processing of data collected. These techniques are useful as the meaning of a sentence is kept, and the amount of data to process is reduced. This will result in faster processing for the models discussed later. An example of how this will be used is in text cleaning of reddit text content. In this sort of text content, there is potential for lots of noise in the data that does not add to the meaning of a sentence. Like when a user is being tagged (u/<username>), or a link in a piece of text (https:). By appropriately applying these text preprocessing techniques, the collected data for the models to process will be reduced to only the essentials for deciphering the meaning of a sentence. The aim of this is to minimise the reduction in accuracy from noisy data.

In addition, lemmatization will be used over stemming. This is due to value of preserving the meaning of a sentence. While stemming offers benefits in terms of processing times, the accuracy benefits from lemmatisation outweigh these.

When performing tokenisation, the most suitable tokenisation tools will be found by considering what each tool does. For example, the library NLTK provides a tokenizer suitable for Tweets and other social media texts (Ghosh & Gunning, 2019). Whereas a Large Language Model may have its own, specialised tokenizer. A balance between producing the tokens that capture the meaning of a sentence most accurately as well as providing compatibility for the models will have to be struck.

3.6 Large Language Models

3.6.1 Background

A Large Language Model (LLM) (Elastic NV, n.d.) are typically artificial neural networks that can carry out a variety of NLP tasks. LLMs function by receiving input, encoding it into a format that the model can understand, and decoding it for the prediction of outputs. Many involve transformer-based architecture (Vaswani, et al., 2017) and are trained on massive datasets, hence the “Large” in their name. Typical uses of LLMs are the translation, prediction and generation of text or other content. A common example of an LLM would be ChatGPT, in particular as a generative AI (one capable of producing content). To aid in the use of LLMs, many have APIs (IBM, n.d.) provided with comprehensive documentation available online. An example is the OpenAI API (OpenAI, n.d.), which provides users with many tools for accessing the unique features of the OpenAI Language Models.

Large language models are artificial neural networks. Neural networks (Graupe, 2013), or artificial neural networks (ANNs), are computing systems inspired by the human brain. They attempt to simulate the decision process in nerve cells (neurons) and comprise of different interconnected nodes organised in layers. Layer examples include recurrent, feedforward, embedding and attention layers. Nodes process and send information to each other. The different layers mentioned work as follows (Elastic NV, n.d.):

1. Recurrent Layers: interpret sequential data, such as text or time-series data, capturing dependencies and relationships between elements. After applying a new input, the network output is calculated and fed back to adjust the input. This

iterative process continues until the output stabilizes. In LLMS, recurrent layers help understand the context of words in a sentence by considering their order. (Negnevitsky, 2011)

2. Feedforward Layers: connected layers where information flows in one direction—from input to output. These can be an input layer, at least one hidden layer, and an output layer (Negnevitsky, 2011). These layers perform a series of operations on input data to understand the higher-level concepts. In LLMs, they help in understanding the user intent with text input.
3. Embedding Layers: create embeddings from input text. These embeddings can be the token form representation of the words in a vector form (Vaswani, et al., 2017). In LLMs, these layers allow the model to capture the syntactic meaning of inputs.
4. Attention Layers: enable neural networks to focus on specific parts of input data that are most relevant for the given task. In LLMs, attention layers allow the model to selectively attend to different words or tokens in a sentence, aiding in generating correct outputs (Luong, et al., 2015).

A transformer model is a common architecture of an LLM. It involves an encoder and a decoder. Inputted data is tokenised for processing, then relationships between the tokens are discovered mathematically. Transformer models came about as a departure from traditional recurrent and convolutional architectures. They rely primarily on the attention layer. The core of a transformer revolves around the self-attention mechanism (Vaswani, et al., 2017), allowing the model to weigh the significance of an element in a sequence compared to others. This mechanism, alongside using multi-head attention, layer normalisation, and residual connections, results in excellent performance for various tasks. An example of these tasks are natural language processing tasks. The figure below displays this architecture.

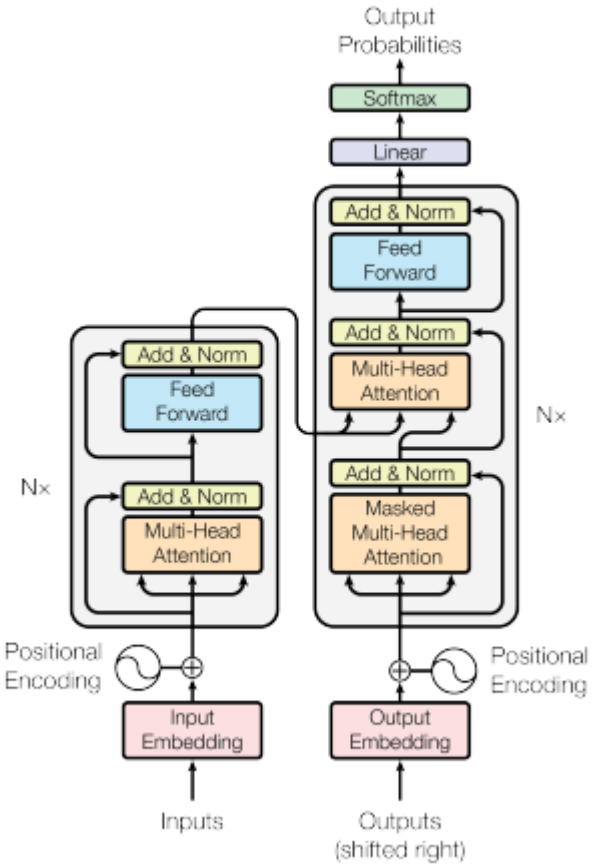


Figure 1: The Transformer - model architecture (Vaswani, et al., 2017, p. 3)

Transformer models utilise a form of tokenization known as vector-based tokenization (Ghosh & Gunning, 2019). This is like regular tokenization in that a sequence of text is broken down into smaller units. However, here a token is represented as a high-dimensional vector in a continuous vector space. Tokens that have similar meanings are closer together in these spaces. This approach offers advantages to the model in that it can better understand context, leading to an overall better understanding of the text.

There are three main types of large language models (Elastic NV, n.d.):

- Generic models: predict the next word based on its training data. Typically perform information retrieval tasks
- Instruction-tuned language models predict responses to instructions given in the input. They can be used to generate code, text, or perform sentiment analysis.
- Dialog-tuned language models are trained in dialog by predicting the next response. An example of this is chatbots.

Due to the versatility of many LLMs, they can be used for a variety of tasks, such as language translation or solving coding problems. One study investigating the performance of LLMs in coding (Chen, et al., 2021) found that “our models displayed strong performance on a dataset of human-written problems”. They improve continuously due to learning through context, meaning existing knowledge is used to quickly adapt to new tasks and inputs. However, despite being a powerful tool, they can produce inaccurate outputs. In addition, LLMs are vulnerable to potentially leaking sensitive data and having biases in outputs. Ethical concerns are sometimes raised about these issues.

Hugging Face is a French-American company focusing on NLP technologies. They are known for developing and supporting the open-source library “Transformers”, which provides implementations of NLP models like BERT, and GPT (Hugging Face, n.d.). This library has gained significant popularity due to its ease of use and extensive documentation. On the Hugging Face website, there are a variety of these transformer models available for use. These models can be fine-tuned for an NLP task or are already pre-trained for a certain task. The Hugging Face community can share models and datasets.

3.6.2 Project Implications

For the sentiment analysis in this project, transformer models made available via Hugging Face will be used. When choosing the models to be used, considerations will be:

- Is the model an LLM?
- Computing resources taken up by the model. Some models, for example, require more memory than others. This means that a balance needs to be struct between model complexity and processing times.
- Suitability towards the given task. Many models are already available for the analysis tasks, pre-fine-tuned. Others may require fine-tuning and training first.

In addition to these considerations when choosing a model, the choice of model will also be a consideration when performing tasks like tokenization. The tokens to be produced need to be in a format that the model will be best able to interpret for accuracy of outputs. As well as token formatting, the maximum size of input for a model needs to be considered too, some text data could be too large and result in too many tokens for a model to process accurately. This may require reducing the number of tokens in a given input to the model to fit.

3.7 Sentiment analysis

3.7.1 Background

Sentiment analysis (Liu, 2020), also known as opinion mining, is the study of analysing the emotions, opinions, sentiments, and attitudes of people, expressed in written text. The written text could discuss a variety of entities, like products, events, issues, or individuals. Sentiment analysis consists of a variety of slightly different and related tasks. These tasks include, but are not limited to, opinion mining, emotion analysis and sentiment analysis. Due to sentiment analysis mostly focusing on written text, it is a common research topic in the field of natural language processing.

This field of study has grown rapidly since the 2000s due to much larger availability of text-based data from the rise in social media such as Instagram or Twitter. Social media content often focuses on people’s opinions and discussions, so social media is a valuable source for understanding societal views. Monitoring opinions in social media helps gauge public sentiments about a variety of diverse topics and can contribute to decision-making processes in areas such as politics and business.

There was some confusion about the distinction between sentiment and opinion in the earlier days of sentiment analysis. This stems widely from lack of distinction between these two terms, as well as the interchangeable nature of the terms “sentiment analysis” and “opinion mining”. Many studies still use these terms interchangeably and are vague about whether sentiment or opinion is being referred to specifically. However, it is now more widely accepted that sentiment refers to a feeling, while an opinion is a more concrete view

about something (Liu, 2020). “I am concerned about <topic>” would be an expression of sentiment, while “I think <topic> is not doing well″ would be an opinion. However, sentiment and opinion are still very intricately linked to each other despite the distinctions that have now been made. Sentiments often can stem from opinions, and opinions can imply sentiments.

In addition, two types of opinions have been defined in sentiment analysis research:

- Regular opinions – opinions expressing sentiments about a particular entity.
- Comparative opinions – comparing multiple entities based on a shared aspect, such as comparing the taste of two foods.

Sentiment analysis was traditionally viewed as a subset of NLP. Now, while sharing many of the same core challenges as NLP, it is considered more as a special case of NLP in general (Cambria, et al., 2013). Each fundamental problem in NLP has an equivalent in sentiment analysis, and vice versa. These problems include aspects like lexical semantics, word sense disambiguation, discourse analysis, information extraction, and semantic analysis.

Sentiment analysis can have various levels of analysis, primarily at a document, sentence, or aspect level:

1. Document-level analysis: focus on the overall sentiment of an entire document, focusing on if there is a positive, negative, or neutral sentiment expressed about a specific entity or topic. This level of analysis assumes that each document relates to a single entity or subject.
2. Sentence-level analysis: finding the sentiment expressed within individual sentences, regardless of the overall sentiment of the document that holds the sentence.
3. Aspect-level analysis: an even deeper form of analysis. Not only is sentence sentiment found, but the specific target of the sentiment. An example, provided by (Liu, 2020), could be the sentence “Although the service is not great, I still love this restaurant.” expressing a positive tone overall, but still expressing a negative sentiment towards the service of the restaurant.

Sentiment lexicons are crucial to sentiment analysis, these are lists of words and phrases that indicate sentiment. These words are typically called sentiment words and are categorised according to if they typically express positive, negative, or neutral sentiments. The creation of these lexicons often relies heavily on the corpus, a structured and labelled set of text.

Another consideration in sentiment analysis research is the type of text content being analysed. They are defined (Liu, 2020) as follows:

- Online dialogue: interactive exchanges among multiple participants, such as a debate or a discussion. Typically, this content holds many opinions, expressing different sentiments and stances on topics. Analysing dialogues can help to identify the behaviours of participants.
- Comments: this type is varied, and can include reviews, questions, answers, and discussions about topics. These can be a mix of stand-alone posts and dialogues.

There are several methods that can be used as an approach to sentiment analysis (Cambria, et al., 2013):

- Keyword spotting – classifying text based on the presence of words like happy, sad, and bored. This is a popular method due to its accessibility, but sometimes cannot

reliably recognize negating words. For example, (Cambria, et al., 2013) while keyword spotting would correctly classify “today was a happy day” as being positive, it is also likely to assign the same classification to “today wasn’t a happy day at all.”.

- Lexical affinity – in addition to detecting obvious affect words, this method also detects and assigns arbitrary words a probable "affinity" to certain emotions. However, it faces two main problems: negated sentences like “I avoided an accident” trick lexical affinity. In addition, the affinities tend to be biased towards the source of the text corpus used.
- Statistical methods – another popular approach to sentiment analysis. Through feeding a machine-learning algorithm a large training corpus, the system might not only learn the affect keywords (like in keyword spotting), but also consider other arbitrary keywords (like in lexical affinity), and punctuation. However, statistical methods can be weak on a sentence level.
- Concept-based approaches - (Cambria, et al., 2013) These methods use semantic networks to analyse natural language opinions, detecting subtly expressed sentiments and analysing multi-word expressions related to concepts. They rely on large semantic knowledge bases and are superior to purely syntactical techniques. However, their effectiveness is limited by the depth and breadth of the knowledge bases they use, which restricts their capability to manage semantic nuances effectively.
- Multimodal sentiment analysis - (Cambria, et al., 2013) This area explores sentiment analysis beyond text, incorporating data sources like audio and video. For example, webcams installed in smartphones and touchpads enable users to post opinions in audio or audiovisual formats, offering opportunity to mine opinions and sentiment.

3.7.2 Project implications

The use of transformer models, specifically transformer-based language models (LLMs), in the analysis of social media data falls under the category of statistical methods. While literature (Cambria, et al., 2013) has suggested weaknesses in older statistical methods such as Bayesian inference and support vector machines, it's important to note that this pre-dates the transformer architecture (Vaswani, et al., 2017). LLMs, unlike traditional statistical methods, do not rely heavily on predefined lexicons or rules; instead, they learn representations of language through training. There has been speculation about the potential of LLMs in performing NLP tasks like sentiment analysis due to their ability to understand, generate, and predict text (Elastic NV, n.d.). This project aims to contribute to literature about the role LLMs can play in sentiment analysis and the strength of this statistical method.

3.8 Example Sentiment Analyses

3 Example sentiment analyses have been identified as relevant examples for this project. They are titled as “Understanding Cyber Attack Behaviors with Sentiment Information on Social Media” (Shu, et al., 2018), “Geospatial sentiment analysis using twitter data for UK-EU referendum“ (Agarwal, et al., 2017), and “Big data analytics and international negotiations: sentiment analysis of Brexit negotiating outcomes” (Georgiadou, et al., 2020) . These analyses will be summarised, followed by an exploration of their implications for this project. While there is further literature available to use as examples, these were considered especially relevant due to their use of social media data.

3.8.1 Understanding Cyber Attack Behaviors with Sentiment Information on Social Media

This analysis looked to use sentiment polarity (definition) as a sensor for analysing social behaviour to predict cyber-attacks (Shu, et al., 2018). The proposed model is based on the observation that Twitter interaction includes emotional signals like emoticons then uses them to find the sentiment score of entire posts. Experiments were performed using real-world datasets from Twitter that corresponds to known attacks.

The model extracts various features from post text, including platform-independent and platform-specific features. Platform-independent features are captured using n-gram features with TF-IDF adaptation, while platform-dependent features (like hashtags) capture specific linguistic patterns in the particular social media platform. The model then applies statistical methods, such as Naïve Bayes, decision trees, logistic regression, K-nearest neighbours (KNN) clustering, and support vector machines (SVM). The data was then used to create a time series with the average sentiment scores per tag per day. The data was combined with cyber incident reports provided by a financial company and a defence company to develop a predictive model of cyber-attacks.

The experimental results show that the proposed sentiment prediction framework can recognize distinct behavioural patterns associated with these attacks. The sentiment prediction model was able to use emotional signal features to classify sentiment with a high degree of discrimination. The cluster analysis results for the hacker event dataset showed three very distinct clusters for positive, negative, and neutral sentiment.

This analysis deployed temporal sentiment analysis, which looks at the variation in sentiment over time. It was found that there tended to be a stronger negative sentiment before an attack, with a slow rise afterwards. This suggests behavioural patterns in sentiment over time for indicating cyber-attacks.

Finally, the performance of the model in predicting an attack before it occurs was observed. The model performed better in predicting malicious-email or endpoint-malware attacks, with high precision and recall scores.

Project Implications

The use of statistical methods here will support the use of similar methods in this project. For example, while the data collected will not be pre-labelled, it will be explored how similar metrics (precision and recall scores) can be examined to determine the performance of the model selected. In addition, deriving insight through variations in sentiment over time like in the study discussed above will be deployed in this project. While Reddit does not allow (at time of writing) searching by date, data collected will have the date of posting stored, so this will still be possible.

3.8.2 Geospatial sentiment analysis using twitter data for UK-EU referendum Brexit

This analysis looked to perform sentiment and geospatial analysis on tweets about Brexit. The study analysed real-time tweets from the UK-EU referendum, collecting data on hashtags and commonly occurring hashtags related to the referendum. The dataset consisted of 4,55,345 tweets, with only 0.6% being geotagged tweets. The data was pre-processed using the AFINN-111 dictionary, which holds 2477 words with their pre-assigned

sentiment strength and polarity. This also involved negation words, to adjust the sentiment polarity of the word following it.

The study's deployed geospatial analysis. This involved an analysis for tweets based on the location of events versus the tweet distribution for that event on a global level. In addition, a sentiment analysis was provided for these geotagged tweets based on the location of events versus geotagged tweet distribution for that event on a global level.

The most frequently used hashtags were "brexit," "euref," and "eureferendum." The hashtag "leave" carried more negative sentimental value than positive, while hashtags like "euref" or "eureferendum" had more tweets with positive sentiment. The number of tweets with positive sentiments for Theresa May (around 74%) was more than other politicians, like Boris Johnson. The analysis suggests that majorly people from America and Europe tweeted in support of "leaving," with 80% of Americans wanting Britain to leave the EU. Around half of European votes were in support of leave and half were in support of remain. Frequently used terms were displayed on a word cloud. The sentiment distribution of hashtags and the mention of different politicians were also displayed for providing insight.

Project Implications

When displaying the results of the sentiment analysis performed here, platform-specific features will be used to derive further insight, like in the one discussed above. For the platform of choice, Reddit, this will be finding sentiment distribution across subreddits. In addition, word clouds will also be used to help visualise frequently used terms. While exploring spatial analysis techniques like in the study discussed above would provide further insight, location data is not available through Reddit.

3.8.3 Big data analytics and international negotiations: sentiment analysis of Brexit negotiating outcomes

This analysis used a sentiment analysis approach to tweets about the potential outcomes of the Brexit negotiations. The objectives were to conduct a sentiment analysis of tweets during a crucial phase (May 5th to November 7th, 2018) of Brexit negotiations, to decipher sentiments expressed on Twitter regarding Brexit outcomes, and to explore how real-time analysis of public opinion can be gauged for Governments to better understand the views of citizens.

The data collection methods for this study involved the use of a Python-based application using Twitter's API to collect and analyse real-time tweets related to Brexit negotiations, filtering tweets written in English containing relevant keywords and hashtags. This resulted in a dataset of 13,018,367 tweets collected.

The model for this analysis used a lexicon-based approach, which analyses texts and returns three polarity scores. These are positive polarity, negative polarity, and neutral polarity. In addition, a combination of the previous scores is returned, this has been normalised between -1 and one.

Approach of the analysis was like (Shu, et al., 2018) in that it deployed temporal sentiment analysis. This gave a sign of how sentiment results varied across hashtags over time, to give a view of change in public sentiment regarding certain topics relating to Brexit.

The study analysed user sentiment swings on Twitter during the Brexit negotiations between the EU and the UK. The results showed that user sentiment towards many Brexit outcomes was markedly neutral for a prolonged period, suggesting that sentiment towards Brexit was trending towards a numbing of emotional response. This was suggested to reflect the gradual realisation in the public and political space that Brexit is a highly technical and complicated matter.

The findings highlighted the potential of Twitter sentiment as a measure of public preferences, especially in complex situations like Brexit. It was proposed that citizen preferences be studied during the decision-making process of complex negotiations, creating a more inclusive approach to them, and that tracking Twitter sentiments during negotiations can offer valuable insights for decision-makers.

Project implications

While a lexicon-based approach was used, the use of normalising combined sentiment polarity scores to -1 and 1 will be deployed in this project. This provides an easy-to-understand scale of sentiment. This example, like the two examples discussed above, also supports the use of platform-specific features when displaying the results of the analysis. In addition to this, the study also looks at change in sentiment over time, which supports the use of a similar view of sentiment over time when this project conducts analysis.

3.9 Conclusion

This chapter has consisted of a literature review, providing context on key aspects of this project, with relevant sources for information. Topics of the discussion include:

- The scope and severity of cyber-attacks.
- Various challenges in NLP (Ghosh & Gunning, 2019), as well as relevant techniques in the pre-processing of text-based data.
- A background on large language models, and transformer architecture. The open-source Hugging Face platform as a resource for LLMs has also been discussed.
- Understanding the levels of sentiment analysis, the differences in the distinct types of text content as well as different approaches to performing sentiment analysis.
- Data collection, specifically an API-based approach has been discussed. The platform of choice, Reddit, has been justified as well as the influence Twitter's rate-limits had on that decision.
- In addition, example sentiment analyses have been discussed. The example analyses have provided a helpful reference point for how to communicate the results of a sentiment analysis, as well as how to conduct one.

In addition to these discussions, it has been detailed what the implications of this discussion are on the project. Going forward, decisions made in, as well as the approach to, the rest of this project will be aided by this literature review.

4 Methodology

4.1 Introduction

This chapter will provide an overview of the steps taken during the practical components of this project. Decisions made are influenced by the research conducted in the literature review in chapter 3. The content of this chapter will lead to completion of several objectives laid out in section 2.2.

4.2 Data Collection and storage

4.2.1 Selection Criteria and Justification

To select which cyber-attacks to analyse responses for, criteria for the selection process needs to be determined. The criteria are as follows:

- Time period:
    - The cyber-attack must ideally have been in the last 10 years. Due to the large rise in social media activity and usage in this time, this can lead to the access of far more data than with older attacks.
    - In addition, older social media posts are more likely to have been deleted, lost, or in a format that makes storage difficult or impossible.
- Scale of attack:
    - Attacks that have had a significant impact on many individuals or organisations are preferable for selection. These attacks are more likely to produce a substantial number of media coverage and social media posts, resulting in a larger amount of data to process for analysis.
- Industry diversity
    - Attacks chosen must either effect a diverse range of industries or be in an industry not represented already in the previously chosen attacks. A broader perspective on how people react to cyber threats can be produced from this.
- Location
    - Attacks affecting organisations from English speaking countries or targeting English speaking countries are preferred for selection. These attacks are likely to produce the most English-speaking discourse. The justification for this choice is the model to be selected for sentiment analysis will likely be trained on English language. This means that analysis on non-English discourse will not produce relevant results.
    - While language translation is a potential solution to this, this is not a flawless procedure, and the meaning of a sentence can be lost. This can also affect results, so language translation is ruled out as an option.

Cyber-attacks are to be chosen with the aim of each one meeting as much of the criteria as possible. Due to rate limits, there are limitations in amount of data that can be collected using APIs. This means the attacks chosen will be limited to three at most. This number was selected as it is a good compromise in maximising the amount of data that can be collected per attack, and different attacks to be compared.

4.2.2 Chosen Attacks

Insomniac Games Ransomware Attack (2023). On December 12, 2023, (Davies, 2023) a ransomware group named Rhysida posted 1.67 terabytes of data obtained from game developer Insomniac. The leak came after a demand made to the company of around £1,750,000 was not fulfilled.
Criteria met:

- Time period: This attack occurred within the last decade.
- Scale of attack: The leak of huge quantities of data and sensitive information garnered significant media coverage and attention.
- Industry diversity: Represents the video games industry, providing insights into the reactions and responses from stakeholders in this industry.
- Location: Discourse occurred globally, but primarily in English-speaking countries, providing English-language discourse for analysis.

SolarWinds Supply Chain Attack (2020). In 2020 (Oladimeji & Kerner, 2023), a sophisticated cyber-espionage campaign targeted the software supply chain of SolarWinds, a leading IT management software provider. The attackers compromised SolarWinds' software updates, distributing malware to many organizations globally, including government agencies and technology firms.
Criteria met:

- Time period: This attack occurred within the last decade.
- Scale of attack: The SolarWinds attack had a significant impact on a wide range of organizations, generating extensive media coverage and public discourse.
- Industry diversity: Represented diverse sectors due to SolarWinds' broad customer base, providing insights into various industries' responses to cyber threats.
- Location: Targeted organizations primarily in English-speaking countries, providing English-language discourse for analysis.

WannaCry Ransomware Attack (2017). The WannaCry (NHS England, 2023) ransomware attack, which occurred in May 2017, targeted computers running Microsoft Windows. WannaCry encrypted files and demanded ransom payments in Bitcoin from users. It had a global impact, affecting hundreds of thousands of computers worldwide and disrupting operations across various industries.
Criteria met:

- Time period: This attack occurred within the last decade.
- Scale of attack: WannaCry had a significant global impact, leading to widespread media coverage and attention.
- Industry diversity: Impacted various sectors, including healthcare, government, and finance, providing diverse perspectives for analysis.
- Location: Targeted organizations worldwide, providing diverse English-language discourse for analysis.

In addition to each attack meeting the selection criteria, these attacks have taken place in different years. This can add an extra element to the analysis as the reactions of users in different years can be compared.

4.2.3 Method of collection

To begin collecting data, and to assist in the database design process, it needs to be decided what is being collected. The data to be collected is as follows:

- For posts:
    - Post ID

    - The subreddit the post was posted in.
    - When a post was created (datetime)
    - Title of the post
    - Content of the post (the main body of text)
    - "Score" of a post (upvotes – downvotes)
    - Score ratio of post (% of voters selecting upvote)
- For comments
    - Comment ID
    - Content of the comment
    - "Score" of a post (upvotes – downvotes)
    - When a post was created (time and date)
- For replies to comments
    - This is the same as for comments.

The intended outcome of these rules is the minimising of the amount of data collected unnecessarily. Information that would be considered essential for analysis and data integrity is stored. For example, all text-based data has been kept, as well as information like post ids. Another example would be no user information being stored, as this will not be used in the analysis.

4.2.4 Database Design and Implementation

Upon identifying the entities involved in the data collection process, the following entity-relationship diagram (ERD) was produced, using the website Lucidchart (Lucidchart, n.d.):

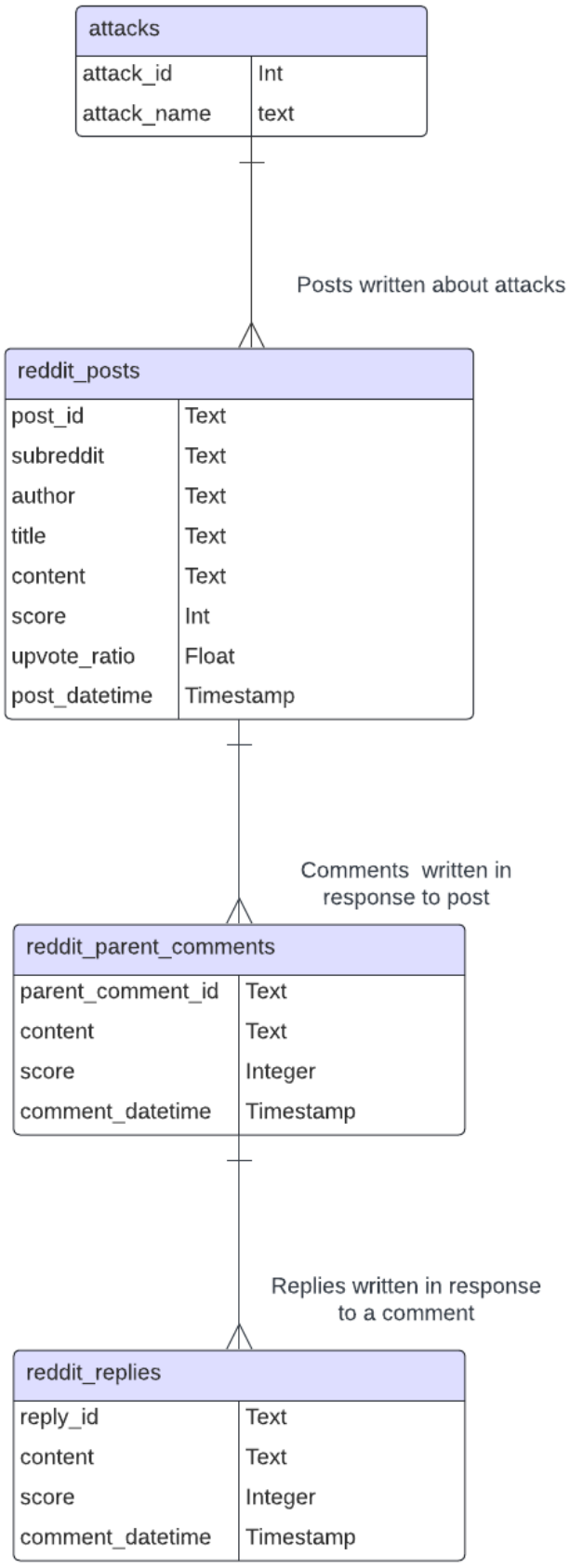


Figure 2: initial ERD for the database.

This figure provides the entities and their corresponding attributes. An “entity” is something with an independent existence, and an “attribute” is anything that provides information about that entity. The lines connecting entities show the relationship between them. For example, for the line connecting “reddit_posts” and “reddit_parent_comments”, this line indicates one reddit post can contain many comments.

However, the attributes of these entities have no information that could link them together. For example, there is no way to tell what comment a reply is linked to. The updated diagram is as such:

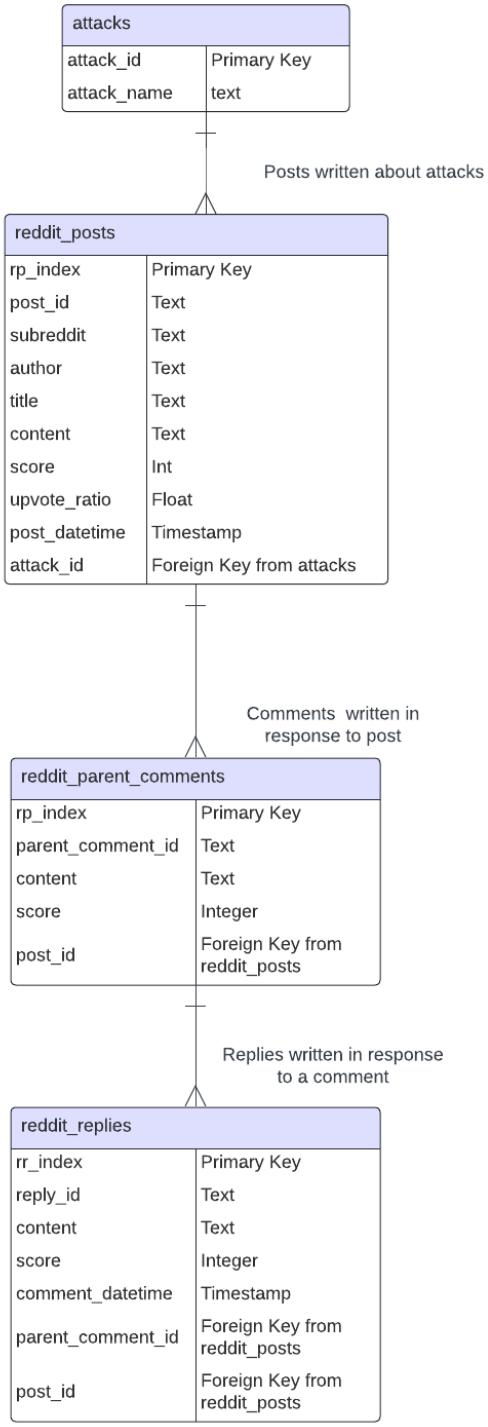


Figure 3: ERD after update to attributes.

The reasons for the changes are as such:

- To “attacks”:
    - “attack_id” is now a primary key, to help identify an attack when referenced in other tables.
- To “reddit_posts”:
    - There is now a primary key that identifies each row of the table.
    - “attack_id” is a foreign key referencing the “attack_id” of the attack a post is related to.
- To “reddit_parent_comments”:
    - There is now a primary key that identifies each row of the table.
    - “post_id” is a foreign key referencing the “post_id” of the entry in “reddit_posts” that a comment belongs to.
- To “reddit_replies”:
    - There is now a primary key that identifies each row of the table.
    - “parent_comment_id” is a foreign key referencing the “parent_comment_id” value of the entry in “reddit_parent_comments” the reply belongs to.
    - “post_id” is a foreign key referencing the “post_id” of the entry in “reddit_posts” that a reply belongs to.

This database meets the requirements for third normal form. Based on the ERD diagram, the following SQL file was created to set up the database tables:

```sql
CREATE TABLE IF NOT EXISTS attacks (
    attack_id   INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    attack_name TEXT    NOT NULL UNIQUE
);

CREATE TABLE IF NOT EXISTS reddit_posts (
    rp_index INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    post_id   TEXT  NOT NULL,
    subreddit TEXT,
    title     TEXT      NOT NULL,
    content   TEXT,
    score     INTEGER,
    upvote_ratio FLOAT,
    post_datetime TIMESTAMP,
    attack_id INTEGER,
    FOREIGN KEY (attack_id)
    REFERENCES attacks (attack_id)
);

CREATE TABLE IF NOT EXISTS reddit_parent_comments (
    rc_index INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    parent_comment_id TEXT NOT NULL,
    content TEXT,
    score INTEGER DEFAULT 0,
    comment_datetime TIMESTAMP NOT NUll,
    post_id TEXT NOT NULL,
    FOREIGN KEY (post_id) REFERENCES reddit_posts(post_id)
);

CREATE TABLE IF NOT EXISTS reddit_replies (
    rr_index INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    reply_id TEXT NOT NULL,
    content TEXT,
    score INTEGER,
    comment_datetime TIMESTAMP NOT NUll,
    parent_id TEXT NOT NULL,
    post_id   TEXT  NOT NULL,
    FOREIGN KEY (parent_id) REFERENCES reddit_parent_comments(parent_comment_id),
    FOREIGN KEY (post_id) REFERENCES reddit_posts(post_id)
);

INSERT INTO attacks (attack_name)
VALUES ("Insomniac Games"), ("WannaCry"), ("SolarWinds");
```

Figure 4: Create db.sql.

This file sets up the database tables, and automatically inserts the relevant cyber-attacks into the “attacks” table. As the database is now set up, data collection can begin.

4.2.5 Data Collection Implementation

To collect data, a Python file was created to connect to Reddit's API and store the relevant data for each cyber-attack. To begin with, a series of search terms was created for each attack, using the Boolean operators in Reddit's search features (Reddit Help, 2023).

```
queries_wannacry = ["WannaCry AND attack",
                    "WannaCry AND hack",
                    "WannaCry AND ransomware",
                    "WannaCry AND malware",
                    "WannaCry AND vulnerability",
                    "WannaCry AND exploit",
                    "WannaCry AND prevention",
                    "WannaCry AND mitigation",
                    "WannaCry AND impact",
                    "WannaCry AND consequences",
                    "WannaCry AND recovery",
                    "WannaCry AND remediation",
                    "WannaCry AND cybersecurity",
                    "WannaCry AND defense",
                    "WannaCry AND analysis",
                    "WannaCry AND assessment",
                    "WannaCry AND incident",
                    "WannaCry AND breach",
                    "WannaCry AND ransom",
                    "WannaCry AND ransomware",
                    "WannaCry cyber attack"]
```

Figure 5: An example of the search terms created for the WannaCry cyber-attack.

The figure above shows the search terms. These search terms were created with the intent of getting as much different data as possible. Normally, the structure of the search is "<attack name> AND <keyword>", allowing for variations in terminology to be caught when searching.

Then, connection was set up with Reddit.

```
# Establish connection to reddit
    try:
        reddit = praw.Reddit(
            client_id=reddit_client_id,
            client_secret=reddit_client_secret,
            user_agent=reddit_user_agent,
            username=reddit_username,
            password=reddit_password,
            redirect_uri = reddit_redirect_uri,
        )

        # Print information about the Reddit instance, to prove it worked.
        print(f"Reddit Instance Information:")
        print(f"Authenticated: {not reddit.read_only}")  # If True, it means the instance is read-only (non-authenticated)
        print(f"User: {reddit.user.me()}")  # Print the authenticated user's information

    except praw.exceptions.RedditAPIException as e:
        print(f"Authentication failed: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

Figure 6: Connecting to Reddit.

The figure above shows the connection process. The required information needed for Reddit to confirm connection is given, and if there is an error or connection is unsuccessful, the script returns the relevant information for this.

Then, functions for storing posts, comments, and replies were created.

```
def store_post(submission, attack_id):
    sql_query = "INSERT INTO reddit_posts(post_id, subreddit, title, content, score, upvote_ratio, post_datetime, attack_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"
    subreddit = submission.subreddit.display_name
    conn.execute(sql_query, (submission.id, subreddit, submission.title, submission.selftext, submission.score, submission.upvote_ratio, submission.created_utc, attack_id))
    conn.commit()

def store_comment(comment, post_id):
    sql_query = "INSERT INTO reddit_parent_comments(parent_comment_id, content, score, comment_datetime, post_id) VALUES (?, ?, ?, ?, ?)"
    conn.execute(sql_query, (comment.id, comment.body, comment.score, comment.created_utc, post_id))
    conn.commit()

def store_reply(reply, comment_id, post_id):
    sql_query = "INSERT INTO reddit_replies(reply_id, content, score, comment_datetime, parent_id, post_id) VALUES (?, ?, ?, ?, ?, ?)"
    conn.execute(sql_query, (reply.id, reply.body, reply.score, reply.created_utc, comment_id, post_id))
    conn.commit()
```

Figure 7: Functions for storing posts, comments, and replies.

The figure above shows these functions. The proper queries are created and executed using information from the entry itself, as well as any parameters passed to the function. An example of this being the function store_comment taking the ID of the post it belongs to as a parameter and storing it alongside all other relevant comment information.

Then, a function was created that iterates through the search terms and stores the information.

```
def search_for_information(reddit, attack_id, queries):
    search = reddit.subreddit("all")
    for search_query in queries:
        search_results = search.search(search_query, sort='relevance', limit = 50)
        # Search for posts based on the query
        for submission in search_results:
            print(f"Title: {submission.title}")
            # Store the post
            store_post(submission, attack_id)
            # Search for comments in the post
            submission.comments.replace_more(limit=None)
            for comment in submission.comments.list():
                # Store the comment
                store_comment(comment, submission.id)
                for reply in comment.replies.list():
                    #Store replies
                    store_reply(reply, comment.id, submission.id)
            print(f"Saved for submission {submission.id}")
        print(f"\nSaved query {search_query}")
        print("\n---\n")
    return
```

Figure 8: Search for information function.

The above figure shows this function. The function uses the search term on all subreddits, takes the first fifty posts returned, sorted by relevance, and stores all the posts, comments, and replies. The choice of fifty posts per search term was made as this struck a balance between not going over rate limits and maximising the amount returned per search term. The print statements throughout the function update on the progress of the search.

Throughout the collection process, Reddit would throw an error from the script being left running too long. This is due to making too many requests to the API in a specified time. Praw is configured to ensures that requests to Reddit's API have a wait time attached to them, to reduce the odds of going over rate limits. However, Reddit still threw these timeout

errors. As these rate limits reset after some time, a check was created to display how long until collection could continue:

```
#Set up search, and begin
   search_or_test = input("searching or testing? Default is testing ").lower()
   if(search_or_test == "search" or search_or_test == "searching"):
       attack = (input("Which attack is this? Insomniac(I), WannaCry(W), or Solarwinds(S)? ").lower())
       if (attack == "i" or attack == "insomniac"):
           search_for_information(reddit, 1, queries_insomniac)
           print("Saved Insomniac Queries")
       if (attack == "w" or attack == "wannacry"):
           search_for_information(reddit, 2, queries_wannacry)
           print("Saved WannaCry Queries")
       if (attack == "s" or attack == "solarwinds"):
           search_for_information(reddit, 3, queries_solarwinds)
           print("Saved SolarWinds Queries")
   else:
       # Get rate limit information
       rate_limit = reddit.auth.limits
       # Print rate limit information
       print(rate_limit)
       reset_timestamp = rate_limit['reset_timestamp']
       reset_datetime = datetime.datetime.fromtimestamp(reset_timestamp)
       # Print the reset datetime
       print("Reset datetime:", reset_datetime)
       print("Test done")
```

Figure 9: Checking how long until rate limits reset.

This part of the data collection script has two aspects to it.

- First, it checks whether the intent of using the script is to check rate limit usage or searching for data to collect.
    - Data collection was split into the different cyber-attacks to help ensure rate limit usage would not exceed limits.
- If the intent is to check rate limit usage, then the amount of time left until the rate limit usage resets is displayed, as well as the current usage in that time.

This meant that data could be collected while respecting the rate limits imposed by Reddit's API.

4.3 Data Preprocessing

A Jupyter Notebook was created to take the collected data through the appropriate pre-processing steps for the data. A Jupyter Notebook is a common data science tool, it allows for coding, data visualization, and documentation all in the same environment (Jupyter, n.d.). The steps of pre-processing were chosen due to the nature of the data being text-based, as well as collected online. The process is as follows:

- Loading the data for each attack, then for each attack:
    - Remove duplicate posts, comments, and replies.
    - Removing entries by date.
    - Removing off-topic entries.
- Merging the data, then:
    - Filtering the content by language.
    - Text cleaning and removing deleted.
    - Spelling Correction.
    - Lemmatization and stopword removal.
    - Saving the pre-processed data.

This set of steps ensures that no data is repeated, and as much meaning as possible is captured in the remaining data. These steps are based upon the research conducted earlier. The process for these steps will now be discussed.

4.3.1 Loading the data

Upon connecting with the database, the data had to be loaded. Initially, the data was loaded in the form of each attack being loaded individually. The function for this process is shown below.

```
def fetch_attack(attack_id: int) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = pd.read_sql(sql = f"select * from reddit_posts where attack_id = {attack_id}", con = conn, index_col = "rp_index")
    comments = pd.read_sql(sql = f"select * from reddit_parent_comments where post_id in"
                           f"(select post_id from reddit_posts where attack_id = {attack_id})", con = conn, index_col = "rc_index")
    replies = pd.read_sql(sql = f"select * from reddit_replies where parent_id in"
                          f"(select parent_comment_id from reddit_parent_comments where post_id in"
                          f"(select post_id from reddit_posts where attack_id = {attack_id}))",con = conn, index_col = "rr_index")
    return posts, comments, replies
```

Figure 10: fetch_attack - a function using SQL to load the tables based on attack_id.

This function uses nested SQL queries as well as the attack_id value of a post entry to load all posts, comments, and replies associated with a cyber-attack.

4.3.2 Duplicate removal

For each set of posts, comments, and replies associated with a cyber-attack, the Pandas library function drop_duplicates was applied (NumFOCUS, 2024). This allowed for quick and effective duplicate removal. This step was necessary due to the data collection script not skipping over content already accessed by the user, leaving the possibility of lots of repeated data. The parameter “subset” was utilised to ensure the function only checked the id of an entry (such as post_id for posts), to decide if it was a duplicate entry.

```
insomniac_posts = insomniac_posts.drop_duplicates(subset = 'post_id')
insomniac_parent_comments = insomniac_parent_comments.drop_duplicates(subset = 'parent_comment_id')
insomniac_reddit_replies = insomniac_reddit_replies.drop_duplicates(subset = 'reply_id')
```

Figure 11: using drop_duplicates.

4.3.3 Removing by date

For the Insomniac Games and WannaCry cyber-attacks, all data earlier than 2 weeks before the start of the cyber-attack was removed. 2 weeks was identified as a cut-off point as any data that has been captured from these 2 weeks periods can be used to find changes in sentiment during the first days of the public being aware of the attack. For the SolarWinds attack, which was discovered a lot later than when the attack took place, the cut-off point is 2 weeks before the discovery. The function remove_by_date was created for this. It is shown below.

```
def remove_by_date(cutoff_date: int, posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    # Filter posts based on the cutoff date
    posts = posts[posts['post_datetime'] >= cutoff_date]
    # Filter parent_comments based on the cutoff date
    parent_comments = parent_comments[parent_comments['comment_datetime'] >= cutoff_date]
    # Filter reddit_replies based on the cutoff date
    reddit_replies = reddit_replies[reddit_replies['comment_datetime'] >= cutoff_date]
    return posts, parent_comments, reddit_replies
```

Figure 12: remove_by_date.

4.3.4 Removing off-topic entries

As script for collecting data got the first fifty posts of a given search term, there was some content retrieved that was “off-topic”. Here, an off-topic entry in the data refers to posts that have been collected by the data collection script but are not actually relevant to a cyber-attack that has been selected here. A common example found when inspecting the database was a post that had the phrase “wanna cry” in the title, was not about the cyber-attack of similar name.

title	content
I wanna cry	I was playing KC and I've been through nonstop series of losses I was about to rank down if I had lost one mor...

Figure 13: an example of off-topic entries, taken from SQLite Studio.

Due to all comments of these posts being collected, as well as the content of the post itself, it became necessary to find all these posts and remove them. Leaving this data in would result in a lot of unrelated data in the results of the analysis.

The process of using this involves an LLM, namely Facebook’s “bart-large-mnli”, this is a form of bart-large (Lewis, et al., 2019) that is trained on the MultiNLI (MNLI) dataset. This model is suitable for carrying out Zero-Shot Classification (Hugging Face, n.d.). This is a natural language processing task where a model can recognise classes of data that it has not been explicitly trained on. Therefore, this is suitable for the task of finding off-topic data. The figure below shows the process of loading the model for the task.

```
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
```

Figure 14: Using hugging face's pipeline to access LLM for off-topic-removal.

The model will be given a series of classes to classify post titles into, and a score will be returned for each class. This score is the likelihood that the post is in the topics. Posts and their content will be removed based on the score the title receives. Through testing of this process, a cut-off score of 0.65 was developed for the most-likely class. This means that if the post is given a score below 0.65 for all classes, it is removed. This number gives the best balance between removing off-topic posts, and not being too-high a threshold for a score. In higher cut-off scores, it was often seen that on-topic posts would be removed. The figure below shows this process of finding the off-topic posts.

```
def get_off_topic(titles: list, labels: list) -> list:
    classifier_outputs = []
    for title in titles:
        sequence_to_classify=title
        classifier_outputs.append(classifier(sequence_to_classify, labels, multi_label = True))
    off_topic = []
    for index, output in enumerate(classifier_outputs):
        if max(output['scores']) <= 0.65: # Scores cut off point.
            off_topic.append({'index': index, 'title': output['sequence']})
    return off_topic
```

Figure 15: The get_off_topic function, which identifies these off-topic posts.

The figure below shows the topics used for finding off-topic posts for the Insomniac Games cyber-attack.

```
insomniac_labels = ["Insomniac leak", "Wolverine leak", "Spider-Man leak", "Insomniac cyber attack", "Insomniac ransomware", "Insomniac hack", "Insomniac stolen information"]
```

Figure 16: example of classes for the zero-shot classification.

The figure below shows the function remove_off_topic, which takes the list of identified off-topic posts and removes them as well as their related comments and replies.

```
def remove_off_topic(off_topic: list, posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    post_ids = []
    # Iterate through each title and remove accordingly
    for i in off_topic:
        posts_to_remove = posts[posts['title'] == i['title']]
        post_ids.extend(np.unique([posts_to_remove["post_id"]]))
    posts = posts[~posts['post_id'].isin(post_ids)]
    parent_comments = parent_comments[~parent_comments['post_id'].isin(post_ids)]
    reddit_replies = reddit_replies[~reddit_replies['post_id'].isin(post_ids)]
    return posts, parent_comments, reddit_replies
```

Figure 17: remove_off_topic, a function that takes the list of identified "off-topic" posts and removes them.

The above steps were repeated for the other two cyber-attacks. While removing duplicates could have been performed later in the process, removing them before finding off topic posts reduces processing time due to reduced data. Despite this, finding off topic posts was still a lengthy process.

4.3.5 Merging the data

As there are no longer attack-specific steps to perform, the data can now be merged. The figure below shows the function used for this. It was applied to form merged posts, merged comments, and merged replies.

```
def merge_datasets(insomniac_df: pd.DataFrame, wannacry_df: pd.DataFrame, solarwinds_df: pd.DataFrame) -> pd.DataFrame:
    merged_df = pd.concat(objs = [insomniac_df, wannacry_df, solarwinds_df])
    return merged_df
```

Figure 18: merge_datasets function.

The next steps discussed are applied to the comments and replies of posts.

4.3.6 Filtering non-English content

This makes use of the detect function from the langdetect library. The function filter_english was created. This function marks the language of the title of any posts that are not English. In addition, comments and replies that are not English are reduced to an empty string of “”. This is so they can be removed later. The figure below shows this.

```python
def filter_english(df, text_column = "content", posts = False):
    for index, row in df.iterrows():
        text = row[text_column]
        text = str(text)
        try:
            language = detect(text)
            # Handle posts
            if posts == True and language != 'en':
                row[text_column] = f"language detected: {language} for: {text}"
                # Keep post anyway
            # Check if language is English
            elif language != 'en':
                row[text_column] = ''
        except:
            pass  # Skip rows where language detection fails
    return df
```

Figure 19: filter_english function.

4.3.7 Text cleaning

Following the merging of data, the text content was cleaned. This involved:

- If the text is “[removed]”, “[deleted]”, or “” then the text is converted into “”
    - “[removed]” and “[deleted]” are common pieces of text in reddit data, these are to be reduced to an empty string as they are meaningless and represent when an entry has been deleted by a Reddit user or removed by a subreddit moderator. Later in the process, comments and replies with “” as their content will be removed.
- Next, the following is removed from text:
    - HTML tags
    - Common text markup, like markdown for italics or bold
    - New line tags
    - Links
    - User references
    - Subreddit references
    - Extra whitespace

Emojis and emoticons were kept in the text as in social media these are commonly used, and therefore can have an influence on a sentence’s meaning.

The figure below shows the function that performs this process.

```python
def clean_text(text: str) -> str:
    # Convert text to string
    text = str(text)
    # Deleted/Removed content, content marked for deletion
    if text in ['[removed]', '[deleted]']:
        text = ''
    # Remove HTML tags
    cleaned_text = re.sub(r'<[^>]+>', '', text)
    # Remove Reddit-specific markup
    cleaned_text = re.sub(r'(\*|_)(.*?)\1', r'\2', cleaned_text)  # Markdown for italics
    cleaned_text = re.sub(r'\*\*(.*?)\*\*', r'\1', cleaned_text)   # Markdown for bold
    cleaned_text = re.sub(r'\[([^]]+)\]\(([^)]+)\)', r'\1', cleaned_text)  # Markdown for hyperlinks
    cleaned_text = re.sub(r'> (.*?)\n', r'\1', cleaned_text)       # Markdown for quoting text
    # Remove new lines
    cleaned_text = cleaned_text.replace('\n', ' ')
    # Remove links
    cleaned_text = re.sub(r'https?://\S+|www\.\S+', '', cleaned_text)
    # Remove Reddit user mentions
    cleaned_text = re.sub(r'/u/\w+', '', cleaned_text)
    # Remove subreddit references
    cleaned_text = re.sub(r'/r/\w+', '', cleaned_text)
    # Remove emojis and emoticons
    # Remove extra whitespace
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text.strip()
```

Figure 20: clean_text function.

4.3.8 Spelling Correction

The function autocorrect_text was created. This function makes use of the autocorrect library. As certain comments are quite large, this can lead to large inputs. Therefore, the function chunk_text was also created which split inputs up into chunks to be processed one at a time. In addition, the fast parameter was set to true for the speller. Both these additions ensured completion of the autocorrect in a reasonable time period. These functions are shown below.

```
def chunk_text(text, max_words_per_chunk=20):
    # Split data up into chunks of 20
    words = text.split()
    chunks = []
    current_chunk = []
    for word in words:
        current_chunk.append(word)
        if len(current_chunk) >= max_words_per_chunk:
            chunks.append(" ".join(current_chunk))
            current_chunk = []
    if current_chunk:
        chunks.append(" ".join(current_chunk))

    return chunks

spell = Speller(fast = True)
def autocorrect_text(text):
    # Without this setting, it would not complete in a reasonable timeframe
    # Split the text into smaller chunks
    chunks = chunk_text(text)
    # Process each chunk separately
    corrected_chunks = []
    for chunk in chunks:
        corrected_chunks.append(str(spell(chunk)))
    # Combine the corrected chunks into a single string
    corrected_text = " ".join(corrected_chunks)
    return corrected_text
```

Figure 21: The chunk_text and autocorrect functions.

4.3.9 Lemmatization and stopword removal

Finally, lemmatization and stopword removal was applied. This process involved splitting the text into tokens, then performing these steps. As certain stopwords may affect the meaning of a sentence, these were removed from the list of stopwords. An example would be the word “not”, which could invert sentiment entirely. Below is the list of removed stopwords.

```
# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
nltk_stopwords = set(stopwords.words('english'))

# https://gist.github.com/sebleier/554280 - source of list of stopwords
removed_stopwords = ["all", "any", "both", "each", "few", "more", "most",
                     "other", "some", "such", "only", "own", "same", "too",
                     "very", "s", "t", "just", "now", "not", "in"]

for word in removed_stopwords:
        if word in nltk_stopwords:
            nltk_stopwords.remove(word)
```

Figure 22: Removing stopwords and initialising the lemmatizer.

The function lemmatise_remove_stopwords splits a given input into tokens, lemmatizes the tokens, then removes all stopwords found. The NLTK library was used for this. In addition to

this, particularly large inputs were reduced so the later sentiment analysis model can process all inputs. The tokens for an input were merged back into one piece of text, to ensure compatibility with the model deployed later.

```
def lemmatise_remove_stopwords(text:str) -> str:
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Lemmatize and remove stopwords
    processed_tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens if token.lower() not in stop_words and token.isalnum()]
    # Help the model handle larger inputs
    if len(processed_tokens) > 512:
        # Truncate the tokens list to contain only the first 512 tokens
        processed_tokens = processed_tokens[:512]
    # Concatenate tokens into a string
    processed_text = ' '.join(processed_tokens)
    return processed_text
```

Figure 23: the lemmatise_remove_stopwords function.

4.3.10 Saving Pre-Processed Data

Finally, the pre-processed text data was saved in a new SQL database. An example of this is shown below.

```
schema = {
    'rp_index': 'INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE',
    'post_id': 'TEXT NOT NULL',
    'subreddit': 'TEXT',
    'title': 'TEXT NOT NULL',
    'content': 'TEXT',
    'score': 'INTEGER DEFAULT 0',
    'upvote_ratio': 'FLOAT',
    'post_datetime': 'TIMESTAMP',
    'attack_id': 'INTEGER',
    'FOREIGN KEY (attack_id)': 'REFERENCES attacks(attack_id)'
}
merged_posts.to_sql(name = "reddit_posts", con = conn, if_exists = "replace", dtype = schema)
```

Figure 24: Saving the pre-processed posts.

4.4 Analysis Requirements and Model Selection

4.4.1 Analysis Requirements

This sentiment analysis, as discussed in chapter 3, will use a statistical-methods based approach. A hugging face transformer pre-trained for sentiment analysis will be used. Each comment and reply in the data collected will be processed, and the model will return the following:

- The determined sentiment class (positive, negative, and neutral)
- A “sentiment score” derived from the likelihood of an input being positive, negative, or neutral. These scores will fall in the range of -1 (negative) to 1 (positive).

The use of both a determined sentiment class and a sentiment score will result in more refined capturing of sentiment, as there may be many cases where a piece of text has similar odds of being one of two classes, so assigning one class to the text would not be sufficient.

Using the determined sentiment classes and scores, it will be determined:

- The distribution of sentiment class and score for each cyber-attack.
- The change in sentiment online during the first few months of each cyber-attack
    - How the sentiment changes in this period for the most active subreddits discussing each attack.
- The relationship between the sentiment of a comment and the upvotes (score) it receives.

Visualisations will be used when appropriate for conveying this information. Key dates for each attack will be highlighted when appropriate. The aim of these requirements is to provide a comprehensive overview of how social media users have reacted to the cyber-attacks chosen.

4.4.2 Model Requirements

Based on the requirements for the analysis, the model used for the sentiment analysis has the following requirements:

- The model must be suitable for the task of sentiment analysis.
- The model preferably should be pre-trained.
- The model should return a determined sentiment class (positive, negative, neutral)
- The model's outputs can be used for calculating an overall sentiment score.
- The model must be an LLM.

Based on the requirements, the hugging face transformer "cardiffnlp/twitter-roberta-base-sentiment-latest" (Camacho-Collados, et al., 2022) was found for use. This meets all requirements set out above, and in addition to this, it is trained on social media data. Being trained on social media data suggests it should perform better on the data collected than models not trained this way would.

A Jupyter Notebook was created for this analysis, as well as a Python file holding all functions created for this process. For the below figures of example code, please note that the file was imported as "f".

4.4.3 Evaluating the performance of the chosen model

Before deploying the model on the collected data, the performance of the model must be evaluated. The dataset of choice must be a dataset the model was not previously trained on, to ensure that the model can perform well on previously unseen data. The evaluation metrics will be:

- Accuracy score – the proportion of correctly classified instances across all classes.
- Precision score – the ability of the model to correctly identify instances of a specific class among all instances classified as that class. This is averaged for all classes.
- Recall score – the ability of the model to capture all instances of a specific class. This is the ratio of true positives to the total amount of true positives and false negatives. This is averaged for all classes.
- F1 score – a measure of a model's performance across all classes, considering both precision and recall.
- Confusion matrix - a visual breakdown of performance across all classes. This helps show which classes are being confused with each other.

These metrics are commonly used when evaluating the performance of an AI model, as discussed in section 3.8. The dataset “sentiment140” was identified for model testing. This dataset consists of tweets with labelled sentiment classes. Due to the social media nature of the text content to be analysed in this dataset, it was considered appropriately similar enough to the data collected for this analysis. Therefore, the scores returned for the above metrics should give a realistic sign of the accuracy of the later analysis results of the data collected.

The figure below shows loading the model.

```
SENTIMENT_MODEL = "cardiffnlp/twitter-roberta-base-sentiment-latest"
sentiment_tokenizer = AutoTokenizer.from_pretrained(SENTIMENT_MODEL)
sentiment_config = AutoConfig.from_pretrained(SENTIMENT_MODEL)
sentiment_model = AutoModelForSequenceClassification.from_pretrained(SENTIMENT_MODEL)
sentiment_model.eval()
```

Figure 25: loading the sentiment model.

Before evaluating the model, the dataset first had to be loaded and pre-processed.

```
# Load the Sentiment140 dataset
dataset = load_dataset("sentiment140", split = 'test')
dataset = f.preprocess_test_dataset(dataset)
```

Figure 26: loading the dataset.

The function preprocess_test_dataset takes the data through the same preprocessing steps as discussed earlier, except the functions for this have been modified for the given dataset. In addition to this, there is a new function, remove_blank, for this preprocessing that removes empty rows (“”). It is shown below.

```
def remove_blank(dataset, text_column="text"):
    # Filter out examples with empty text
    dataset = dataset.filter(lambda row: row[text_column] != '')
    return dataset
```

Figure 27: remove_blank.

Next, the labels assigned to rows in the test dataset are updated. These are labels like “0” to represent negative sentiment, and “4” to represent positive. This is so the outputs of the LLM used are matching to the labels of the dataset. The LLM is configured for the labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive. This process is shown below.

```python
test_texts = dataset["text"]
test_labels = dataset["sentiment"]
# Map sentiment labels from Sentiment140 dataset to match model output classes
def map_sentiment_label(label):
    if label == 0:  # Negative sentiment
        return 0
    elif label == 2:  # Neutral sentiment
        return 1
    elif label == 4:  # Positive sentiment
        return 2
    else:
        raise ValueError("Invalid sentiment label")

# Preprocess sentiment labels in the dataset
test_labels_mapped = [map_sentiment_label(label) for label in test_labels]
```

Figure 28: Matching the test labels to the LLM's labels.

Finally, the inputs are tokenized, and the model processes them. In addition to this, the predicted labels are extracted from the outputs. This is shown below.

```python
# Tokenize the validation or test data
inputs = sentiment_tokenizer(test_texts, return_tensors="pt", padding=True, truncation=True, max_length = 512)
# Forward pass through the model to obtain predictions for validation data
with torch.no_grad():
    outputs = sentiment_model(**inputs)
    logits = outputs.logits
# Convert logits to predicted labels for validation data
pred_labels = torch.argmax(logits, dim=1).tolist()
```

Figure 29: Inputting the test data.

Upon the model fully processing the test data, the scores were evaluated. The function evaluate_test_dataset calculates the metrics discussed earlier by using the outputs of the model, and the actual values of the test dataset. This function is shown below.

```python
def evaluate_test_dataset(test_labels_mapped, pred_labels):
   # Calculate evaluation metrics for validation data
    accuracy = round(accuracy_score(test_labels_mapped, pred_labels), 5)
    precision = round(precision_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    recall = round(recall_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    f1 = round(f1_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    headers = ['Metric', 'Value']
    scores_table = [
        ("Accuracy", accuracy),
        ("Precision", precision),
        ("Recall", recall),
        ("F1-score", f1)
    ]
     # Display the results in a table using Matplotlib
    fig, ax = plt.subplots(figsize=(5, 4))
    ax.axis('off')  # Turn off axis
    table = ax.table(cellText=scores_table, colLabels=headers, loc='center', cellLoc='left', cellColours=[['#f2f2f2', '#f2f2f2']]*len(scores_table))
    table.auto_set_font_size(False)
    table.set_fontsize(12)
    table.scale(1, 1.5)  # Scale table to fit better
    table.auto_set_column_width([0, 1])  # Adjust column width automatically
    for key, cell in table.get_celld().items():
        cell.set_linewidth(0.5)  # Set cell border width
    plt.title('Evaluation Metrics:')
    plt.show()
    ConfusionMatrixDisplay.from_predictions(test_labels_mapped, pred_labels)
    plt.show()
```

Figure 30: evaluate_test_dataset.

The outputs were as such:

Evaluation Metrics:

Metric	Value
Accuracy	0.86747
Precision	0.87243
Recall	0.86747
F1-score	0.86777

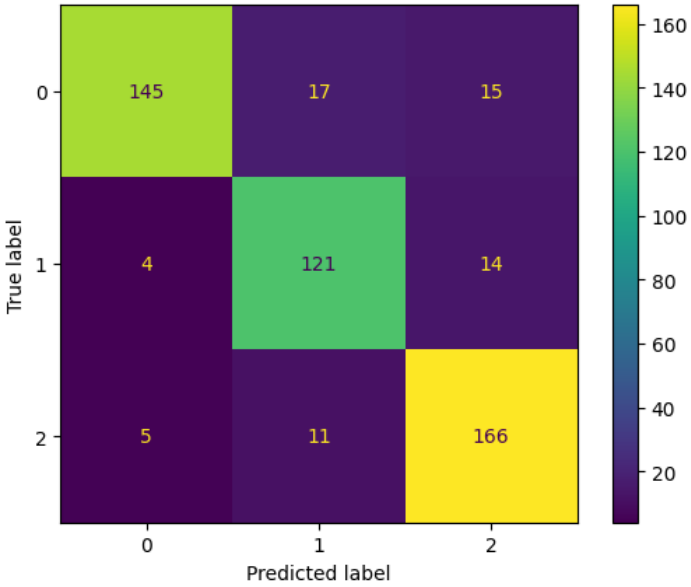


Figure 31: Table and confusion matrix for model evaluation.

Based on this performance, it can be concluded the model performs well on previously unseen social media data. The model can correctly identify the sentiment of a given piece of social media text, with a low rate of error. This suggests when the model is deployed on the collected data, the results will be reliable. Due to these high scores, the model can now be deployed on the collected data.

4.5 Deploying the model

4.5.1 Fetching the data and final preparation

First, the pre-processed data must be loaded. Connection is set up with the SQL database. The function get_all_attacks performs this task.

```
def get_all_attacks(conn) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = pd.read_sql(sql = f"select * from reddit_posts", con = conn)
    comments = pd.read_sql(sql = f"select * from reddit_parent_comments", con = conn)
    replies = pd.read_sql(sql = f"select * from reddit_replies", con = conn)
    posts.set_index(posts.columns[0], inplace=True, drop=True)
    comments.set_index(comments.columns[0], inplace=True, drop=True)
    replies.set_index(replies.columns[0], inplace=True, drop=True)
    return posts, comments, replies
```

Figure 32: get_all_attacks.

Then, the data is split up into individual attacks. The function get_attack_from_merged performs this, as shown below.

```
def get_attack_from_merged(posts_df: pd.DataFrame, comments_df: pd.DataFrame, replies_df: pd.DataFrame,attack_id: int) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = posts_df[posts_df['attack_id'] == attack_id]
    comments = comments_df[comments_df['post_id'].isin(posts['post_id'])]
    replies = replies_df[replies_df['post_id'].isin(posts['post_id'])]
    return posts, comments, replies
```

Figure 33: get_attack_from_merged.

Finally, the comments and replies DataFrame for each attack has the final preparation steps applied to them. These steps are to ensure that all required information for analysis can be accessed in one location. Based on this, as well as the requirements for the analysis discussed earlier, a function was needed that:

- Returns a singular DataFrame, of all comments and replies for a singular cyber-attack.
- Labels each comment with the subreddit the comment belongs to, as well as the id of the attack the comment is about.
- Removes all meaningless entries.
    - These would be comments with an empty text body of “”, typically due to the preprocessing steps.
- Removes further duplicates that may have occurred because of this process.

The result of these requirements was the function get_all_comments_df. Which used posts, parent comments and replies of a cyber-attack to create a single DataFrame with all this information. These comments will be used as the inputs for the model. The function is shown below.

```
def get_all_comments_df(posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> pd.DataFrame:
    # Merge dfs
    comments_df = pd.concat([parent_comments, reddit_replies], ignore_index=False)
    # Handle empty values in p_c_id (from replies)
    comments_df['parent_comment_id'] = comments_df['parent_comment_id'].fillna(comments_df['reply_id'])
    # Rename p_c_id, remove now irrelevant comments
    comments_df.rename(columns={'parent_comment_id': 'comment_id'}, inplace=True)
    comments_df.drop(columns=['reply_id'], inplace=True)
    comments_df.drop(columns=['parent_id'], inplace=True)
    # Reset the index of the DataFrame
    comments_df.reset_index(drop=True, inplace=True)
    # Add relevant subreddit values
    post_subreddit_dict = posts.set_index('post_id')['subreddit'].to_dict()
    comments_df['subreddit'] = comments_df['post_id'].map(post_subreddit_dict)
    # Do the same for attack id
    post_attack_dict = posts.set_index('post_id')['attack_id'].to_dict()
    comments_df['attack_id'] = comments_df['post_id'].map(post_attack_dict)
    # Remove empty comments
    comments_df = comments_df[comments_df['content'] != '']
    # Ensure no duplicates
    comments_df = comments_df.drop_duplicates(subset = 'comment_id')
    return comments_df
```

An example of applying this function is shown below, as well as the output from this.

Insomniac Games

```
insomniac_all_comments = f.get_all_comments_df(insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies)
insomniac_all_comments
```

[249]

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id
0	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5	TwoBestFriendsPlay	1
1	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5	TwoBestFriendsPlay	1
2	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5	TwoBestFriendsPlay	1
3	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5	TwoBestFriendsPlay	1
4	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5	TwoBestFriendsPlay	1
...	...	...	...	...	...	...	...
16819	j6rgouh	never said anything thousand breach successful...	1	1675249670	10qf3j1	Superstonk	1
16820	j6s6w37	even page admitting breach	1	1675263624	10qf3j1	Superstonk	1
16821	j6s0uzo	theyre manually processing trade yes probably ...	2	1675261129	10qf3j1	Superstonk	1
16822	j6s1lmc	wow much impact huge	2	1675261444	10qf3j1	Superstonk	1
16823	j6r9s8c	admitted	1	1675243937	10qf3j1	Superstonk	1

16085 rows × 7 columns

Figure 34: applying get_all_comments_df.

Finally, the data is ready to be processed by the chosen LLM.

4.5.2 Processing data

To process the data, two functions, process_df and process_comment_sentiment, were made. process_comment_sentiment is shown below. This function:

- Tokenises a comment using the model's tokenizer. This results in vector-based representation of the text. This is then input into the model.
- Then, using the outputs from this, extracts the scores assigned to each class. These scores are a percentage of the likelihood a given comment is positive, neutral, or negative.
- Using these scores, an overall sentiment score is calculated, then the determined sentiment class is assigned to a comment based on the sentiment class with the highest likelihood.
- Then, this sentiment score and determined sentiment class is returned as a DataFrame.

```
def process_comment_sentiment(text, sentiment_tokenizer, sentiment_config, sentiment_model) -> pd.DataFrame:
    # Tokenize input to the format suitable for the model
    encoded_input = sentiment_tokenizer(text, return_tensors='pt', padding=True, truncation=True, max_length=512)
    # Generate predictions
    with torch.no_grad():
        output = sentiment_model(**encoded_input)
    # Process scores
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    # Extract positive, neutral, and negative scores
    positive_score = scores[sentiment_config.label2id['positive']]
    neutral_score = scores[sentiment_config.label2id['neutral']]
    negative_score = scores[sentiment_config.label2id['negative']]
    # Calculate overall sentiment score
    overall_score = (1 * positive_score + 0 * neutral_score - 1 * negative_score)
    # Determine class with highest score
    classes = ['positive', 'neutral', 'negative']
    max_class_index = np.argmax([positive_score, neutral_score, negative_score])
    determined_sentiment = classes[max_class_index]
    # Create DataFrame
    data = {'sentiment_score': overall_score, 'determined_sentiment': determined_sentiment}
    df = pd.DataFrame([data])
    return df
```

Figure 35: process_comment_sentiment.

The function process_df applies the process_comment_sentiment function to every row of a DataFrame, appending the scores for each row as it goes along. When all comments have been processed, the new DataFrame of all comments and their scores is returned.

```
def process_df(df:pd.DataFrame, column:str, sentiment_tokenizer, sentiment_config, sentiment_model)-> pd.DataFrame:
    result_df = pd.DataFrame()
    # Iterate over each row
    for index, row in df.iterrows():
        # Get sentiment scores for the content in the current row
        sentiment_scores = process_comment_sentiment(row[f'{column}'], sentiment_tokenizer, sentiment_config, sentiment_model)
        # Append the sentiment scores to the result DataFrame
        result_df = pd.concat(objs = [result_df, sentiment_scores], ignore_index=True)
    result_df.index = df.index
    df = df.join(result_df)
    return df
```

Figure 36: process_df.

These functions were applied individually to each DataFrame, one for every cyber-attack. The figure below shows an example of this application to the data collected for the Insomniac Games cyber-attack.

```
insomniac_all_comments = f.process_df(insomniac_all_comments, 'content', sentiment_tokenizer, sentiment_config, sentiment_model)
```

Figure 37: Applying process_df.

Then the results were joined together. This was done as when analysing the results, it is easy to access an attack individually, or the results for all attacks at once for comparison purposes. The figure below shows the creation of this joined DataFrame, as well as some example rows.

```
all_comments = pd.concat([insomniac_all_comments, wannacry_all_comments, solarwinds_all_comments])
all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id	sentiment_score	determined_sentiment
0	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5	TwoBestFriendsPlay	1	-0.885556	negative
1	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5	TwoBestFriendsPlay	1	-0.840470	negative
2	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5	TwoBestFriendsPlay	1	-0.289011	neutral
3	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5	TwoBestFriendsPlay	1	-0.461533	negative
4	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5	TwoBestFriendsPlay	1	-0.553312	negative
...	...	...	...	...	...	...	...	...	...
16624	gsow4wk	shocked well really shocked	6	1617028497	mfhuq7	technology	3	-0.840873	negative
16625	gsq4dj7	started war operation olympic game look stuxne...	1	1617048746	mfhuq7	technology	3	-0.234056	neutral
16626	gspo05b	true blessed incredibly ineffective incapable ...	1	1617041266	mfhuq7	technology	3	-0.904313	negative
16627	gsqgnwd	biden done anything	3	1617054551	mfhuq7	technology	3	-0.025591	neutral
16628	gsx6aiw	yup military congress openly advocating manhat...	1	1617202880	mfhuq7	technology	3	-0.243846	neutral

38957 rows × 9 columns

Figure 38: creating all_ comments, and example rows.

Finally, data has been created that matches the requirements of the sentiment analysis as discussed above. Results will be shown and discussed in chapter 5.

4.6 Conclusion

This chapter has provided a comprehensive overview of the steps taken to collect and store data, as well as the approach taken to data pre-processing. Furthermore, the sentiment analysis had requirements defined, and an appropriate LLM has been selected to perform the task.

5 Results

5.1 Introduction

In this section, the results of the sentiment analysis will be discussed. The topics of discussion will be influenced by the sentiment analysis requirements discussed in section 4.3.

5.2 Sentiment distributions

The distributions of sentiment score, as well as the determined sentiment class for each cyber-attack are:

- Insomniac Games

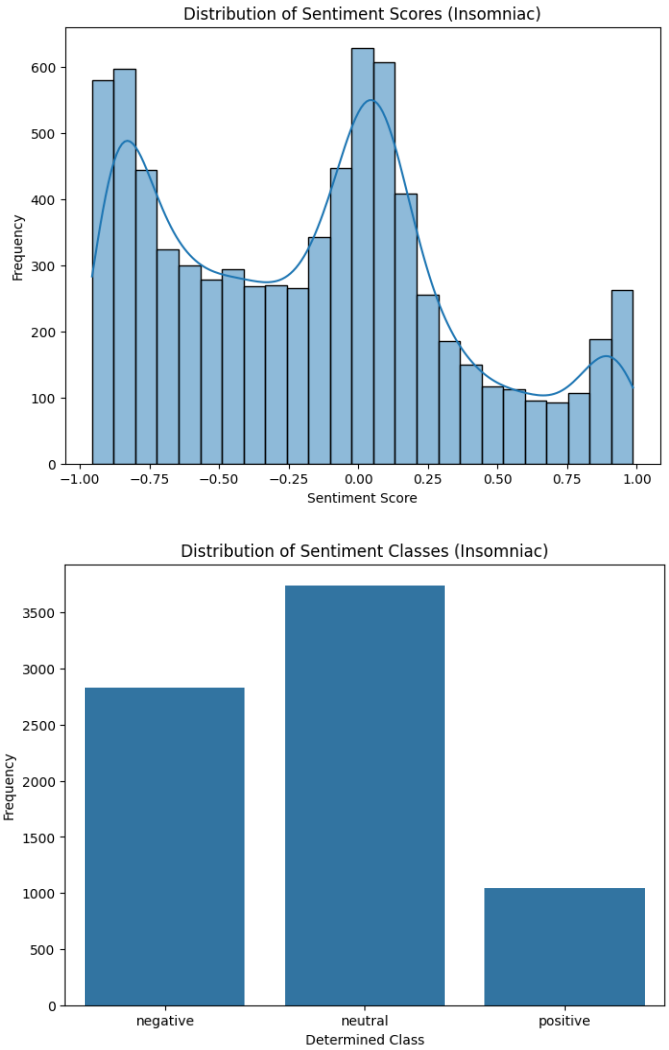


Figure 39: The distribution of sentiment scores and classes for Insomniac Games

- WannaCry

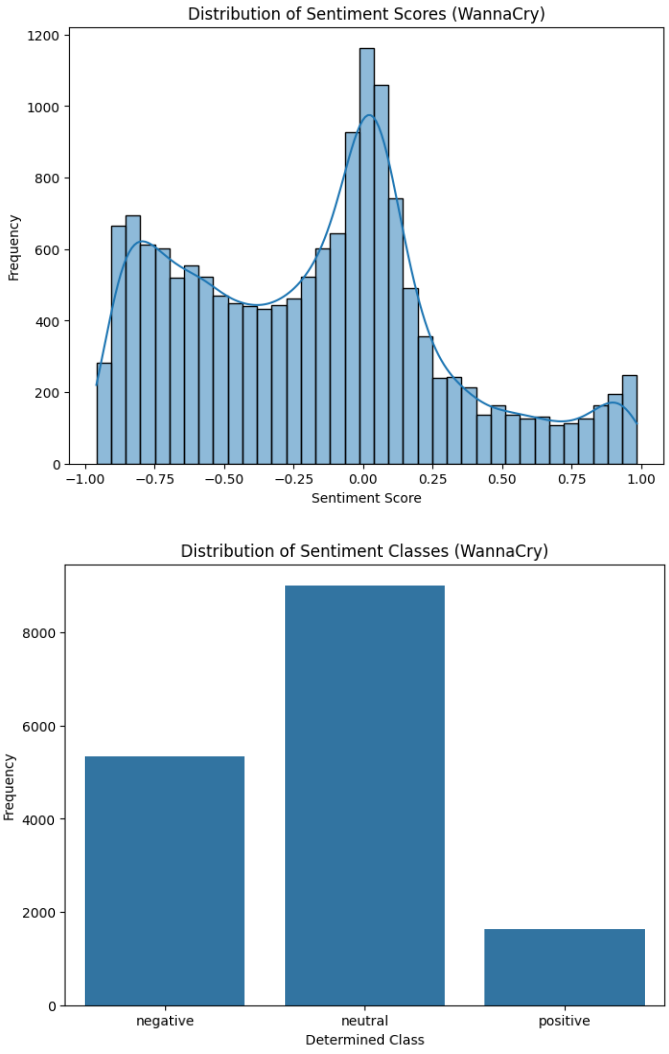


Figure 40: The distribution of sentiment scores and classes for WannaCry.

- SolarWinds

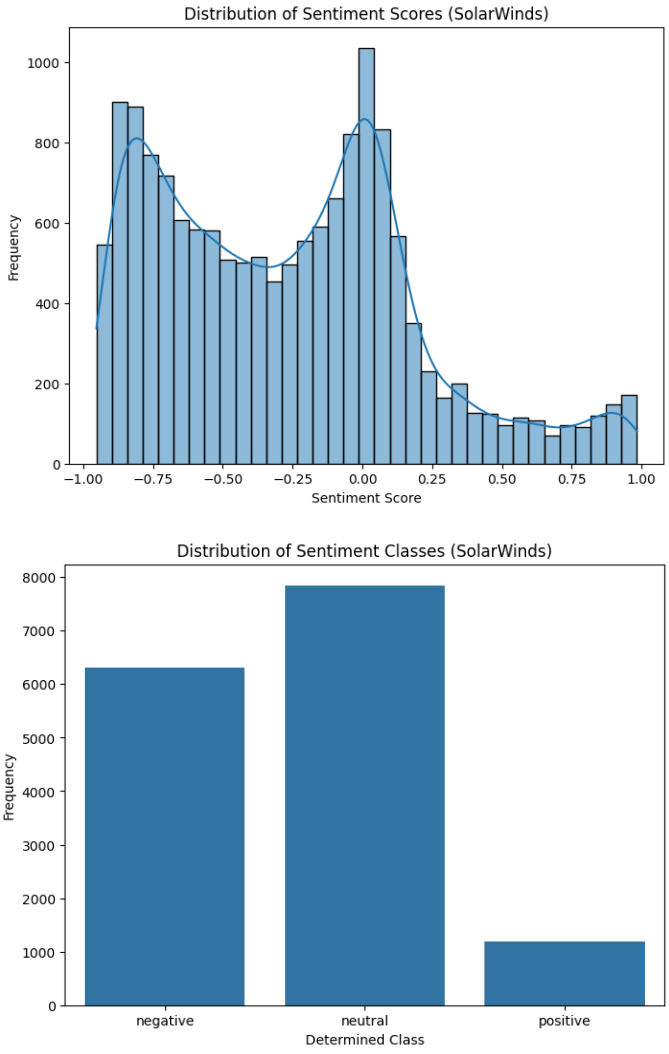


Figure 41: The distribution of sentiment scores and classes for SolarWinds.

- For all attacks combined:

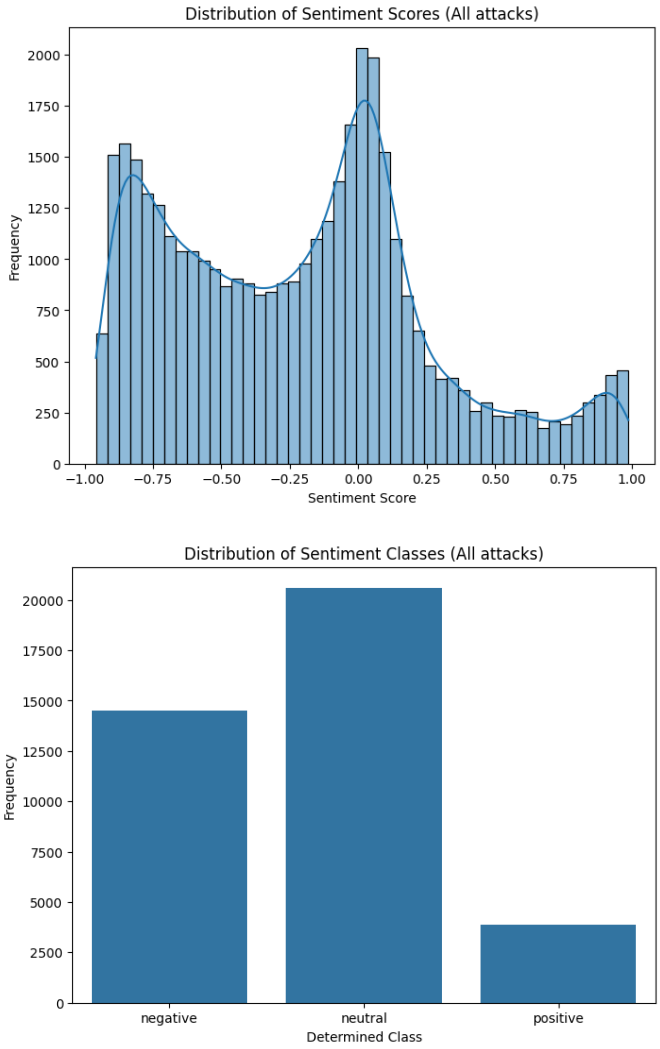


Figure 42: The distribution of sentiment scores and classes across all attacks.

Based on these results, it is clear most conversation is either of a neutral or negative sentiment. Neutral classes are the dominant assigned class of comment. Based on the distribution of sentiment scores, most conversation falls in the sentiment range of around -1 to +0.25, with most scores falling below 0. There is no outlier attack in terms of sentiment scores or distribution, with all attacks following similar distributions of sentiment.

5.3 The change in sentiment over time

Next, the change in sentiment over time will be examined for each cyber-attack. This will involve, for each cyber-attack:

- The first 2 months since the attack's discovery will be shown.
- The average sentiment scores over this period will be displayed for the top ten most active subreddits.
- To provide context for any changes in sentiment, key dates will be displayed.

Then, all cyber-attacks will be compared together, without the use of subreddits or key dates.

For Insomniac Games, with key dates provided by (Davies, 2023) and (Shirey, 2023):

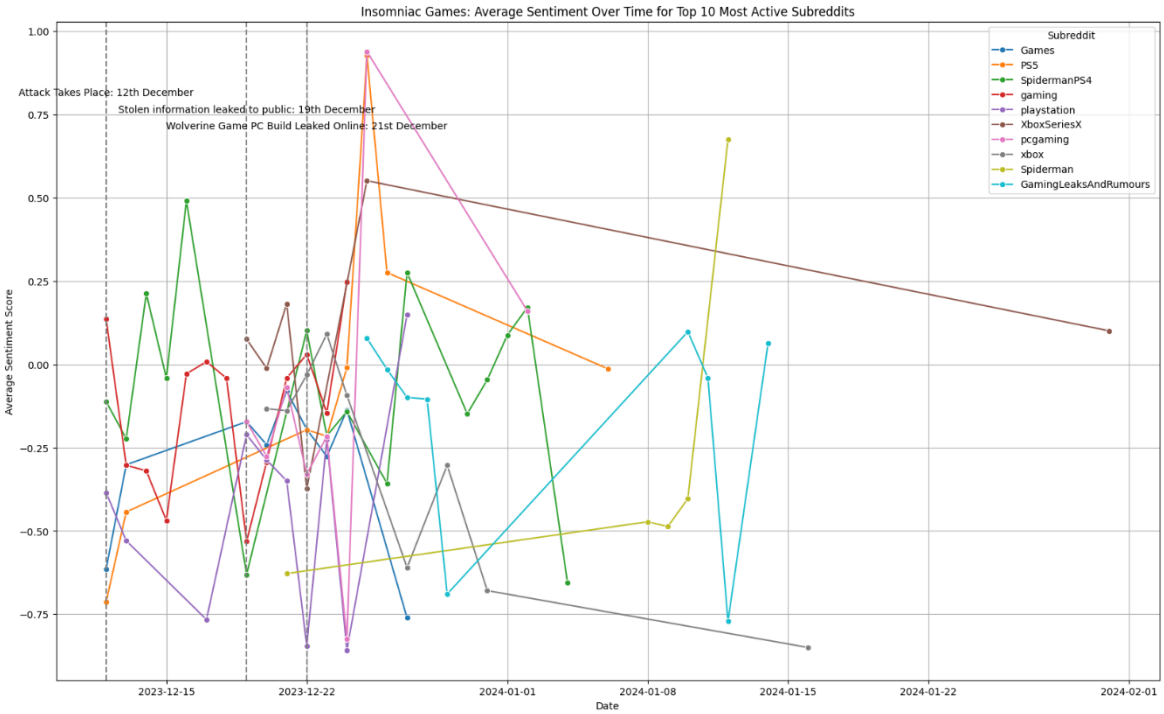


Figure 43: Subreddit sentiments over time for Insomniac.

For this attack, sentiment initially started negative, with increases in negative sentiment around the key dates. However, some positive sentiments also seem to have come as a reaction after some key dates. For example, the leaked Wolverine Game PC build received some positive reaction. This is shown further by the subreddits discussing this, like the "pcgaming" subreddit seeming to positively receive the news of this game build leak.

For WannaCry, with key dates provided by (ANY.RUN, n.d.):

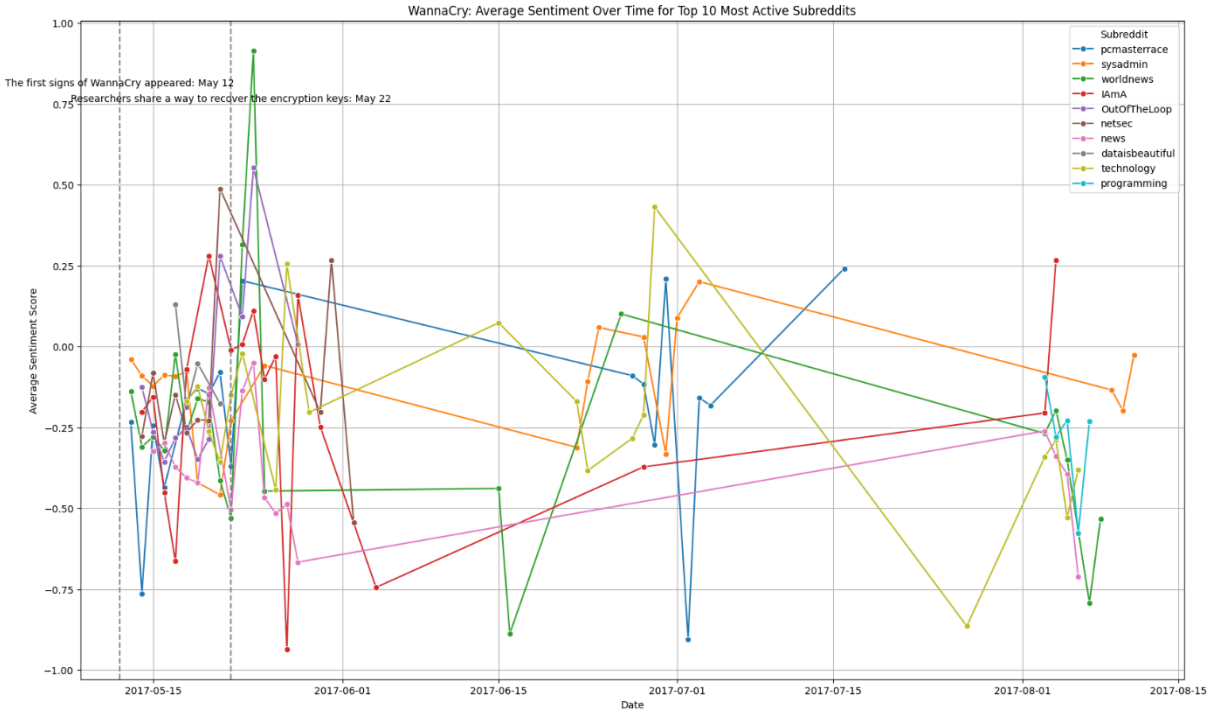


Figure 44: Subreddit sentiments over time for WannaCry.

This attack, much like with the Insomniac Games attack, started with negative reactions. Sentiment stays especially negative for that ten-day period between the two key dates shown. However, once a method to recover the encryption keys for the ransomware is revealed, there is an increase in positive sentiment reactions. This shows a clear change in sentiment as a reaction to an event associated to this cyber-attack.

For SolarWinds, with key dates provided by (Baker, 2021):

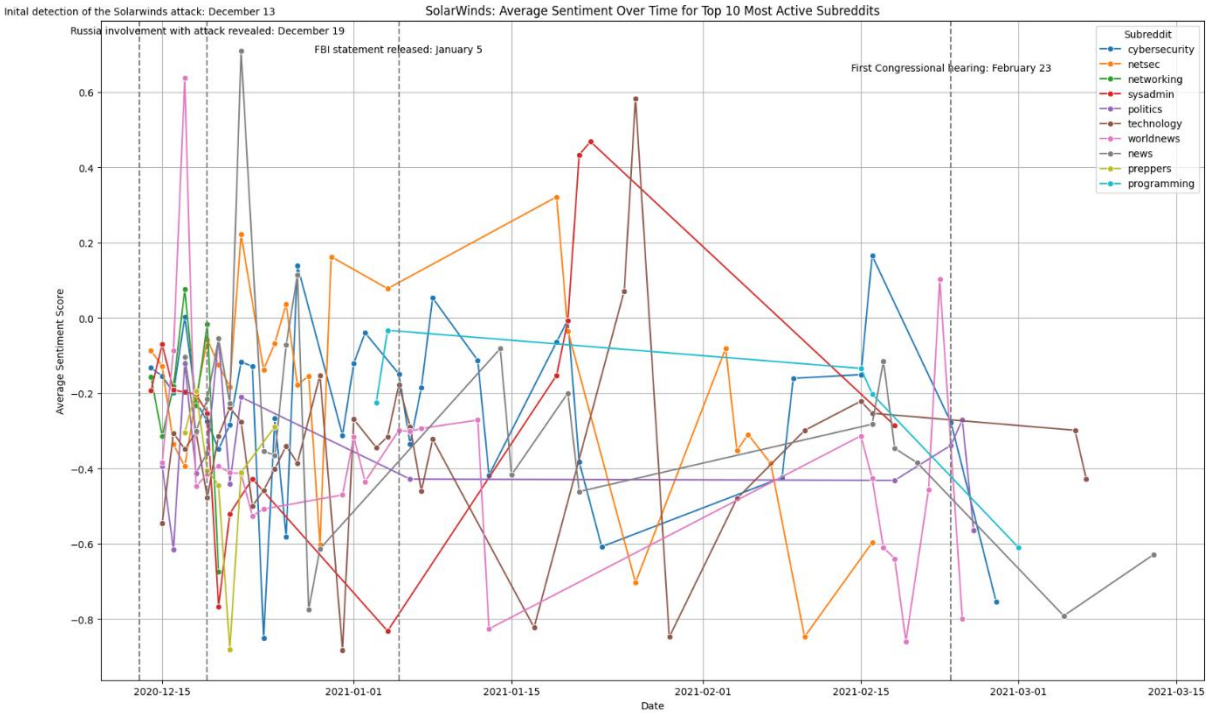


Figure 45: Subreddit sentiments over time for SolarWinds.

Sentiment tends towards being negative for most of this cyber-attack. The key dates shown appear to have had an impact, like more extreme negative sentiment occurring because of the reveal of alleged Russian involvement with the cyber-attack. This could also indicate a negative sentiment towards Russia around this time, too.

For all attacks:

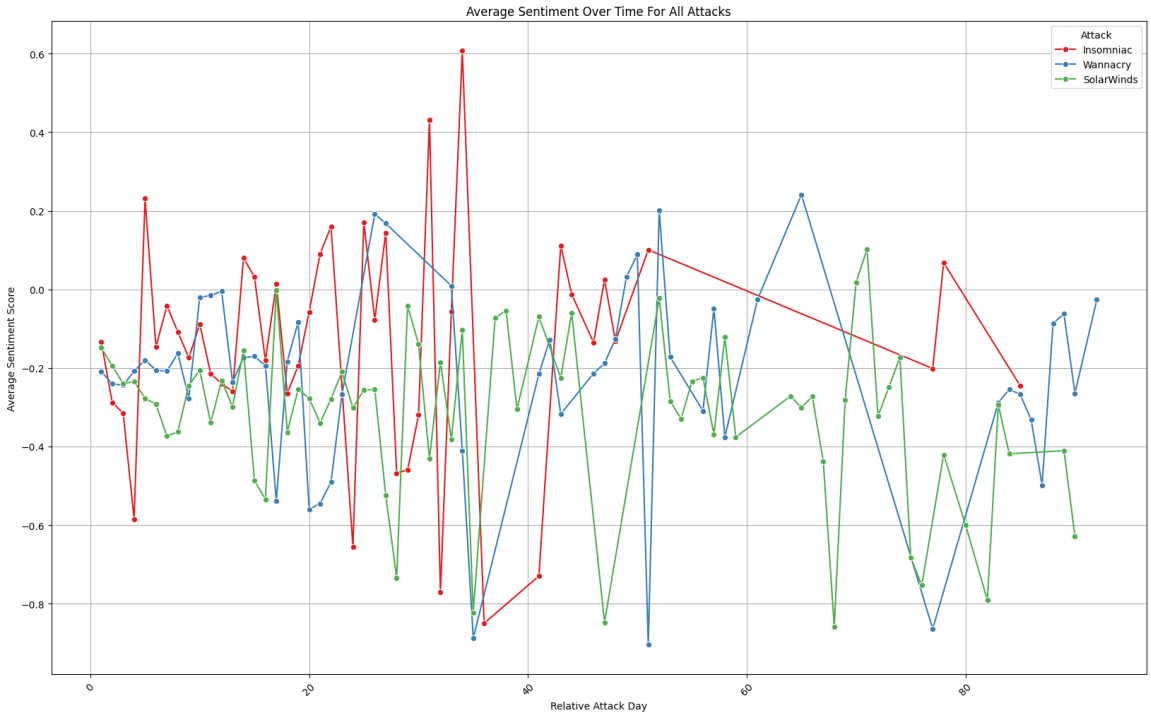


Figure 46: Sentiment over time for all three attacks.

These results suggest that reactions to cyber-attacks follow a pattern. This pattern tends to be majority negative responses for the first month, with fluctuations in sentiment dependent on key events that take place. Following this month period, discussion seems to die down a little, with less extreme fluctuations in score, and the scores themselves scores becoming less extreme (close to 0). Therefore, it could be said that as people discussing cyber-attacks online become used to the situation as time passes, leading to less intense emotions.

5.4 The relationship between sentiment and score

As discussed earlier, “score” refers to the upvotes a comment receives, the value is the aggregate of upvotes and downvotes a comment receives. This is especially relevant here as it could be suggested that the comments with the highest score represent the most popular opinions. To help determine the relationship between sentiment and score, the following metrics will be used:

- Correlation Coefficient – this measures the relationship between sentiment and score, a positive value would indicate that as sentiment becomes more positive, the score tends to increase, and vice versa.
- Mean entry score by sentiment category – the average score of positive, neutral, and negative comments.
- Standard Deviation of score by sentiment category – the variation of scores within each sentiment category.

The figure below shows the results of calculating these metrics:

Metric	Value
Correlation Coefficient	-0.005100692404649605
Mean Entry Score by Sentiment Category	{'negative': 16.708, 'neutral': 15.412, 'positive': 12.032}
Standard Deviation of Entry Scores by Sentiment Category	{'negative': 164.113, 'neutral': 153.396, 'positive': 82.32}

Figure 47: Relationship between sentiment score and entry score.

Based on these results, it can be said that there is a weak or no relationship between sentiment score and entry score. However, entries with negative sentiment tend to have slightly higher average scores than others.

To gain further insight into this, the top ten highest scoring (most upvotes) comments are shown below:

	Comment	Sentiment	Sentiment Score	Score
7683	hey everyone head real reddit account appear s...	neutral	0.379138	12566
2529	virus first hit friday exploit vulnerability w...	negative	-0.599739	9751
5130	wish guy would target people like corrupt poli...	negative	-0.888849	9163
1656	best retaliatory action go renewables sell tec...	neutral	0.019215	7777
439	probably took one look shitposting meme twitte...	negative	-0.643898	7573
3058	feeling technicality like play role lack accou...	negative	-0.842971	5644
1662	like subscribe encrypt file	neutral	0.026194	5393
9716	order work computer must rebooted infected ple...	neutral	-0.065996	5227
438	always wanted educational video possibly confe...	neutral	-0.048131	4709
9963	last thing want know computer infected anythin...	neutral	-0.332996	4009

Figure 48: Top 10 comments – by upvote.

These comments have all been classified as either negative or neutral, with the sentiment score being below 0 (indicating negative sentiment) for most comments. While the most popular comment has a positive sentiment score (around 0.4), many of the most popular opinions shared about these cyber-attacks are of negative sentiment.

In summary, there is evidence to suggest that there is a weak relationship between sentiment score and comment score. However, the most popular comments, while outliers in terms of score, have tended to have a negative sentiment score. These outliers are still relevant examples due to the high upvotes indicating popularity of a comment. Therefore, it can be said that negative and neutral opinions were the most popular on social media. This falls in line with what has been discussed previously about distributions of sentiment.

5.5 Topics of discussion

The below figures are word clouds for each cyber-attack.

For Insomniac Games:

Figure 49: Word cloud for Insomniac.

For WannaCry:

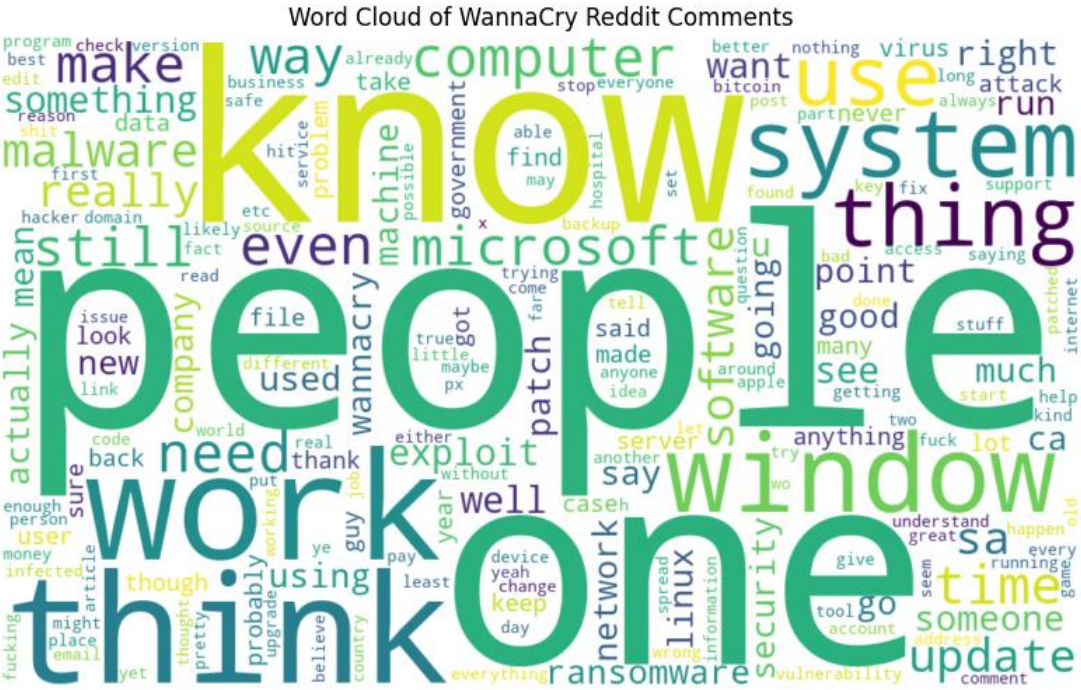


Figure 50: Word cloud for WannaCry.

For SolarWinds:

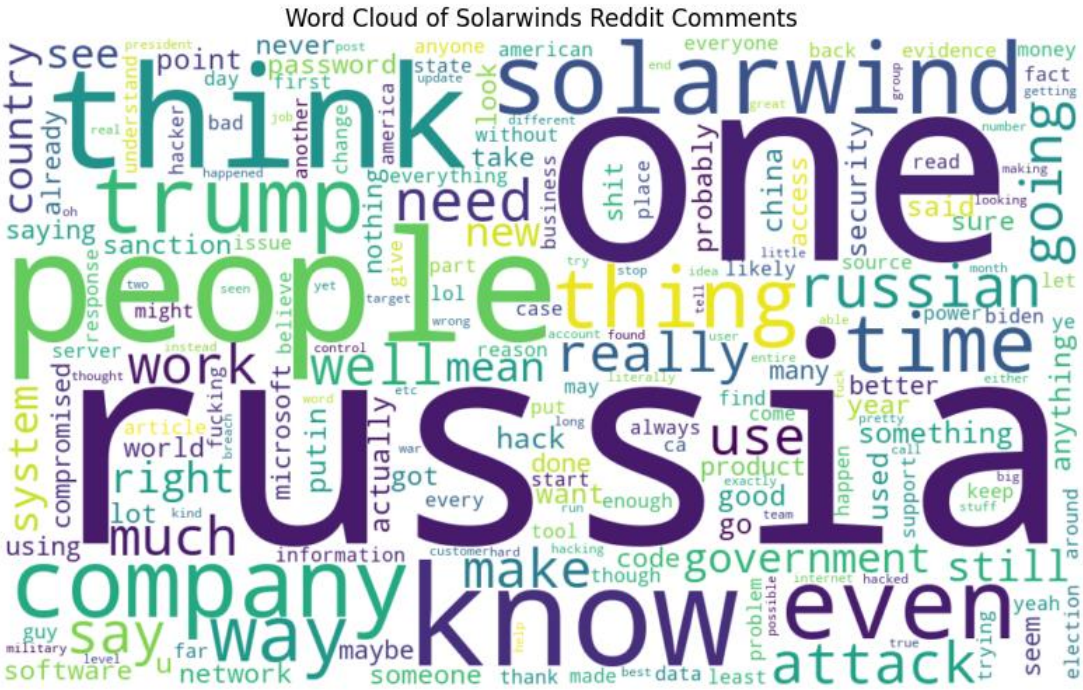


Figure 51: Word cloud for SolarWinds.

The titles of the posts with the most comments are shown below for each cyber-attack:
For Insomniac Games:

-

	post_id	comment_count	average_sentiment_score	title
0	18mi5ep	934	-0.132897	Xbox is not allowed X-Men games until 2036, according to illegally breached Insomniac Games materials
1	18luxxa	858	-0.180325	More than a terabyte of Insomniac Games' internal data has been leaked by hacker group, including internal HR documents, 'Wolverine' game files, and timeline of upcoming projects
2	18n8nws	675	-0.083512	Illegally breached data from Insomniac Games reveals that Xbox is not allowed X-men games until 2036 due to Sony signing a deal with Marvel
3	18hjfx6	536	-0.304509	Insomniac Games hacked, Wolverine screenshots and employee data stolen - Ransom of 2 Million Dollars
4	191swtu	500	-0.475445	Insomniac Spider-Man "fans" harassing MJ's video game model
5	18og8im	395	-0.186226	Insomniac Games releases statement regarding recent leaks
6	18m5g5f	393	0.052479	Insomniac Games Has Made Just $567 off Sunset Overdrive
7	18gkw38	388	-0.128790	So apparently, Insomniac got hit with a ransomware attack, wolverine got leaked, but there is no DLC or NG+ for SM2 yet
8	18n8mdj	387	-0.034999	Xbox is not allowed X-Men games until 2036, according to illegally breached Insomniac Games materials.
9	18og8n0	357	-0.221835	Insomniac Games releases statement regarding recent leaks

Figure 52: Active Insomniac Games posts.

For WannaCry:

-

	post_id	comment_count	average_sentiment_score	title
0	6cmmdf	2399	-0.005824	IamA the "accidental hero" who helped stop the WannaCry attack AMA!
1	6b7gd4	2324	-0.323143	Microsoft president blasts NSA for its role in 'WannaCry' computer ransom attack
2	6c495x	1147	-0.164970	WannaCry Ransomware Decryption Tool Released; Unlock Files Without Paying Ransom
3	6b5j9h	1056	-0.161477	[AMA Request] The 22 year old hacker who stopped the recent ransomware attacks on British hospitals.
4	6bacmd	842	-0.107104	WannaCry Megathread
5	6jsxq5	538	-0.252061	Huge cyber attack spreading across the world in potential repeat of 'Wannacry' hack
6	6rg0pj	442	-0.281171	FBI arrests WannaCry hero Marcus Hutchins in Las Vegas over malware claims
7	6rdkoy	402	-0.245931	Researcher Who Stopped WannaCry Ransomware Detained in US After Def Con
8	6b0qa0	377	-0.300400	It's Not Over, WannaCry 2.0 Ransomware Just Arrived With No 'Kill-Switch'
9	6bs66v	331	-0.180309	Balance of the bitcoin wallets tied to the WannaCry ransomware attack [OC]

Figure 53: Active WannaCry posts.

For SolarWinds:

	post_id	comment_count	average_sentiment_score	title
0	kgwi6h	2012	-0.400253	Biden is considering Russian financial sanctions or other retaliatory action in response to the SolarWinds hack
1	kp942d	1063	-0.343802	SolarWinds hack may be much worse than originally feared
2	lk37qq	1027	-0.332474	SolarWinds hack was 'largest and most sophisticated attack' ever: Microsoft president
3	mrcdah	797	-0.371458	U.S. to sanction Russia for alleged election interference, SolarWinds hack
4	kdx0p9	605	-0.329802	No One Knows How Deep Russia's Hacking Rampage Goes. A supply chain attack against IT company SolarWinds has exposed as many as 18,000 companies to Cozy Bear's attacks.
5	kdvaov	364	-0.116680	SolarWinds use the password 'solarwinds123' on their Update Servers
6	kebhgj	331	-0.259483	SolarWinds writes blog describing open-source software as vulnerable because anyone can update it with malicious code - Ages like fine wine
7	kcrd7y	325	-0.160192	According to FireEye, SolarWinds Orion platform allegedly compromised by foreign hackers.
8	keo19e	273	-0.300978	Preppers: SolarWinds should be a wake up call for you
9	lkt35u	251	-0.275077	Microsoft says it found 1,000-plus developers' fingerprints on the SolarWinds attack

Figure 54: Active SolarWinds posts.

Based on these results, most of the discussion for all cyber-attacks is related to updates relating to key events of the cyber-attack. In addition to this, the more active posts tend to have negative sentiment scores. This suggests that negative updates on the cyber-attack can lead to more users engaging with a post.

5.6 Conclusion

This chapter has discussed the findings of the sentiment analysis conducted on the collected data. Visualisations have been used where appropriate to convey information. The findings from the discussion can be summarised as:

- Most responses to the cyber-attacks were classed as either negative or neutral, and the sentiment scores suggest that most of the conversation fell in the negative value range.
- The reactions to all cyber-attacks discussed followed a pattern. The emotional responses tended to be negative for the first month or the attack occurring, with big fluctuations in sentiment influenced by key events for each attack. As time passed, less extreme sentiment was expressed in discussions, with sentiment score trending closer to zero.
- Comments with negative scores formed most of the popular comments. However, when all comments were considered, little relationship was found in general between sentiment score and popularity (upvotes) of a comment.
- The most engaged with posts tended to hold comments with an overall negative sentiment.

6 Evaluation

6.1 Introduction

This chapter will give an overview of how the project's objectives have been completed. In addition, suggestions will be provided for further work that could be conducted.

6.2 Completion of Objectives

The objectives set out in 2.1, and the project's completion of them, will be discussed here.

1. Determine project aims and objectives.

This objective was completed in section 2.2.

2. Conduct research and write a literature review for key aspects of the project.

This objective was completed throughout chapter 3. The literature review goes through key knowledge needed for the project, as well as influences on the approach to the project certain examples or pieces of information may have.

3. Using knowledge of sentiment analysis, decide what is being looked for in the data to be collected.

This objective was completed throughout section 4.4. The requirements of the analysis performed in this project, as well as the questions to be answered was discussed.

4. Determine criteria for the specific cyber-attacks to collect data about and select the attacks.

This objective was completed throughout section 4.2. The criteria for cyber-attack selection were discussed, and the chosen attacks were selected based upon these criteria. An introduction into the events of each attack was also provided.

5. Collect data using appropriate methods.

This objective was completed, the implementation of the data collection process was discussed in section 4.2. Discussions of the appropriate method of data collection was conducted during the literature review, in section 3.2.

6. Preprocess and store data appropriately.

This objective was completed, the approach to data storage was discussed in section 4.2. The implementation of the preprocessing of data was discussed throughout section 4.3.

7. Utilise Large Language Models to perform sentiment analysis.

Sections 4.4 and 4.5 detail the approach taken towards completion of this objective. Results are discussed in chapter 5.

8. Represent findings visually.

Chapter 5 involves the discussion of the results of the sentiment analysis, with visual representation used throughout. This objective was completed.

9. Report on findings and evaluate the results.

This objective was completed in chapter 5.

10. Evaluate the project.

This objective has been completed in this chapter, chapter 6. In section 6.2, the completion of the initial project objectives was discussed and in section 6.4, the project will be evaluated further.

6.3 Future Work and Improvements

6.3.1 Expanding data and analysis

Collecting data from additional social media websites like Twitter (X) could provide further insight into how users have responded. In addition, Twitter provides searching by date in queries, which could lead to a more in-depth analysis of certain time periods during a particular cyber-attack. For example, around the time of a key date discussed earlier.

6.3.2 Assessing multiple models for analysis, and metrics expansion

While in this project a single LLM was initially chosen for the sentiment analysis conducted, selecting several models, and choosing the best-performing one may lead to better model selection. This could involve the use of the metrics discussed for evaluating the model chosen in this project. In addition, more metrics could be introduced for model comparison. An example of this could be accounting for model size.

6.3.3 Fine-tuning of chosen model

Hugging Face provides the capabilities to fine-tune models. Obtaining a set of data for fine-tuning purposes is an additional route that could be explored. The performance of this fine-tuned model could also be evaluated against the pre-trained models available, with the aim conducting a more rigorous model selection process. This process could lead to more accurate results.

6.4 Conclusion

This chapter assessed how the project has completed the objectives that it set out to achieve in the “introduction” chapter. In addition to this, suggestions were provided for further work that could be conducted related to the topics covered in this project. Further evaluation of the project is provided below.

The project successfully achieved all objectives, and the extent to which they were completed was also discussed. This project has contributed to the literature regarding sentiment analysis by adding to the discussion surrounding how large language models can be utilised to perform this task. In addition to this, there has been contributions to literature regarding emotional responses to cyber-attacks.

7 References

Agarwal, A., Singh, R. & Toshniwal, D., 2017. Geospatial Sentiment analysis using twitter data for UK-EU referendum. *Journal of Information and Optimization Sciences,* Volume 39, pp. 1-15.

ANY.RUN, n.d. *WannaCry: The Most Preventable Ransomware is Still at Large.* [Online]
Available at: https://any.run/cybersecurity-blog/wannacry-ransomware/
[Accessed 27 April 2024].

Baker, P., 2021. *CSO Online: The SolarWinds hack timeline: Who knew what, and when?.* [Online]
Available at: https://www.csoonline.com/article/570537/the-solarwinds-hack-timeline-who-knew-what-and-when.html
[Accessed 27 April 2024].

Bandyopadhyay, S., Naskar, S. K. & Ekbal, A., 2013. *Emerging Applications of Natural Language Processing: Concepts and New Research,* Hershey, Pa.: IGI Global.

Camacho-Collados, J. et al., 2022. *TweetNLP: Cutting-Edge Natural Language Processing for Social Media.* s.l., Association for Computational Linguistics.

Cambria, E., Schuller, B., Xia, Y. & Havasi, C., 2013. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems,* 28(2), pp. 15-21.

Chen, M. et al., 2021. Evaluating large language models trained on code.. *arXiv.*

Cloudflare.com, n.d. *What is rate limiting?.* [Online]
[Accessed 15 April 2014].

Davies, R., 2023. *Insomniac Games hack: what happened?.* [Online]
Available at: https://www.standard.co.uk/news/tech/insomniac-games-hack-what-happened-rhysida-malware-b1128434.html
[Accessed 6 February 2024].

Dawson, R., 2012. *Relational databases: design and use.* s.l.:Loughborough University.

Dixon, S. J., 2023. *Number of social media users worldwide from 2017 to 2027.* [Online]
Available at: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/
[Accessed 15 April 2024].

Elastic NV, n.d. *What is a large language model (LLM)?.* [Online]
Available at: https://www.elastic.co/what-is/large-language-models#/what-is/large-language-models
[Accessed 23 December 2023].

Florida State University, 2016. *CSV Files.* [Online]
Available at: https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html
[Accessed 28 March 2024].

Georgiadou, E., Angelopoulos, S. & Drake, H., 2020. Big data analytics and international negotiations: sentiment analysis of Brexit negotiating outcomes. *International Journal of Information Management,* Volume 51.

Ghosh, S. & Gunning, D., 2019. *Natural Language Processing Fundamentals : Build Intelligent Applications That Can Interpret the Human Language to Deliver Impactful Results.* Birmingham: Packt Publishing.

Graupe, D., 2013. *Principles of Artificial Neural Networks.* 3rd ed. Singapore: World Scientific Publishing Company.

Hugging Face, n.d. *What is Zero-Shot Classification? - Hugging Face.* [Online]
Available at: https://huggingface.co/tasks/zero-shot-classification
[Accessed 13 April 2024].

Hugging Face, n.d. *Transformers.* [Online]
Available at: https://huggingface.co/docs/transformers/index
[Accessed 17 April 2024].
IBM, n.d. *What is a cyberattack?.* [Online]
Available at: https://www.ibm.com/topics/cyber-attack
[Accessed 24 December 2023].
IBM, n.d. *What is an API?.* [Online]
Available at: https://www.ibm.com/topics/api
[Accessed 7 January 2024].
Jupyter, n.d. *Project Jupyter | Home.* [Online]
Available at: https://jupyter.org/
[Accessed 28 April 2024].
Kietzmann, J. H., Hermkens, K., McCarthy, I. P. & Silvestre, B. S., 2011. Social media? Get serious! Understanding the functional building blocks of social media. *Business Horizons,* 54(3), pp. 241-251.
Lewis, M. et al., 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR,* 31 October.Volume abs/1910.13461.
Liu, B., 2020. *Sentiment analysis: mining opinions, sentiments, and emotions.* 2nd ed. Cambridge: Cambridge University Press.
Lucidchart, n.d. *Lucidchart.* [Online]
Available at: https://www.lucidchart.com/pages
[Accessed 21 April 2024].
Luong, T., Pham, H. & Manning, C. D., 2015. *Effective Approaches to Attention-based Neural Machine Translation.* Lisbon, Association for Computational Linguistics, pp. 1412 - 1421.
McCarthy, P. M. & Boonthum-Denecke, C., 2012. *Applied Natural Language Processing: Identification, Investigation and Resolution.* Hershey, PA.: IGI Global..
Negnevitsky, M., 2011. *Artifical Intelligence: A Guide to Intelligent Systems.* 3 ed. Harlow: Pearson Education, Limited.
NHS England, 2023. *NHS England business continuity management toolkit case study: WannaCry attack.* [Online]
Available at: https://www.england.nhs.uk/long-read/case-study-wannacry-attack/#:~:text=The%20WannaCry%20ransomware%20attack%20was,threatened%20to%20release%20data%2Finformation.
[Accessed 25 March 2024].
NumFOCUS, 2024. *pandas.DataFrame.drop_duplicates — pandas 2.2.2 documentation.* [Online]
Available at:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html#pandas-dataframe-drop-duplicates
[Accessed 12 April 2024].
Oladimeji, S. & Kerner, S. M., 2023. *SolarWinds hack explained: Everything you need to know.* [Online]
Available at: https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know
[Accessed 12 April 2024].
OpenAI, n.d. *Introduction - OpenAI API.* [Online]
Available at: https://platform.openai.com/docs/introduction
[Accessed 28 December 2023].

Oracle , n.d. *What Is a Database?.* [Online]
Available at: https://www.oracle.com/uk/database/what-is-database/
[Accessed 15 April 2024].
Reddit Help, 2023. *Available search features - Reddit Help.* [Online]
Available at: https://support.reddithelp.com/hc/en-us/articles/19696541895316-Available-search-features
[Accessed 22 April 2024].
Salawa, P., n.d. *SqLiteStudio.* [Online]
Available at: https://sqlitestudio.pl/
[Accessed 15 April 2024].
Shirey, J. B., 2023. *Game Rant: Wolverine Game PC Build Leaked Online.* [Online]
Available at: https://gamerant.com/wolverine-game-pc-build-leak/
[Accessed 27 April 2024].
Shu, K., Silva, A., Sampson, J. & Liu, H., 2018. *Understanding Cyber Attack Behaviors with Sentiment Information on Social Media..* s.l., Springer, Cham.
Shu, K., Sliva, A. L., Sampson, J. & Liu, H., 2018. *Understanding Cyber Attack Behaviours with Sentiment Information on Social Media.* s.l., Springer, Cham.
Stacey, P., Taylor, R., Olowosule, O. & Spanaki, K., 2021. Emotional reactions and coping responses of employees to a cyber-attack: A case study. *International Journal of Information Management.*
University of Texas, 2021. *Working with APIs: API Wrappers.* [Online]
Available at: https://guides.lib.utexas.edu/c.php?g=897091
[Accessed 27 March 2024].
Vaswani, A. et al., 2017. *Attention Is All You Need.* s.l., Curran Associates, Inc..
Yerashenia, N., Bolotov, A., Chan, D. & Pierantoni, G., 2020. *Semantic Data Pre-Processing for Machine Learning Based Bankruptcy Prediction Computational Model.* Antwerp, IEEE.

8 Appendix

8.1 reddit_collection.py

Please note that all credentials needed for connection to Reddit's API have been removed.

```
import datetime
import praw
import sqlite3

# Set up

reddit_client_id = ''
reddit_client_secret = ''
reddit_user_agent = ''
reddit_username = ''
reddit_password = ''
reddit_redirect_uri = ' '

conn = sqlite3.connect('Collected.db')
conn.cursor()

queries_insomniac = ["Insomniac Games AND ransomware",
                     "Insomniac Games AND cyber attack",
                     "Insomniac Games AND data breach",
                     "Insomniac Games AND Rhysida",
                     "Insomniac Games AND data leak",
                     "Insomniac Games AND ransom demand",
                     "Insomniac Games AND cyber extortion",
                     "Insomniac Games AND breach",
                     "Insomniac Games AND security incident",
                     "Insomniac Games AND ransomware group",
                     "Insomniac Games AND cyber security",
                     "Insomniac Games AND hacker group",
                     "Insomniac Games AND data encryption",
                     "Insomniac Games AND ransom negotiation",
                     "Insomniac Games AND breach impact"
                     "Insomniac Games AND cybersecurity response",
                     "Insomniac Games AND data protection",
                     "Insomniac Games AND ransom",
                     "Insomniac Games AND ransomware recovery",
                     "Insomniac Games cyber attack"]


queries_wannacry = ["WannaCry AND attack",
                    "WannaCry AND hack",
                    "WannaCry AND ransomware",
                    "WannaCry AND malware",
                    "WannaCry AND vulnerability",
                    "WannaCry AND exploit",
                    "WannaCry AND prevention",
                    "WannaCry AND mitigation",
                    "WannaCry AND impact",
                    "WannaCry AND consequences",
                    "WannaCry AND recovery",
                    "WannaCry AND remediation",
                    "WannaCry AND cybersecurity",
                    "WannaCry AND defense",
                    "WannaCry AND analysis",
                    "WannaCry AND assessment",
                    "WannaCry AND incident",
                    "WannaCry AND breach",
                    "WannaCry AND ransom",
                    "WannaCry AND ransomware",
                    "WannaCry cyber attack"]

queries_solarwinds = ["SolarWinds AND attack",
                      "SolarWinds AND breach",
                      "SolarWinds AND hack",
                      "SolarWinds AND compromised",
                      "SolarWinds AND cyber attack",
                      "SolarWinds AND intrusion",
                      "SolarWinds AND incident",
                      "SolarWinds AND cyber espionage",
                      "SolarWinds AND infiltration",
                      "SolarWinds AND malicious activity",
                      "SolarWinds AND vulnerability exploit",
                      "SolarWinds AND data breach",
                      "SolarWinds AND compromise detection",
                      "SolarWinds AND threat intelligence",
                      "SolarWinds AND forensics",
                      "SolarWinds AND investigation",
                      "SolarWinds AND remediation",
                      "SolarWinds AND recovery",
                      "SolarWinds AND vulnerability",
                      "SolarWinds hack"]
```

```python
def store_post(submission, attack_id):
    sql_query = "INSERT INTO reddit_posts(post_id, subreddit, title, content, score, upvote_ratio, post_datetime, attack_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"
    subreddit = submission.subreddit.display_name
    conn.execute(sql_query, (submission.id, subreddit, submission.title, submission.selftext, submission.score, submission.upvote_ratio, submission.created_utc, attack_id))
    conn.commit()

def store_comment(comment, post_id):
    sql_query = "INSERT INTO reddit_parent_comments(parent_comment_id, content, score, comment_datetime, post_id) VALUES (?, ?, ?, ?, ?)"
    conn.execute(sql_query, (comment.id, comment.body, comment.score, comment.created_utc, post_id))
    conn.commit()

def store_reply(reply, comment_id, post_id):
    sql_query = "INSERT INTO reddit_replies(reply_id, content, score, comment_datetime, parent_id, post_id) VALUES (?, ?, ?, ?, ?, ?)"
    conn.execute(sql_query, (reply.id, reply.body, reply.score, reply.created_utc, comment_id, post_id))
    conn.commit()


def search_for_information(reddit, attack_id, queries):
    search = reddit.subreddit("all")
    for search_query in queries:
        search_results = search.search(search_query, sort='relevance', limit = 50)
        # Search for posts based on the query
        for submission in search_results:
            print(f"Title: {submission.title}")
            # Store the post
            store_post(submission, attack_id)
            # Search for comments in the post
            submission.comments.replace_more(limit=None)
            for comment in submission.comments.list():
                # Store the comment
                store_comment(comment, submission.id)
                for reply in comment.replies.list():
                    #Store replies
                    store_reply(reply, comment.id, submission.id)
            print(f"Saved for submission {submission.id}")
        print(f"\nSaved query {search_query}")
        print("\n---\n")
    return

def startup():
    # Establish connection to reddit
    try:
        reddit = praw.Reddit(
            client_id=reddit_client_id,
            client_secret=reddit_client_secret,
            user_agent=reddit_user_agent,
            username=reddit_username,
            password=reddit_password,
            redirect_uri = reddit_redirect_uri,
        )

        # Print information about the Reddit instance, to prove it worked.
        print(f"Reddit Instance Information:")
        print(f"Authenticated: {not reddit.read_only}")  # If True, it means the instance is read-only (non-authenticated)
        print(f"User: {reddit.user.me()}")  # Print the authenticated user's information

    except praw.exceptions.RedditAPIException as e:
        print(f"Authentication failed: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

    #Set up search, and begin
    search_or_test = input("searching or testing? Default is testing ").lower()
    if(search_or_test == "search" or search_or_test == "searching"):
        attack = (input("Which attack is this? Insomniac(I), WannaCry(W), or Solarwinds(S)? ").lower())
        if (attack == "i" or attack == "insomniac"):
            search_for_information(reddit, 1, queries_insomniac)
            print("Saved Insomniac Queries")
        if (attack == "w" or attack == "wannacry"):
            search_for_information(reddit, 2, queries_wannacry)
            print("Saved WannaCry Queries")
        if (attack == "s" or attack == "solarwinds"):
            search_for_information(reddit, 3, queries_solarwinds)
            print("Saved SolarWinds Queries")
    else:
        # Get rate limit information
        rate_limit = reddit.auth.limits
        # Print rate limit information
        print(rate_limit)
        reset_timestamp = rate_limit['reset_timestamp']
        reset_datetime = datetime.datetime.fromtimestamp(reset_timestamp)
        # Print the reset datetime
        print("Reset datetime:", reset_datetime)
        print("Test done")

startup()
```

8.2 Data Preprocessing.ipynb

Table of contents


```
[1]
import re
import warnings
import nltk
import sqlite3
import numpy as np
import pandas as pd
from typing import Tuple
from transformers import pipeline
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from autocorrect import Speller
from langdetect import detect
```

```
[2]
# Filter out warnings, they were annoying
warnings.simplefilter(action='ignore', category=FutureWarning)
pd.options.mode.chained_assignment = None  # default='warn'
```

Data Preprocessing: Insomniac Games Attack

Loading Posts, Comments, and Replies

```
conn = sqlite3.connect('Collected.db')
attacks = pd.read_sql(sql = "select * from attacks", con = conn, index_col = "attack_id" )
attacks
```

[3]

attack_id	attack_name
1	Insomniac Games
2	WannaCry
3	SolarWinds

```
def fetch_attack(attack_id: int) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = pd.read_sql(sql = f"select * from reddit_posts where attack_id = {attack_id}", con = conn, index_col = "rp_index")
    comments = pd.read_sql(sql = f"select * from reddit_parent_comments where post_id in"
                           f"(select post_id from reddit_posts where attack_id = {attack_id})", con = conn, index_col = "rc_index")
    replies = pd.read_sql(sql = f"select * from reddit_replies where parent_id in"
                          f"(select parent_comment_id from reddit_parent_comments where post_id in"
                          f"(select post_id from reddit_posts where attack_id = {attack_id}))",con = conn, index_col = "rr_index")
    return posts, comments, replies
```

[4]

```
insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies = fetch_attack(1)
```

[5]

```
insomniac_posts
```

[6]

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
1	18mf8b5	TwoBestFriendsPlay	Statement from Remedy on the Insomniac Games r...		505	0.98	1703027019	1
2	18mf8b5	TwoBestFriendsPlay	Statement from Remedy on the Insomniac Games r...		497	0.98	1703027019	1
3	18ogglo	pcgaming	Insomniac Games Releases Statement on recent r...		630	0.90	1703256167	1
4	18gkw38	SpidermanPS4	So apparently, Insomniac got hit with a ransom...		1584	0.88	1702384490	1
5	18lzgnk	Marvel	Several Insomniac Marvel games and estimate re...		208	0.88	1702984282	1
...	...	...	...	...	...	...	...	...
2355	18vx15o	videogames	Which games do you like more - The Batman Arkh...		459	0.92	1704116661	1
2356	10ru6wt	Games	Rumor: Insomniac's Wolverine game is reportedl...		6284	0.96	1675355840	1
2357	1byqmuy	InsomniacGames	How would you rank these 6 Insomniac games?		668	1.00	1712556487	1
2358	16ytirx	SpidermanPS4	Posts like these always make me wonder how man...	Like yeah, he does have financial problems. Th...	2466	0.95	1696343570	1
2359	vekpox	worldnews	Putin blasts West in 73-minute rant after cybe...		17774	0.94	1655485305	1

363 rows × 8 columns

[7]

```
insomniac_parent_comments
```

rc_index	parent_comment_id	content	score	comment_datetime	post_id
1	ke3saoh	The personal info getting leaked is honestly s...	232	1703027544	18mf8b5
2	ke3u1iu	It's one thing to leak the games for everyone ...	136	1703028248	18mf8b5
3	ke45kdu	It's a sobering reminder that the people who d...	73	1703032967	18mf8b5
4	ke3xow5	Although this may strike some readers as harsh...	75	1703029736	18mf8b5
5	ke3xbdr	It's good that Insom is at least receiving sup...	51	1703029581	18mf8b5
...	...	...	...	...	...
227748	j5pbeng	>\tFrom developing the best VR headset on the ...	3	1674579887	10k7l1p
227749	j5pk2kj	For those reading this thread in the future, w...	4	1674583040	10k7l1p
227750	j5pywow	> Was it lackluster for having waited over a d...	3	1674588451	10k7l1p
227751	j5r85fs	Oh I get it you're trolling, WP man.	1	1674605651	10k7l1p
227752	j5rdmhy	Because the only ones who have issues with ker...	1	1674607948	10k7l1p

51916 rows × 5 columns

[8]

```
insomniac_reddit_replies
```

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
1	ke3sw9d	Right? I'll have *no* sympathy for the hackers...	162	1703027787	ke3saoh	18mf8b5
2	ke4grbb	It sucks for a couple reasons\n\n1- timing, it...	57	1703037616	ke3saoh	18mf8b5
3	ke496do	Especially since a certain crowd has already m...	47	1703034454	ke3saoh	18mf8b5
4	ke4tx0q	Honestly, I wouldn't get your hopes up for jus...	48	1703043362	ke3saoh	18mf8b5
5	ke47jzq	[removed]	-44	1703033783	ke3saoh	18mf8b5
...	...	...	...	...	...	...
654931	j5pbeng	>\tFrom developing the best VR headset on the ...	3	1674579887	j5pasxl	10k7l1p
654932	j5pk2kj	For those reading this thread in the future, w...	4	1674583040	j5phwp0	10k7l1p
654933	j5pywow	> Was it lackluster for having waited over a d...	3	1674588451	j5phwp0	10k7l1p
654934	j5r85fs	Oh I get it you're trolling, WP man.	1	1674605651	j5r7zaz	10k7l1p
654935	j5rdmhy	Because the only ones who have issues with ker...	1	1674607948	j5r91at	10k7l1p

134466 rows × 6 columns

[9]

```
print(np.shape(insomniac_posts))
print(np.shape(insomniac_parent_comments))
print(np.shape(insomniac_reddit_replies))
```

```
(363, 8)
(51916, 5)
(134466, 6)
```

Removing duplicates

```
insomniac_posts = insomniac_posts.drop_duplicates(subset = 'post_id')
insomniac_parent_comments = insomniac_parent_comments.drop_duplicates(subset = 'parent_comment_id')
insomniac_reddit_replies = insomniac_reddit_replies.drop_duplicates(subset = 'reply_id')
print(np.shape(insomniac_posts))
print(np.shape(insomniac_parent_comments))
print(np.shape(insomniac_reddit_replies))
```
[10]

```
(238, 8)
(35391, 5)
(25215, 6)
```

Removing based on date of post

- Remove all posts earlier than 2 weeks before the attack beginning.
- Each attack has a date that can be pinpointed as the start of the attack.
- In the case of solarwinds, this will be treated as the date of its discovery

https://news.sky.com/story/wolverine-what-we-know-about-the-cyberattack-that-leaked-one-of-playstations-most-anticipated-games-13034721#:~:text=When and how did the,stolen data to be leaked.

```
def remove_by_date(cutoff_date: int, posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    # Filter posts based on the cutoff date
    posts = posts[posts['post_datetime'] >= cutoff_date]
    # Filter parent_comments based on the cutoff date
    parent_comments = parent_comments[parent_comments['comment_datetime'] >= cutoff_date]
    # Filter reddit_replies based on the cutoff date
    reddit_replies = reddit_replies[reddit_replies['comment_datetime'] >= cutoff_date]
    return posts, parent_comments, reddit_replies
```
[11]

```
insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies = remove_by_date(1701129600, insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies)
print(np.shape(insomniac_posts))
print(np.shape(insomniac_parent_comments))
print(np.shape(insomniac_reddit_replies))
```
[12]

```
(148, 8)
(13021, 5)
(8830, 6)
```

Removing off topic entries

- facebook/bart-large-mnli model perfroms zero-shot classification
- This is topic classification, each title will get a score based on its likelihood of being about each given topic.
- Posts below the minimum cut-off point are removed

```
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")
```
[13]

```
def get_off_topic(titles: list, labels: list) -> list:
    classifier_outputs = []
    for title in titles:
        sequence_to_classify=title
        classifier_outputs.append(classifier(sequence_to_classify, labels, multi_label = True))
    off_topic = []
    for index, output in enumerate(classifier_outputs):
        if max(output['scores']) <= 0.65: # Scores cut off point.
            off_topic.append({'index': index, 'title': output['sequence']})
    return off_topic
```
[14]

```
insomniac_titles = insomniac_posts['title']
insomniac_titles
```
[15]

```
rp_index
1       Statement from Remedy on the Insomniac Games r...
3       Insomniac Games Releases Statement on recent r...
4       So apparently, Insomniac got hit with a ransom...
5       Several Insomniac Marvel games and estimate re...
6       Insomniac Games alerts employees hit by ransom...
                              ...
2342    Who's your favorite and least favorite charact...
2347      Insomniac leak: Demand for the remastered games
2354    Should insomniac make a spider man 2099 game (...
2355    Which games do you like more - The Batman Arkh...
2357          How would you rank these 6 Insomniac games?
Name: title, Length: 148, dtype: object
```

```
insomniac_labels = ["Insomniac leak", "Wolverine leak", "Spider-Man leak", "Insomniac cyber attack", "Insomniac ransomware", "Insomniac hack", "Insomniac stolen information"]
insomniac_labels
```
[16]

```
['Insomniac leak',
 'Wolverine leak',
 'Spider-Man leak',
 'Insomniac cyber attack',
 'Insomniac ransomware',
 'Insomniac hack',
 'Insomniac stolen information']
```

```
insomniac_off_topic =  get_off_topic(insomniac_titles, insomniac_labels)
```
[17]

```
insomniac_off_topic
```

```
[{'index': 29,
  'title': 'Rhysida užpuolė "Sony" dukterinę įmonę "Insomniac Games"'},
 {'index': 32,
  'title': 'So... should gaming journalists be gaming journalists?'},
 {'index': 39,
  'title': 'Discussion, News, and Request Thread - week beginning 12/17/23'},
 {'index': 43, 'title': 'Epic hackerata'},
 {'index': 46, 'title': 'Cyber Briefing - 2023.12.12'},
 {'index': 47, 'title': 'news del 12.12.23'},
 {'index': 48, 'title': 'Monthly data breach alert'},
 {'index': 49,
  'title': 'Week 51 | Cybersecurity - technology - privacy News recap'},
 {'index': 55,
  'title': 'DQM: The Dark Prince Accessories List with effects and which monster drops them'},
 {'index': 59,
  'title': 'Sysadmins that work for Game Studios: is it as fucking awful as it seems?'},
 {'index': 60,
  'title': 'Do breaches actually cause for a loss in trust and credibility?'},
 {'index': 61, 'title': 'i got sued, been a fun time boys'},
 {'index': 62,
  'title': 'LittleBigPlanet 3 has sold 5.4mil units (making it the highest selling game in the series), Sackboy has sold 1.28mil units, as of February 2022.'},
 {'index': 63, 'title': 'Pc release Through leaks?'},
 {'index': 68,
  'title': 'Leak: Sony has signed a contract with Marvel for three games in the X-Men universe for PlayStation and PC'},
 {'index': 69, 'title': 'Leaks Compilation + How to find the leaks'},
...
 {'index': 145,
  'title': 'Should insomniac make a spider man 2099 game (ignore the image)'},
 {'index': 146,
  'title': 'Which games do you like more - The Batman Arkham games or the Insomniac Spider-Man games?'},
 {'index': 147, 'title': 'How would you rank these 6 Insomniac games?'}]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
def remove_off_topic(off_topic: list, posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    post_ids = []
    # Iterate through each title and remove accordingly
    for i in off_topic:
        posts_to_remove = posts[posts['title'] == i['title']]
        post_ids.extend(np.unique([posts_to_remove["post_id"]]))
    posts = posts[~posts['post_id'].isin(post_ids)]
    parent_comments = parent_comments[~parent_comments['post_id'].isin(post_ids)]
    reddit_replies = reddit_replies[~reddit_replies['post_id'].isin(post_ids)]
    return posts, parent_comments, reddit_replies
```

```
insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies = remove_off_topic(insomniac_off_topic, insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies)
print(np.shape(insomniac_posts))
print(np.shape(insomniac_parent_comments))
print(np.shape(insomniac_reddit_replies))
```

```
(96, 8)
(8107, 5)
(6394, 6)
```

Data Preprocessing: WannaCry Attack

Load

```
wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies = fetch_attack(2)
```

```
wannacry_posts
```

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
314	6cmmdf	IAmA	IamA the "accidental hero" who helped stop the...	**My short bio:** Hey I'm MalwareTech, a malwa...	24035	0.88	1495451693	2
315	6b7gd4	news	Microsoft president blasts NSA for its role in...		26998	0.91	1494810543	2
316	6bs66v	dataisbeautiful	Balance of the bitcoin wallets tied to the Wan...		4428	0.94	1495060723	2
317	nwa7t4	cybersecurity	A WannaCry documentary that I made	Hi everyone,\n\nnot sure if I'm allowed to be ...	734	0.99	1623283908	2
318	gid4jk	hacking	Confessions of Marcus Hutchins, the hacker who...		664	0.98	1589296605	2
...	...	...	...	...	...	...	...	...
2441	6aujob	technology	Ransomeware Cyberattack Mega-Thread	Hi folks, \n\nIn light of the ongoing world-wi...	1002	0.93	1494629211	2
2442	1byxk37	conspiracy	WannaCry Ransomware Attack Was Launched by N.S...	\n\nhttps://preview.redd.it/h4drfasm99tc1.png...	0	0.48	1712582043	2
2443	hbpc9o	worldnews	Australia hit by massive cyber attack		31975	0.93	1592522346	2
2444	6cmmdf	IAmA	IamA the "accidental hero" who helped stop the...	**My short bio:** Hey I'm MalwareTech, a malwa...	24032	0.88	1495451693	2
2445	6b7gd4	news	Microsoft president blasts NSA for its role in...		26995	0.91	1494810543	2

1082 rows × 8 columns

```
wannacry_parent_comments
```

rc_index	parent_comment_id	content	score	comment_datetime	post_id
14980	dhvr48z	First of all - you have the thanks of many! I'...	3420	1495451975	6cmmdf
14981	dhvrx7q	Did you receive some job offerings from govern...	2220	1495453860	6cmmdf
14982	dhvr4pb	What's your PC setup specs?\n\nAlso, what VM s...	2178	1495452007	6cmmdf
14983	dhvr463	What are some good resources or ways to learn ...	1336	1495451969	6cmmdf
14984	dhvs8md	HELLO SIR, GOOD WORK WITH THE KILL SWITCH. MY ...	1327	1495454550	6cmmdf
...	...	...	...	...	...
230686	dhwys1y	Hey trey, long time fan of the company, first ...	5	1495508162	6cmmdf
230687	dhwwjhq	The best thing you can do is read the original...	3	1495505343	6cmmdf
230688	dhx2r1u	Hey Josh, glad you chose Emoji-propriate Inc. ...	7	1495514143	6cmmdf
230689	dhwwwtx	Thanks a bunch. And no worries, my job revolve...	3	1495505815	6cmmdf
230690	dhwwys4	> my job revolves around googling\n\nThis is t...	3	1495505880	6cmmdf

133306 rows × 5 columns

```
wannacry_reddit_replies
```

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
39789	dhvs1up	I've always wanted to do educational videos an...	4709	1495454139	dhvr48z	6cmmdf
39790	dhvur2r	Is this the beginning of a YouTuber?	1301	1495459154	dhvr48z	6cmmdf
39791	dhw4ay3	You can probably be a keynote speaker for any ...	28	1495471292	dhvr48z	6cmmdf
39792	dhw8piv	Man, you should get in contact with Brady Hara...	13	1495476307	dhvr48z	6cmmdf
39793	dhw4u90	A TED talk would be awesome	10	1495471900	dhvr48z	6cmmdf
...	...	...	...	...	...	...
663396	dhwwys4	> my job revolves around googling\n\nThis is t...	3	1495505880	dhwsue8	6cmmdf
663397	dhx2r1u	Hey Josh, glad you chose Emoji-propriate Inc. ...	7	1495514143	dhwys1y	6cmmdf
663398	dhwwwtx	Thanks a bunch. And no worries, my job revolve...	3	1495505815	dhwwjhq	6cmmdf
663399	dhwwys4	> my job revolves around googling\n\nThis is t...	3	1495505880	dhwwjhq	6cmmdf
663400	dhwwys4	> my job revolves around googling\n\nThis is t...	3	1495505880	dhwwwtx	6cmmdf

385312 rows × 6 columns

Remove duplicates

```
wannacry_posts = wannacry_posts.drop_duplicates(subset = 'post_id')
wannacry_posts
```

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
314	6cmmdf	IAmA	IamA the "accidental hero" who helped stop the...	**My short bio:** Hey I'm MalwareTech, a malwa...	24035	0.88	1495451693	2
315	6b7gd4	news	Microsoft president blasts NSA for its role in...		26998	0.91	1494810543	2
316	6bs66v	dataisbeautiful	Balance of the bitcoin wallets tied to the Wan...		4428	0.94	1495060723	2
317	nwa7t4	cybersecurity	A WannaCry documentary that I made	Hi everyone,\n\nnot sure if I'm allowed to be ...	734	0.99	1623283908	2
318	gid4jk	hacking	Confessions of Marcus Hutchins, the hacker who...		664	0.98	1589296605	2
...	...	...	...	...	...	...	...	...
2375	1ax59d1	wallstreetbets	Cyber attack? AT&T is down premarket		1122	0.92	1708604948	2
2378	7kpiac	politics	US to announce North Korea behind WannaCry cyb...		38	0.82	1513643331	2
2379	lfxs44	gaming	CD projekt red just got cyber attacked		45662	0.95	1612857917	2
2380	7kphs3	worldnews	U.S. to blame North Korea for 'WannaCry' cyber...		77	0.80	1513643169	2
2381	tz4q6e	worldnews	Finland Hit by Cyber Attack, Airspace Breach a...		28208	0.96	1649427913	2

819 rows × 8 columns

```
wannacry_parent_comments = wannacry_parent_comments.drop_duplicates(subset ='parent_comment_id')
wannacry_parent_comments
```

rc_index	parent_comment_id	content	score	comment_datetime	post_id
14980	dhvr48z	First of all - you have the thanks of many! I'...	3420	1495451975	6cmmdf
14981	dhvrx7q	Did you receive some job offerings from govern...	2220	1495453860	6cmmdf
14982	dhvr4pb	What's your PC setup specs?\n\nAlso, what VM s...	2178	1495452007	6cmmdf
14983	dhvr463	What are some good resources or ways to learn ...	1336	1495451969	6cmmdf
14984	dhvs8md	HELLO SIR, GOOD WORK WITH THE KILL SWITCH. MY ...	1327	1495454550	6cmmdf
...	...	...	...	...	...
210158	drhc468	let's think about it.... a poor country with b...	1	1513706744	7kphs3
210159	drhd0ss	The war sell itself, fuck nuclear proliferatio...	1	1513707674	7kphs3
210160	drhdi6v	War never sells itself unless we're attacked d...	2	1513708153	7kphs3
210161	drhmpsd	That was when the American public was bloodthi...	2	1513717463	7kphs3
210162	drhdum3	>NK already has nuclear weapons\n\nYes, but th...	1	1513708500	7kphs3

68980 rows × 5 columns

```
wannacry_reddit_replies = wannacry_reddit_replies.drop_duplicates(subset = 'reply_id')
wannacry_reddit_replies
```

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
39789	dhvs1up	I've always wanted to do educational videos an...	4709	1495454139	dhvr48z	6cmmdf
39790	dhvur2r	Is this the beginning of a YouTuber?	1301	1495459154	dhvr48z	6cmmdf
39791	dhw4ay3	You can probably be a keynote speaker for any ...	28	1495471292	dhvr48z	6cmmdf
39792	dhw8piv	Man, you should get in contact with Brady Hara...	13	1495476307	dhvr48z	6cmmdf
39793	dhw4u90	A TED talk would be awesome	10	1495471900	dhvr48z	6cmmdf
...	...	...	...	...	...	...
594143	drgw9zm	It's easy to spoof IP address or sprinkle Russ...	3	1513687945	drgitsw	7kphs3
594144	drgjsfz	I don't know about those stuffs,but according ...	-2	1513660701	drgitsw	7kphs3
594145	drgz7zx	Yeah but in that case they'd maybe not use day...	1	1513692603	drgitsw	7kphs3
594146	drgsww3	yes, that includes anonymous, they aint no thing.	1	1513680242	drgitsw	7kphs3
594147	drhc468	let's think about it.... a poor country with b...	1	1513706744	drgitsw	7kphs3

47310 rows × 6 columns

Remove by date

https://www.vox.com/new-money/2017/5/15/15641196/wannacry-ransomware-windows-xp

```
wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies = remove_by_date(1493161200, wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies)
print(np.shape(wannacry_posts))
print(np.shape(wannacry_parent_comments))
print(np.shape(wannacry_reddit_replies))
```

[28]

```
(811, 8)
(66752, 5)
(45644, 6)
```

Remove off topic entries

```
wannacry_titles = wannacry_posts['title']
wannacry_titles
```

[29]

```
rp_index
314     IamA the "accidental hero" who helped stop the...
315     Microsoft president blasts NSA for its role in...
316     Balance of the bitcoin wallets tied to the Wan...
317                   A WannaCry documentary that I made
318     Confessions of Marcus Hutchins, the hacker who...
                              ...
2375                Cyber attack? AT&T is down premarket
2378    US to announce North Korea behind WannaCry cyb...
2379              CD projekt red just got cyber attacked
2380    U.S. to blame North Korea for 'WannaCry' cyber...
2381    Finland Hit by Cyber Attack, Airspace Breach a...
Name: title, Length: 811, dtype: object
```

```
wannacry_labels = ["WannaCry attack", "WannaCry cyber attack", "WannaCry hack", "WannaCry ransomware", "WannaCry Windows", "WannaCry Bitcoin"]
wannacry_labels
```

[30]

```
['WannaCry attack',
 'WannaCry cyber attack',
 'WannaCry hack',
 'WannaCry ransomware',
 'WannaCry Windows',
 'WannaCry Bitcoin']
```

```
wannacry_off_topic = get_off_topic(wannacry_titles, wannacry_labels)
```

[31]

```
wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies = remove_off_topic(wannacry_off_topic, wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies)
print(np.shape(wannacry_posts))
print(np.shape(wannacry_parent_comments))
print(np.shape(wannacry_reddit_replies))
```

[32]

```
(419, 8)
(17355, 5)
(14569, 6)
```

Data Preprocessing: SolarWinds Attack

Load

```
solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies = fetch_attack(3)
```

[33]

Remove duplicates

```
solarwinds_posts = solarwinds_posts.drop_duplicates(subset ='post_id')
solarwinds_posts
```

[34]

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
1360	lk37qq	worldnews	SolarWinds hack was 'largest and most sophisti...		14737	0.97	1613355014	3
1361	lkt35u	news	Microsoft says it found 1,000-plus developers'...		4221	0.97	1613440951	3
1362	lk9wu2	programming	Microsoft says it found 1,000-plus developers'...		1824	0.92	1613380225	3
1363	kgwi6h	worldnews	Biden is considering Russian financial sanctio...		38014	0.90	1608480284	3
1364	kdx0p9	technology	No One Knows How Deep Russia's Hacking Rampage...		11515	0.96	1608074277	3
...	...	...	...	...	...	...	...	...
2305	mg48j9	RedPacketSecurity	SolarWinds Patches Four New Vulnerabilities in...		1	1.00	1617066068	3
2307	hb1zu9	msp	SolarWinds RMM: Security Notice Regarding An A...	Just an FYI for those using Solarwinds RMM\n\n...	28	0.84	1592433844	3
2308	ld4f7i	ScienceUncensored	SolarWinds patches vulnerabilities that could ...		2	1.00	1612520829	3
2309	pgld00	blueteamsec	A deep-dive into the SolarWinds Serv-U SSH vul...		8	1.00	1630600245	3
2431	mfhuq7	technology	AP sources: SolarWinds hack got emails of top ...		435	0.96	1616992896	3

712 rows × 8 columns

```
solarwinds_parent_comments = solarwinds_parent_comments.drop_duplicates(subset = 'parent_comment_id')
solarwinds_parent_comments
```
[35]

rc_index	parent_comment_id	content	score	comment_datetime	post_id
27238	h4tmgo8	God needs to nerf Supply Chain attacks immedia...	46	1626021381	oi6wwa
27239	h4tnqbp	Great video, you've got a lot of potential! Ex...	17	1626022023	oi6wwa
27240	h4tqcgt	Loved your WANNACRY video, excited to see this...	9	1626023330	oi6wwa
27241	h4tvwqa	Awesome video	4	1626026103	oi6wwa
27242	h4txnrp	I have literally been looking for a solar wind...	3	1626026975	oi6wwa
...	...	...	...	...	...
220411	gsow4wk	Shocker. Well. Not really a shocker.	6	1617028497	mfhuq7
220412	gsq4dj7	But Bush/Obama did! They started this war with...	1	1617048746	mfhuq7
220413	gspo05b	That's true. We're all blessed that he was inc...	1	1617041266	mfhuq7
220414	gsqgnwd	Has Biden done anything?	3	1617054551	mfhuq7
220415	gsx6aiw	Yup. US military and both Dems/Republicans in ...	1	1617202880	mfhuq7

30071 rows × 5 columns

```
solarwinds_reddit_replies = solarwinds_reddit_replies.drop_duplicates(subset ='reply_id')
solarwinds_reddit_replies
```
[36]

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
78198	h4tmpkk	I second this 😍 as it stands, it's incredibly OP	12	1626021505	h4tmgo8	oi6wwa
78199	h4tny9g	Thank you so much!	6	1626022133	h4tnqbp	oi6wwa
78200	h4tqwav	Thank you so much :') looking forward for some...	3	1626023609	h4tqcgt	oi6wwa
78201	h4y4hkj	It was great, informative and entertaining. I ...	2	1626113807	h4tqcgt	oi6wwa
78202	h50sgsg	Oh I'm sorry to hear that :/ I'll amp it up ne...	1	1626171366	h4tqcgt	oi6wwa
...	...	...	...	...	...	...
629989	gsow4wk	Shocker. Well. Not really a shocker.	6	1617028497	gsorx9x	mfhuq7
629990	gsq4dj7	But Bush/Obama did! They started this war with...	1	1617048746	gsorx9x	mfhuq7
629991	gspo05b	That's true. We're all blessed that he was inc...	1	1617041266	gsorx9x	mfhuq7
629992	gsqgnwd	Has Biden done anything?	3	1617054551	gsorx9x	mfhuq7
629993	gsx6aiw	Yup. US military and both Dems/Republicans in ...	1	1617202880	gsorx9x	mfhuq7

22737 rows × 6 columns

Remove by date

https://www.csoonline.com/article/570537/the-solarwinds-hack-timeline-who-knew-what-and-when.html

```
solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies = remove_by_date(1606176000, solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies)
print(np.shape(solarwinds_posts))
print(np.shape(solarwinds_parent_comments))
print(np.shape(solarwinds_reddit_replies))
```
[37]

```
(636, 8)
(24960, 5)
(19855, 6)
```

Remove off topic entries

```
solarwinds_titles = solarwinds_posts['title']
solarwinds_titles
```
[38]

```
rp_index
1360    SolarWinds hack was 'largest and most sophisti...
1361    Microsoft says it found 1,000-plus developers'...
1362    Microsoft says it found 1,000-plus developers'...
1363    Biden is considering Russian financial sanctio...
1364    No One Knows How Deep Russia's Hacking Rampage...
                              ...
2304    SolarWinds patches vulnerabilities that could ...
2305    SolarWinds Patches Four New Vulnerabilities in...
2308    SolarWinds patches vulnerabilities that could ...
2309    A deep-dive into the SolarWinds Serv-U SSH vul...
2431    AP sources: SolarWinds hack got emails of top ...
Name: title, Length: 636, dtype: object
```

```
solarwinds_labels = ["SolarWinds attack", "SolarWinds cyber attack", "SolarWinds hack", "SolarWinds exploit", "SolarWinds data breach"]
solarwinds_labels
```
[39]

```
['SolarWinds attack',
 'SolarWinds cyber attack',
 'SolarWinds hack',
 'SolarWinds exploit',
 'SolarWinds data breach']
```

```
solarwinds_off_topic = get_off_topic(solarwinds_titles, solarwinds_labels)
```
[40]

```
solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies = remove_off_topic(solarwinds_off_topic, solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies)
print(np.shape(solarwinds_posts))
print(np.shape(solarwinds_parent_comments))
print(np.shape(solarwinds_reddit_replies))
```
[41]

```
(360, 8)
(16640, 5)
(13415, 6)
```

Merge datasets and final steps

- Next processes are not specific to one attack so merging justifiable

[42]
```
def merge_datasets(insomniac_df: pd.DataFrame, wannacry_df: pd.DataFrame, solarwinds_df: pd.DataFrame) -> pd.DataFrame:
    merged_df = pd.concat(objs = [insomniac_df, wannacry_df, solarwinds_df])
    return merged_df
```

[43]
```
merged_posts = merge_datasets(insomniac_posts, wannacry_posts, solarwinds_posts)
merged_posts
```

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
1	18mf8b5	TwoBestFriendsPlay	Statement from Remedy on the Insomniac Games r...		505	0.98	1703027019	1
3	18ogglo	pcgaming	Insomniac Games Releases Statement on recent r...		630	0.90	1703256167	1
4	18gkw38	SpidermanPS4	So apparently, Insomniac got hit with a ransom...		1584	0.88	1702384490	1
5	18lzgnk	Marvel	Several Insomniac Marvel games and estimate re...		208	0.88	1702984282	1
6	1b0m4vj	technology	Insomniac Games alerts employees hit by ransom...		62	0.88	1708965991	1
...	...	...	...	...	...	...	...	...
2304	lcf39o	technology	SolarWinds patches vulnerabilities that could ...		4	0.99	1612443514	3
2305	mg48j9	RedPacketSecurity	SolarWinds Patches Four New Vulnerabilities in...		1	1.00	1617066068	3
2308	ld4f7i	ScienceUncensored	SolarWinds patches vulnerabilities that could ...		2	1.00	1612520829	3
2309	pgld00	blueteamsec	A deep-dive into the SolarWinds Serv-U SSH vul...		8	1.00	1630600245	3
2431	mfhuq7	technology	AP sources: SolarWinds hack got emails of top ...		435	0.96	1616992896	3

875 rows × 8 columns

[44]
```
merged_parent_comments = merge_datasets(insomniac_parent_comments, wannacry_parent_comments, solarwinds_parent_comments)
merged_parent_comments
```

rc_index	parent_comment_id	content	score	comment_datetime	post_id
1	ke3saoh	The personal info getting leaked is honestly s...	232	1703027544	18mf8b5
2	ke3u1iu	It's one thing to leak the games for everyone ...	136	1703028248	18mf8b5
3	ke45kdu	It's a sobering reminder that the people who d...	73	1703032967	18mf8b5
4	ke3xow5	Although this may strike some readers as harsh...	75	1703029736	18mf8b5
5	ke3xbdr	It's good that Insom is at least receiving sup...	51	1703029581	18mf8b5
...	...	...	...	...	...
220411	gsow4wk	Shocker. Well. Not really a shocker.	6	1617028497	mfhuq7
220412	gsq4dj7	But Bush/Obama did! They started this war with...	1	1617048746	mfhuq7
220413	gspo05b	That's true. We're all blessed that he was inc...	1	1617041266	mfhuq7
220414	gsqgnwd	Has Biden done anything?	3	1617054551	mfhuq7
220415	gsx6aiw	Yup. US military and both Dems/Republicans in ...	1	1617202880	mfhuq7

42102 rows × 5 columns

[45]
```
merged_reddit_replies = merge_datasets(insomniac_reddit_replies, wannacry_reddit_replies, solarwinds_reddit_replies)
merged_reddit_replies
```

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
1	ke3sw9d	Right? I'll have *no* sympathy for the hackers...	162	1703027787	ke3saoh	18mf8b5
2	ke4grbb	It sucks for a couple reasons\n\n1- timing, it...	57	1703037616	ke3saoh	18mf8b5
3	ke496do	Especially since a certain crowd has already m...	47	1703034454	ke3saoh	18mf8b5
4	ke4tx0q	Honestly, I wouldn't get your hopes up for jus...	48	1703043362	ke3saoh	18mf8b5
5	ke47jzq	[removed]	-44	1703033783	ke3saoh	18mf8b5
...	...	...	...	...	...	...
629989	gsow4wk	Shocker. Well. Not really a shocker.	6	1617028497	gsorx9x	mfhuq7
629990	gsq4dj7	But Bush/Obama did! They started this war with...	1	1617048746	gsorx9x	mfhuq7
629991	gspo05b	That's true. We're all blessed that he was inc...	1	1617041266	gsorx9x	mfhuq7
629992	gsqgnwd	Has Biden done anything?	3	1617054551	gsorx9x	mfhuq7
629993	gsx6aiw	Yup. US military and both Dems/Republicans in ...	1	1617202880	gsorx9x	mfhuq7

34378 rows × 6 columns

[46]
```
print(np.shape(merged_posts))
print(np.shape(merged_parent_comments))
print(np.shape(merged_reddit_replies))
```

```
(875, 8)
(42102, 5)
(34378, 6)
```

Store the results for usage

- Majority of processing occurred above, this is for speed of access.
- No queries are required here, so csv will do.

```
merged_posts.to_csv('preprocessing_merged_posts.csv', index=True)  # Save DataFrame to a CSV file without including the index
merged_parent_comments.to_csv('preprocessing_merged_parent_comments.csv', index=True)
merged_reddit_replies.to_csv('preprocessing_merged_reddit_replies.csv', index=True)
```

```
merged_posts = pd.read_csv('preprocessing_merged_posts.csv', index_col = 0)  # Save DataFrame to a CSV file without including the index
merged_parent_comments = pd.read_csv('preprocessing_merged_parent_comments.csv', index_col = 0)
merged_reddit_replies = pd.read_csv('preprocessing_merged_reddit_replies.csv', index_col = 0)
```

Removing Non-English content

- The models that are being for sentiment analysis are trained for English language
- Translation is an alternative, but this is flawed (meaning in might get lost/mis-represented) and computationally expensive.
- Only tagged as non-english, will be removed later

```
def filter_english(df, text_column = "content", posts = False):
    for index, row in df.iterrows():
        text = row[text_column]
        text = str(text)
        try:
            language = detect(text)
            # Handle posts
            if posts == True and language != 'en':
                row[text_column] = f"language detected: {language} for: {text}"
                # Keep post anyway
            # Check if language is English
            elif language != 'en':
                row[text_column] = ''
        except:
            pass  # Skip rows where language detection fails
    return df
```

```
merged_posts = filter_english(merged_posts, 'title', posts = True)
merged_parent_comments = filter_english(merged_parent_comments)
merged_reddit_replies = filter_english(merged_reddit_replies)
```

Text Cleaning

- Removal:
  - Links
  - HTML tags
  - New lines
  - Special characters (emojis and emoticons)
  - Punctuation
  - Extra whitespace

```
def clean_text(text: str) -> str:
    # Convert text to string
    text = str(text)
    # Deleted/Removed content, content marked for deletion
    if text in ['[removed]', '[deleted]']:
        text = ''
    # Remove HTML tags
    cleaned_text = re.sub(r'<[^>]+>', '', text)
    # Remove Reddit-specific markup
    cleaned_text = re.sub(r'(\*|_)(.*?)\1', r'\2', cleaned_text)  # Markdown for italics
    cleaned_text = re.sub(r'\*\*(.*?)\*\*', r'\1', cleaned_text)  # Markdown for bold
    cleaned_text = re.sub(r'\[([^]]+)\]\(([^)]+)\)', r'\1', cleaned_text)  # Markdown for hyperlinks
    cleaned_text = re.sub(r'> (.*?)\n', r'\1', cleaned_text)       # Markdown for quoting text
    # Remove new lines
    cleaned_text = cleaned_text.replace('\n', ' ')
    # Remove links
    cleaned_text = re.sub(r'https?://\S+|www\.\S+', '', cleaned_text)
    # Remove Reddit user mentions
    cleaned_text = re.sub(r'/u/\w+', '', cleaned_text)
    # Remove subreddit references
    cleaned_text = re.sub(r'/r/\w+', '', cleaned_text)
    # Remove emojis and emoticons
    # Remove extra whitespace
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
    return cleaned_text.strip()
```

```
merged_parent_comments["content"] = merged_parent_comments["content"].apply(clean_text)
merged_reddit_replies["content"] = merged_reddit_replies["content"].apply(clean_text)
```

Spelling Correction

```
def chunk_text(text, max_words_per_chunk=20):
    # Split data up into chunks of 20
    words = text.split()
    chunks = []
    current_chunk = []
    for word in words:
        current_chunk.append(word)
        if len(current_chunk) >= max_words_per_chunk:
            chunks.append(" ".join(current_chunk))
            current_chunk = []
    if current_chunk:
        chunks.append(" ".join(current_chunk))

    return chunks

spell = Speller(fast = True)
def autocorrect_text(text):
    # Without this setting, it would not complete in a reasonable timeframe
    # Split the text into smaller chunks
    chunks = chunk_text(text)
    # Process each chunk separately
    corrected_chunks = []
    for chunk in chunks:
        corrected_chunks.append(str(spell(chunk)))
    # Combine the corrected chunks into a single string
    corrected_text = " ".join(corrected_chunks)
    return corrected_text
```

[53]

```
merged_parent_comments["content"] = merged_parent_comments["content"].apply(autocorrect_text)
merged_reddit_replies["content"] = merged_reddit_replies["content"].apply(autocorrect_text)
```

[54]

```
print(np.shape(merged_posts))
print(np.shape(merged_parent_comments))
print(np.shape(merged_reddit_replies))
```

[55]

```
(875, 8)
(42102, 5)
(34378, 6)
```

Lemmatization and stopword removal

- The text will be converted into tokens, then these steps are performed.
- Then the text is to be converted back into one String for the transformer to interpret

```
# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
nltk_stopwords = set(stopwords.words('english'))

# https://gist.github.com/sebleier/554280 - source of list of stopwords
removed_stopwords = ["all", "any", "both", "each", "few", "more", "most",
                     "other", "some", "such", "only", "own", "same", "too",
                     "very", "s", "t", "just", "now", "not", "in"]

for word in removed_stopwords:
        if word in nltk_stopwords:
            nltk_stopwords.remove(word)
```

[56]

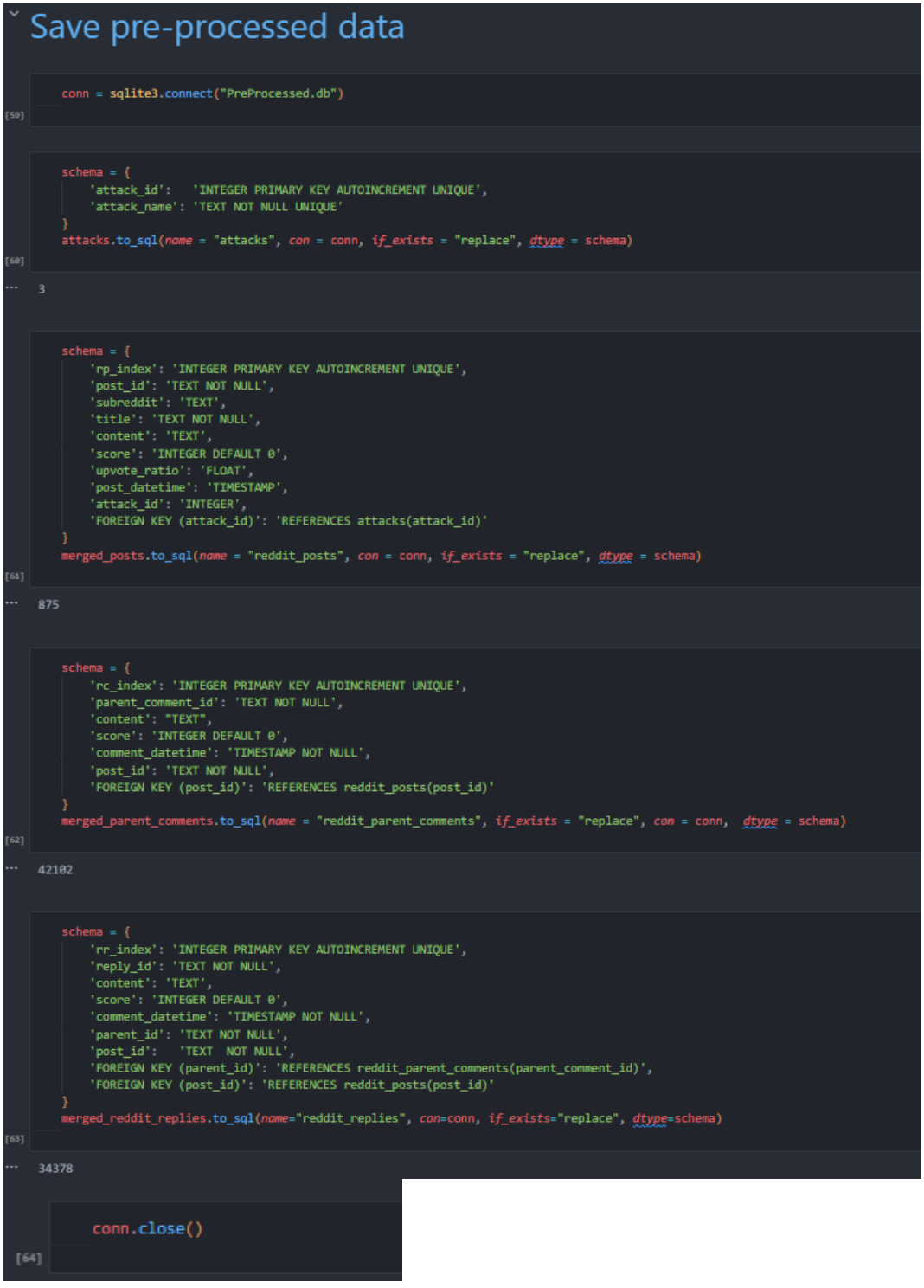
```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\seanm\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\seanm\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\seanm\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
def lemmatise_remove_stopwords(text:str) -> str:
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Lemmatize and remove stopwords
    processed_tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens if token.lower() not in stop_words and token.isalnum()]
    # Help the model handle larger inputs
    if len(processed_tokens) > 512:
        # Truncate the tokens list to contain only the first 512 tokens
        processed_tokens = processed_tokens[:512]
    # Concatenate tokens into a string
    processed_text = ' '.join(processed_tokens)
    return processed_text
```

[57]

```
merged_parent_comments["content"] = merged_parent_comments["content"].apply(lemmatise_remove_stopwords)
merged_reddit_replies["content"] = merged_reddit_replies["content"].apply(lemmatise_remove_stopwords)
```

[58]



8.3 Data Analysis.ipynb

Table of contents


```
import re
import warnings
import nltk
import sqlite3
import torch
import matplotlib
import data_analysis_functions as f
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from scipy.special import softmax
from typing import Tuple, Union
from transformers import pipeline, AutoTokenizer, AutoConfig, AutoModelForSequenceClassification, DistilBertTokenizer, DistilBertForSequenceClassification
from datasets import load_dataset
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, ConfusionMatrixDisplay
from scipy.stats import pearsonr
from tabulate import tabulate
```

```
warnings.simplefilter(action='ignore', category=FutureWarning)
pd.options.mode.chained_assignment = None  # default='warn'
```

The Model

```
SENTIMENT_MODEL = "cardiffnlp/twitter-roberta-base-sentiment-latest"
sentiment_tokenizer = AutoTokenizer.from_pretrained(SENTIMENT_MODEL)
sentiment_config = AutoConfig.from_pretrained(SENTIMENT_MODEL)
sentiment_model = AutoModelForSequenceClassification.from_pretrained(SENTIMENT_MODEL)
sentiment_model.eval()
```

```
Some weights of the model checkpoint at cardiffnlp/twitter-roberta-base-sentiment-latest were not used when initializing RobertaForSequenceClassification: ['roberta.pooler.dense.bias', 'roberta.pooler.dense.weight']
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
...
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (out_proj): Linear(in_features=768, out_features=3, bias=True)
    )
  )
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Assessing the model's performance

```
# Load the Sentiment140 dataset
dataset = load_dataset("sentiment140", split = 'test')
dataset = f.preprocess_test_dataset(dataset)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ssane\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ssane\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ssane\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

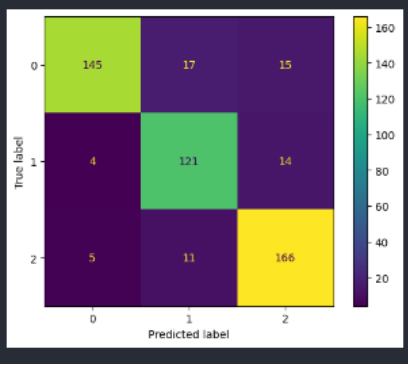
```python
test_texts = dataset["text"]
test_labels = dataset["sentiment"]
# Map sentiment labels from Sentiment140 dataset to match model output classes
def map_sentiment_label(label):
    if label == 0:  # Negative sentiment
        return 0
    elif label == 2:  # Neutral sentiment
        return 1
    elif label == 4:  # Positive sentiment
        return 2
    else:
        raise ValueError("Invalid sentiment label")

# Preprocess sentiment labels in the dataset
test_labels_mapped = [map_sentiment_label(label) for label in test_labels]
```

```python
# Tokenize the validation or test data
inputs = sentiment_tokenizer(test_texts, return_tensors="pt", padding=True, truncation=True, max_length = 512)
# Forward pass through the model to obtain predictions for validation data
with torch.no_grad():
    outputs = sentiment_model(**inputs)
    logits = outputs.logits
# Convert logits to predicted labels for validation data
pred_labels = torch.argmax(logits, dim=1).tolist()
```

```python
f.evaluate_test_dataset(test_labels_mapped, pred_labels)
```

Evaluation Metrics:

Metric	Value
Accuracy	0.86747
Precision	0.87243
Recall	0.86747
F1-score	0.86777



Load all attacks

```
conn = sqlite3.connect("PreProcessed.db")
```

```
attacks = pd.read_sql(sql = "select * from attacks",con = conn, index_col = "attack_id" )
attacks
```

attack_id	attack_name
1	Insomniac Games
2	WannaCry
3	SolarWinds

```
merged_posts, merged_parent_comments, merged_reddit_replies = f.get_all_attacks(conn)
```

```
merged_posts
```

rp_index	post_id	subreddit	title	content	score	upvote_ratio	post_datetime	attack_id
1	18mf8b5	TwoBestFriendsPlay	Statement from Remedy on the Insomniac Games r...	None	505	0.98	1703027019	1
3	18ogglo	pcgaming	Insomniac Games Releases Statement on recent r...	None	630	0.90	1703256167	1
4	18gkw38	SpidermanPS4	So apparently, Insomniac got hit with a ransom...	None	1584	0.88	1702384490	1
5	18lzgnk	Marvel	Several Insomniac Marvel games and estimate re...	None	208	0.88	1702984282	1
6	1b0m4vj	technology	Insomniac Games alerts employees hit by ransom...	None	62	0.88	1708965991	1
...	...	...	...	...	...	...	...	...
2340	18uxkw6	SpidermanPS4	Insomniac's leaked next game.	None	1685	0.89	1703991538	1
2347	18qiwww	GamingLeaksAndRumours	Insomniac leak: Demand for the remastered games	The same guy shared (the Future of PS Plus)(ht...	628	0.93	1703511713	1
2378	7kpiac	politics	US to announce North Korea behind WannaCry cyb...	None	38	0.82	1513643331	2
2380	7kphs3	worldnews	U.S. to blame North Korea for 'WannaCry' cyber...	None	77	0.80	1513643169	2
2431	mfhuq7	technology	AP sources: SolarWinds hack got emails of top ...	None	435	0.96	1616992896	3

875 rows × 8 columns

```
merged_parent_comments
```

rc_index	parent_comment_id	content	score	comment_datetime	post_id
1	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5
2	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5
3	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5
4	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5
5	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5
...	...	...	...	...	...
220411	gsow4wk	shocked well really shocked	6	1617028497	mfhuq7
220412	gsq4dj7	started war operation olympic game look stuxne...	1	1617048746	mfhuq7
220413	gspo05b	true blessed incredibly ineffective incapable ...	1	1617041266	mfhuq7
220414	gsqgnwd	biden done anything	3	1617054551	mfhuq7
220415	gsx6aiw	yup military congress openly advocating manhat...	1	1617202880	mfhuq7

42102 rows × 5 columns

```
merged_reddit_replies
```

rr_index	reply_id	content	score	comment_datetime	parent_id	post_id
1	ke3sw9d	right sympathy hacker sony come got ruin life ...	162	1703027787	ke3saoh	18mf8b5
2	ke4grbb	suck couple reason timing almost christmas peo...	57	1703037616	ke3saoh	18mf8b5
3	ke496do	especially since certain crowd already made mi...	47	1703034454	ke3saoh	18mf8b5
4	ke4tx0q	honestly get hope justice ransomware attack mo...	48	1703043362	ke3saoh	18mf8b5
5	ke47jzq		-44	1703033783	ke3saoh	18mf8b5
...	...	...	...	...	...	...
629989	gsow4wk	shocked well really shocked	6	1617028497	gsorx9x	mfhuq7
629990	gsq4dj7	started war operation olympic game look stuxne...	1	1617048746	gsorx9x	mfhuq7
629991	gspo05b	true blessed incredibly ineffective incapable ...	1	1617041266	gsorx9x	mfhuq7
629992	gsqgnwd	biden done anything	3	1617054551	gsorx9x	mfhuq7
629993	gsx6aiw	yup military congress openly advocating manhat...	1	1617202880	gsorx9x	mfhuq7

34378 rows × 6 columns

```
insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies = f.get_attack_from_merged(merged_posts, merged_parent_comments, merged_reddit_replies, 1)
```

```
wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies = f.get_attack_from_merged(merged_posts, merged_parent_comments, merged_reddit_replies, 2)
```

```
solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies = f.get_attack_from_merged(merged_posts, merged_parent_comments, merged_reddit_replies, 3)
```

Get sentiment scores for all

Insomniac Games

```
insomniac_all_comments = f.get_all_comments_df(insomniac_posts, insomniac_parent_comments, insomniac_reddit_replies)
insomniac_all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id
0	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5	TwoBestFriendsPlay	1
1	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5	TwoBestFriendsPlay	1
2	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5	TwoBestFriendsPlay	1
3	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5	TwoBestFriendsPlay	1
4	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5	TwoBestFriendsPlay	1
...	...	...	...	...	...	...	...
7985	kf34vyr	yea make weird decision like would port dr 1 s...	2	1703654426	18qiwww	GamingLeaksAndRumours	1
7986	kex6d52	pp multiplayer p huh eh hoping first party p h...	7	1703547627	18qiwww	GamingLeaksAndRumours	1
7987	kewakw9	10 ps4 version fan newcomer like post suggests	3	1703533479	18qiwww	GamingLeaksAndRumours	1
7988	keypzjt	px remaster	3	1703577602	18qiwww	GamingLeaksAndRumours	1
7989	keweczm	yeah know meant far main reason bought px5 pla...	-1	1703535130	18qiwww	GamingLeaksAndRumours	1

7622 rows × 7 columns

```
insomniac_all_comments = f.process_df(insomniac_all_comments, 'content', sentiment_tokenizer, sentiment_config, sentiment_model)
```

```
insomniac_all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id	sentiment_score	determined_sentiment
0	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5	TwoBestFriendsPlay	1	-0.885556	negative
1	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5	TwoBestFriendsPlay	1	-0.840470	negative
2	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5	TwoBestFriendsPlay	1	-0.289011	neutral
3	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5	TwoBestFriendsPlay	1	-0.461533	negative
4	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5	TwoBestFriendsPlay	1	-0.553312	negative
...	...	...	...	...	...	...	...	...	...
7985	kf34vyr	yea make weird decision like would port dr 1 s...	2	1703654426	18qiwww	GamingLeaksAndRumours	1	-0.332805	neutral
7986	kex6d52	pp multiplayer p huh eh hoping first party p h...	7	1703547627	18qiwww	GamingLeaksAndRumours	1	0.114489	neutral
7987	kewakw9	10 ps4 version fan newcomer like post suggests	3	1703533479	18qiwww	GamingLeaksAndRumours	1	0.019242	neutral
7988	keypzjt	px remaster	3	1703577602	18qiwww	GamingLeaksAndRumours	1	0.168619	neutral
7989	keweczm	yeah know meant far main reason bought px5 pla...	-1	1703535130	18qiwww	GamingLeaksAndRumours	1	0.557741	positive

7622 rows × 9 columns

WannaCry Attack

```
wannacry_all_comments = f.get_all_comments_df(wannacry_posts, wannacry_parent_comments, wannacry_reddit_replies)
wannacry_all_comments = f.process_df(wannacry_all_comments, 'content', sentiment_tokenizer, sentiment_config, sentiment_model)
```

```
wannacry_all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id	sentiment_score	determined_sentiment
0	dhvr48z	first thanks many sure whirlwind publicity lac...	3420	1495451975	6cmmdf	IAmA	2	-0.631646	negative
1	dhvrx7q	receive job offering government	2220	1495453860	6cmmdf	IAmA	2	-0.000487	neutral
2	dhvr4pb	px setup spec also software provide best isola...	2178	1495452007	6cmmdf	IAmA	2	0.334703	neutral
3	dhvr463	good resource way learn cyber security	1336	1495451969	6cmmdf	IAmA	2	0.725165	positive
4	dhvs8md	hell ir good work kil switch question 2sec4u g...	1327	1495454550	6cmmdf	IAmA	2	0.865946	positive
...	...	...	...	...	...	...	...	...	...
17350	drhc468	let think poor country barely internet access ...	1	1513706744	7kphs3	worldnews	2	-0.694861	negative
17351	drhd0ss	war sell fuck nuclear proliferation amount cas...	1	1513707674	7kphs3	worldnews	2	-0.821200	negative
17352	drhdi6v	war never sell unless attacked directly n alre...	2	1513708153	7kphs3	worldnews	2	-0.732043	negative
17353	drhmpsd	american public bloodthirsty af show far need ...	2	1513717463	7kphs3	worldnews	2	-0.892421	negative
17354	drhdum3	n already nuclear weapon yes keep existing	1	1513708500	7kphs3	worldnews	2	-0.024509	neutral

15999 rows × 9 columns

SolarWinds Attack

```
solarwinds_all_comments = f.get_all_comments_df(solarwinds_posts, solarwinds_parent_comments, solarwinds_reddit_replies)
solarwinds_all_comments = f.process_df(solarwinds_all_comments, 'content', sentiment_tokenizer, sentiment_config, sentiment_model)
```

```
solarwinds_all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id	sentiment_score	determined_sentiment
0	h4tmgo8	god need ner supply chain attack immediately far	46	1626021381	oi6wwa	cybersecurity	3	-0.614974	negative
1	h4tnqbp	great video got lot potential excited see content	17	1626022023	oi6wwa	cybersecurity	3	0.980981	positive
2	h4tqcgt	loved wannacry video excited see one	9	1626023330	oi6wwa	cybersecurity	3	0.982828	positive
3	h4tvwqa	awesome video	4	1626026103	oi6wwa	cybersecurity	3	0.940209	positive
4	h4txnrp	literally looking solar wind doc stored check	3	1626026975	oi6wwa	cybersecurity	3	0.031917	neutral
...	...	...	...	...	...	...	...	...	...
16624	gsow4wk	shocked well really shocked	6	1617028497	mfhuq7	technology	3	-0.840873	negative
16625	gsq4dj7	started war operation olympic game look stuxne...	1	1617048746	mfhuq7	technology	3	-0.234056	neutral
16626	gspo05b	true blessed incredibly ineffective incapable ...	1	1617041266	mfhuq7	technology	3	-0.904313	negative
16627	gsqgnwd	biden done anything	3	1617054551	mfhuq7	technology	3	-0.025591	neutral
16628	gsx6aiw	yup military congress openly advocating manhat...	1	1617202880	mfhuq7	technology	3	-0.243846	neutral

15336 rows × 9 columns

Store the results for usage

- Majority of processing occurred above, this is for speed of access.
- No queries are required here, so csv will do.

```
insomniac_all_comments.to_csv('insomniac_all_comments.csv', index=True)
wannacry_all_comments.to_csv('wannacry_all_comments.csv', index=True)
solarwinds_all_comments.to_csv('solarwinds_all_comments.csv', index=True)
```

```
insomniac_all_comments = pd.read_csv('insomniac_all_comments.csv', index_col = 0)
wannacry_all_comments = pd.read_csv('wannacry_all_comments.csv', index_col = 0)
solarwinds_all_comments = pd.read_csv('solarwinds_all_comments.csv', index_col = 0)
```

```
all_comments = pd.concat([insomniac_all_comments, wannacry_all_comments, solarwinds_all_comments])
all_comments
```

	comment_id	content	score	comment_datetime	post_id	subreddit	attack_id	sentiment_score	determined_sentiment
0	ke3saoh	personal info getting leaked honestly scary th...	232	1703027544	18mf8b5	TwoBestFriendsPlay	1	-0.885556	negative
1	ke3u1iu	one thing leak game everyone talk leaking priv...	136	1703028248	18mf8b5	TwoBestFriendsPlay	1	-0.840470	negative
2	ke45kdu	sobering reminder people thing criminal howeve...	73	1703032967	18mf8b5	TwoBestFriendsPlay	1	-0.289011	neutral
3	ke3xow5	although may strike reader harsh believe nearl...	75	1703029736	18mf8b5	TwoBestFriendsPlay	1	-0.461533	negative
4	ke3xbdr	good insom least receiving support wake whatev...	51	1703029581	18mf8b5	TwoBestFriendsPlay	1	-0.553312	negative
...	...	...	...	...	...	...	...	...	...
16624	gsow4wk	shocked well really shocked	6	1617028497	mfhuq7	technology	3	-0.840873	negative
16625	gsq4dj7	started war operation olympic game look stuxne...	1	1617048746	mfhuq7	technology	3	-0.234056	neutral
16626	gspo05b	true blessed incredibly ineffective incapable ...	1	1617041266	mfhuq7	technology	3	-0.904313	negative
16627	gsqgnwd	biden done anything	3	1617054551	mfhuq7	technology	3	-0.025591	neutral
16628	gsx6aiw	yup military congress openly advocating manhat...	1	1617202880	mfhuq7	technology	3	-0.243846	neutral

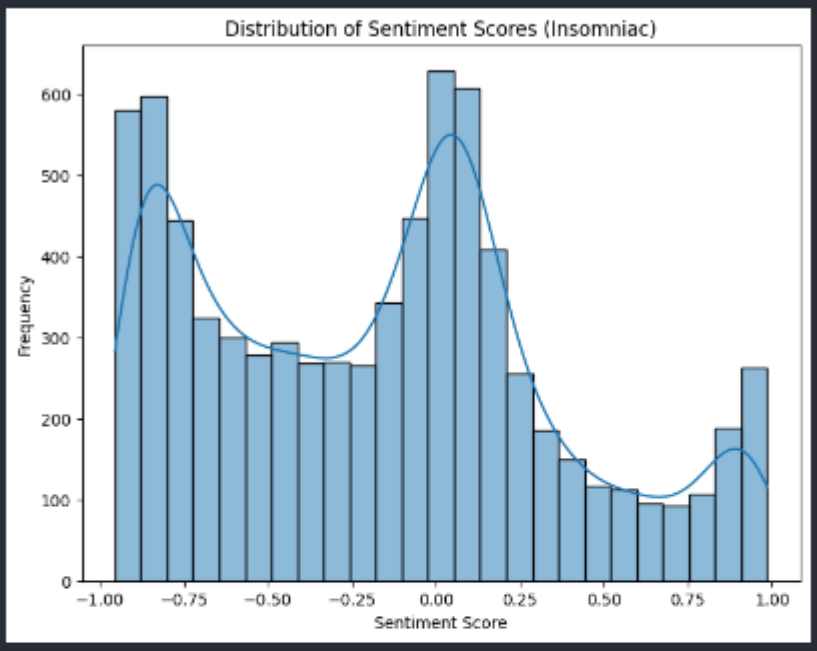
38957 rows × 9 columns

Results

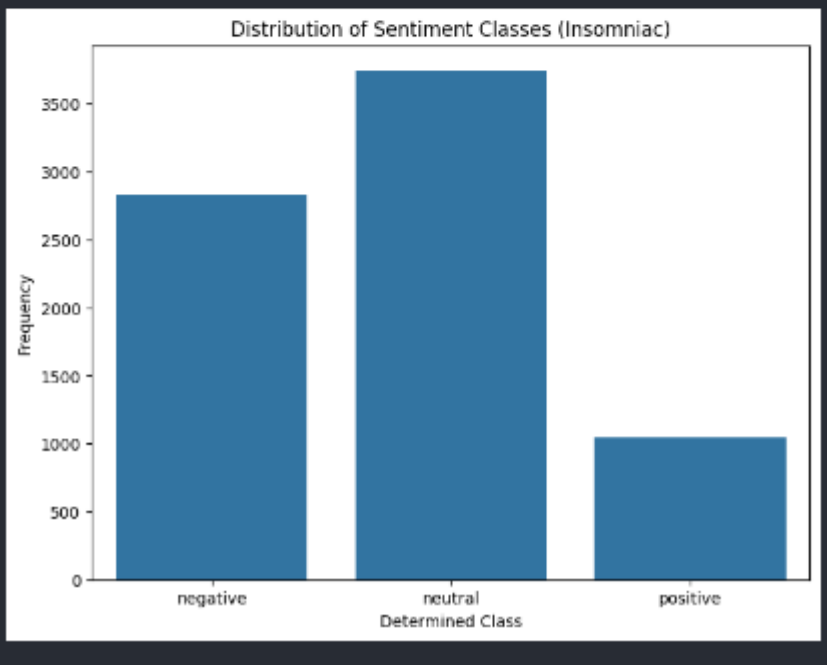
Distribution of Sentiment Scores and Classes

Insomniac Games

```
f.display_sentiment_distribution_hist(insomniac_all_comments.copy(), "Insomniac")
```

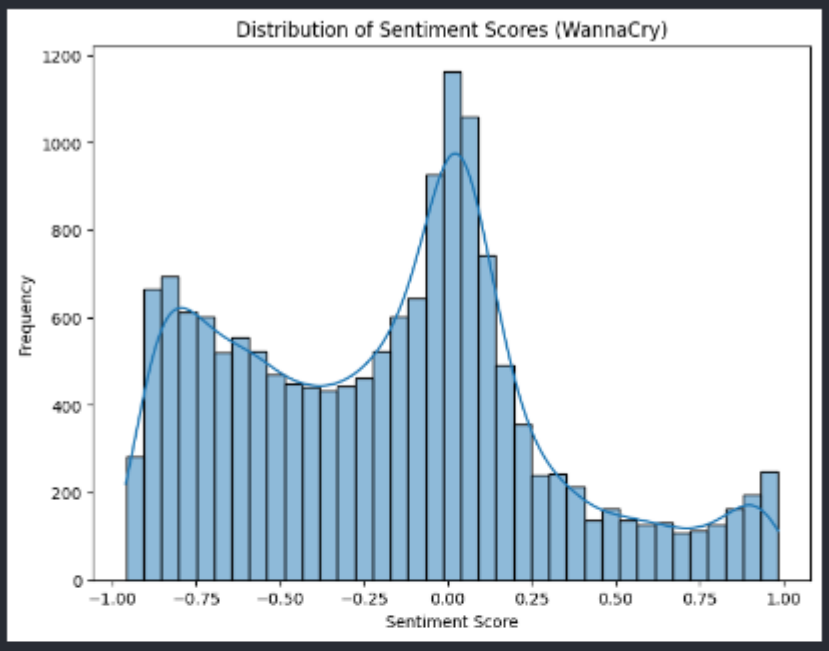



```
f.display_determined_sentiment_distribution_countplot(insomniac_all_comments.copy(), 'Insomniac')
```

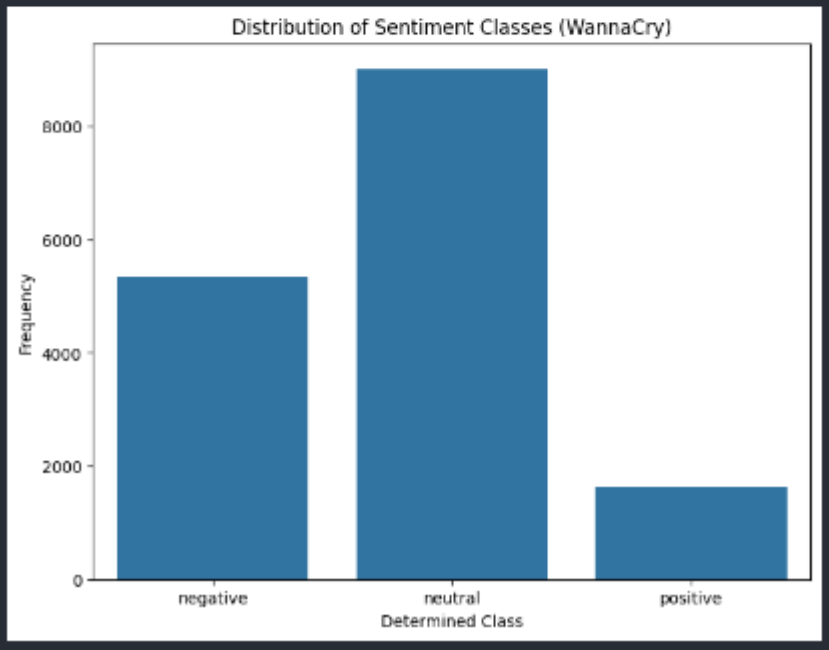


WannaCry

```
f.display_sentiment_distribution_hist(wannacry_all_comments.copy(), "WannaCry")
```

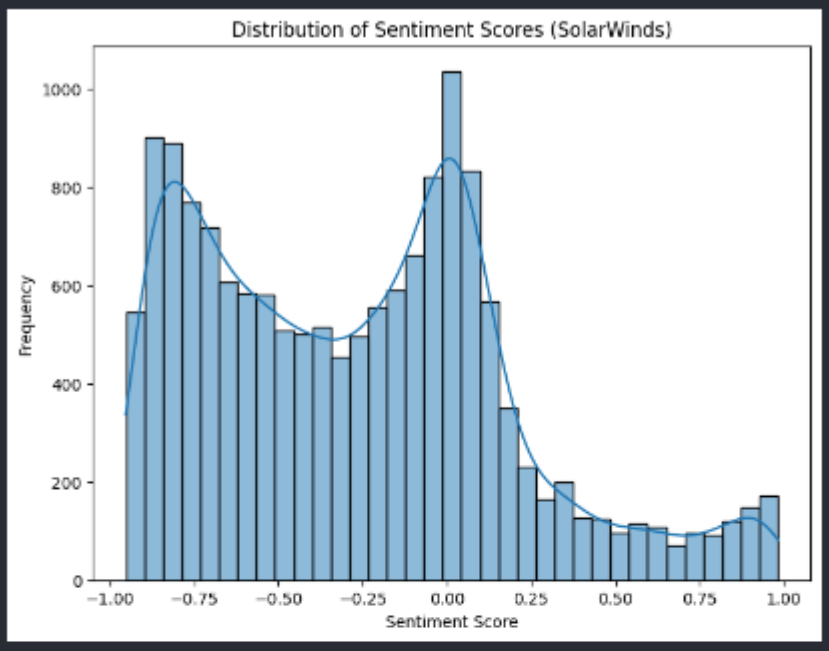



```
f.display_determined_sentiment_distribution_countplot(wannacry_all_comments.copy(), 'WannaCry')
```

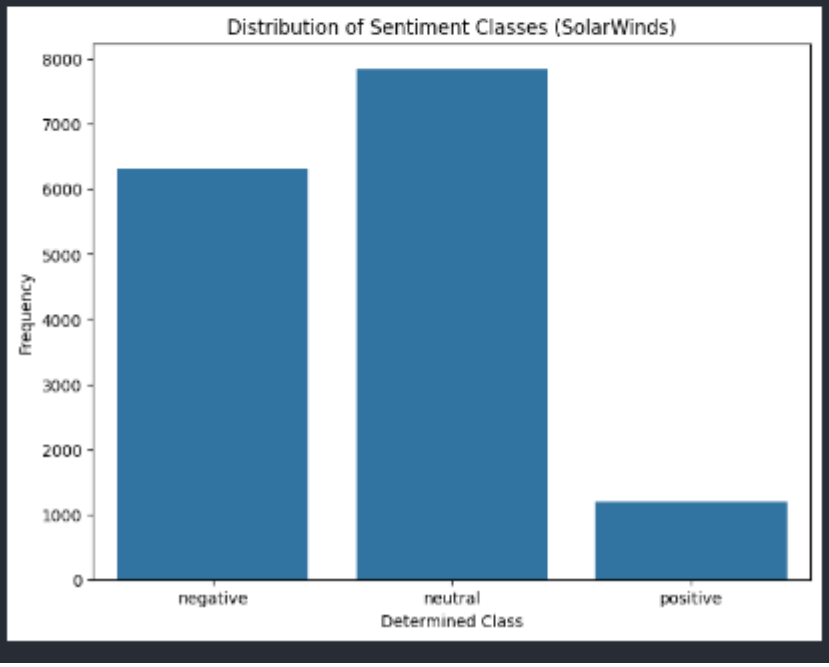


Solarwinds

```
f.display_sentiment_distribution_hist(solarwinds_all_comments.copy(), "SolarWinds")
```

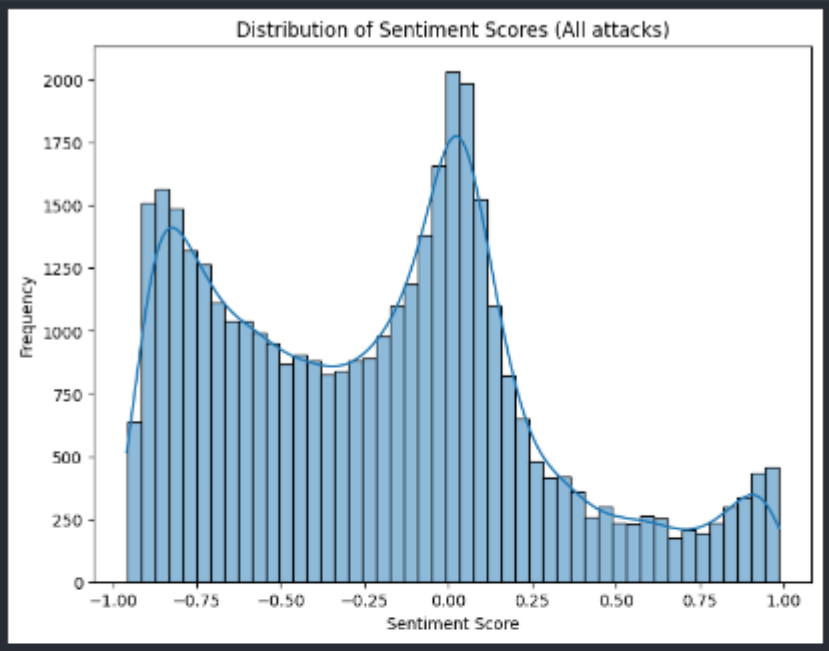



```
f.display_determined_sentiment_distribution_countplot(solarwinds_all_comments.copy(), 'SolarWinds')
```

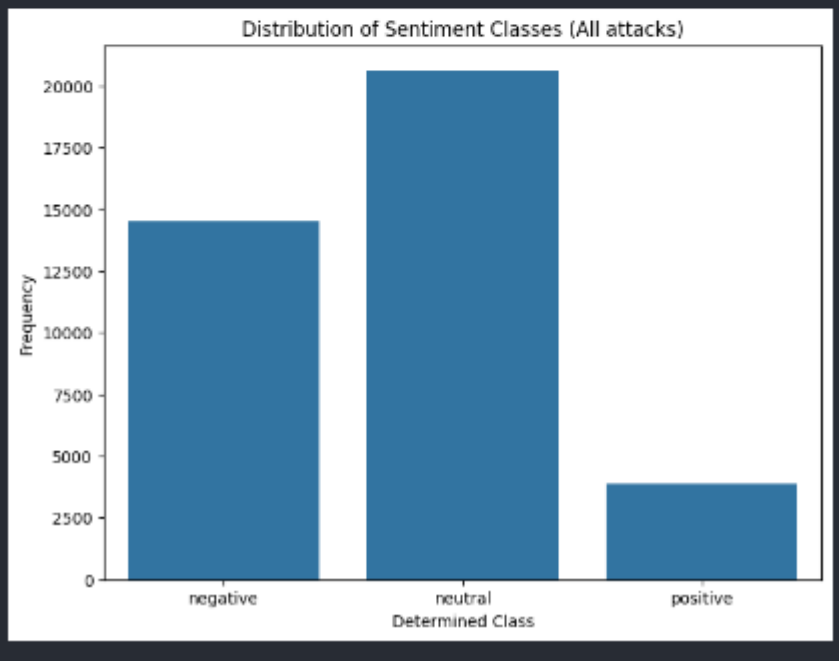


All attacks

```
f.display_sentiment_distribution_hist(all_comments.copy(), "All attacks")
```



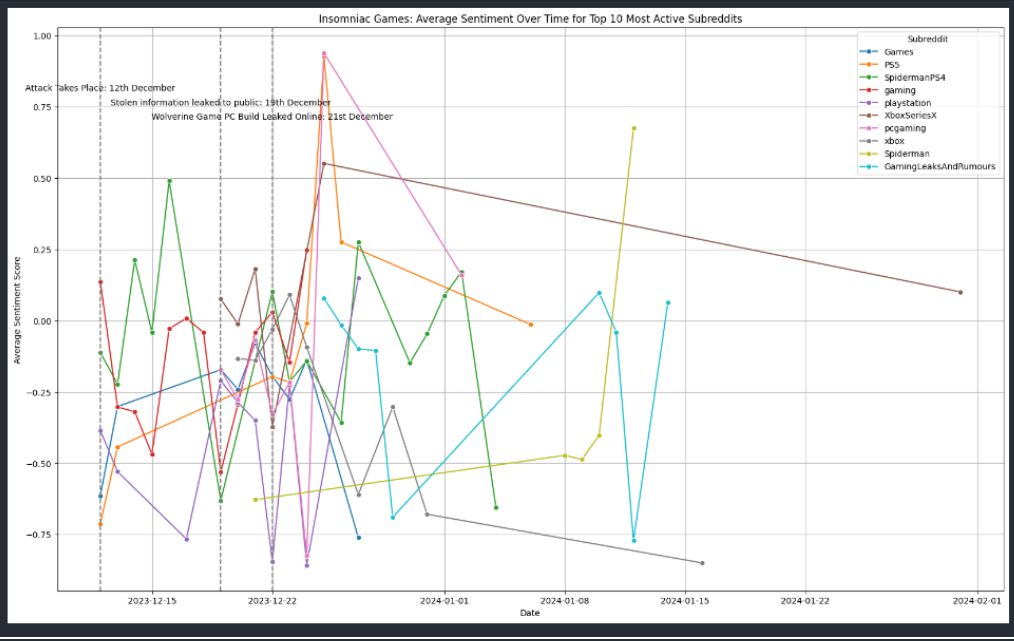

```
f.display_determined_sentiment_distribution_countplot(all_comments.copy(), 'All attacks')
```



Sentiment Scores Over Time - Across Subreddits

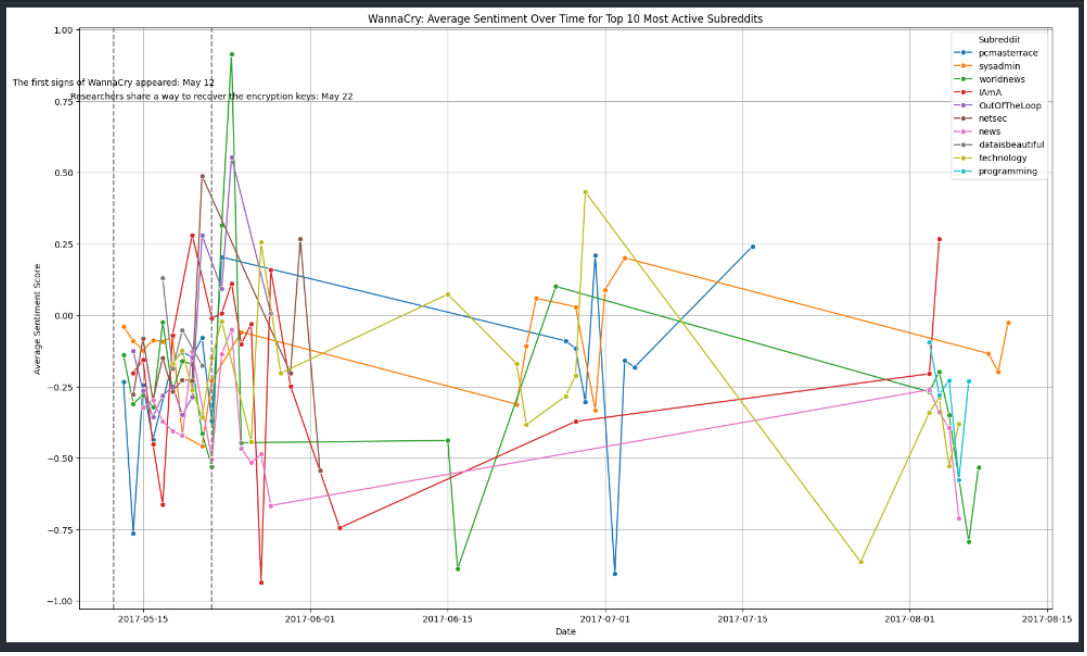
Insomniac Games

```
insomniac_dates = {
    '2023-12-12': 'Attack Takes Place: 12th December',
    '2023-12-19': 'Stolen information leaked to public: 19th December',
    '2023-12-22': 'Wolverine Game PC Build Leaked Online: 21st December'
}
f.display_subreddit_sentiment_over_time_line(insomniac_all_comments.copy(), "Insomniac Games", insomniac_dates, filter_period = "3 Month")
```



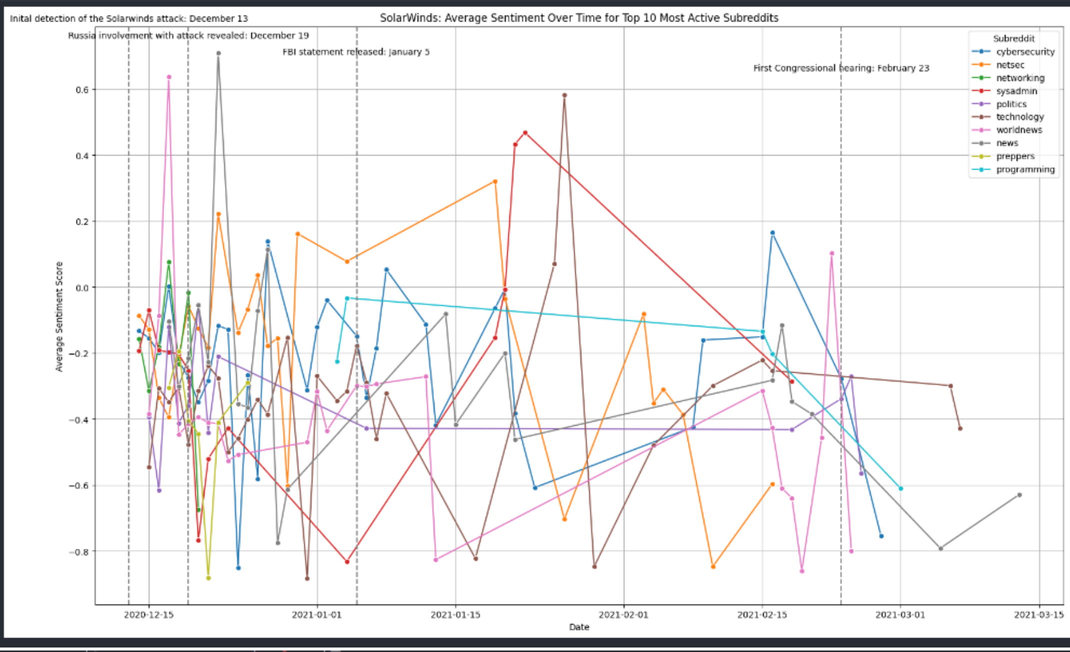
WannaCry

```
wannacry_dates = {
    '2017-05-12': 'The first signs of WannaCry appeared: May 12',
    '2017-05-22': 'Researchers share a way to recover the encryption keys: May 22'
}
f.display_subreddit_sentiment_over_time_line(wannacry_all_comments.copy(), "WannaCry", wannacry_dates, filter_period = "3 Month")
```



Solarwinds

```python
solarwinds_dates = {
    '2020-12-13': 'Inital detection of the Solarwinds attack: December 13',
    '2020-12-19': 'Russia involvement with attack revealed: December 19',
    '2021-01-05': 'FBI statement released: January 5',
    '2021-02-23': 'First Congressional hearing: February 23'
}
f.display_subreddit_sentiment_over_time_line(solarwinds_all_comments.copy(), "SolarWinds", solarwinds_dates, filter_period = "3 Month")
```

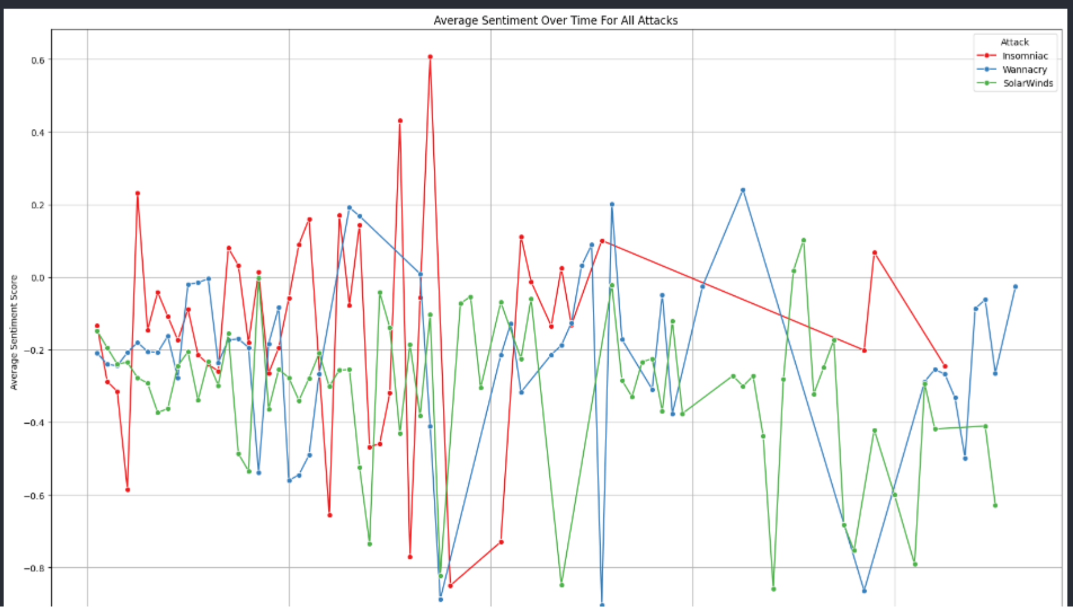


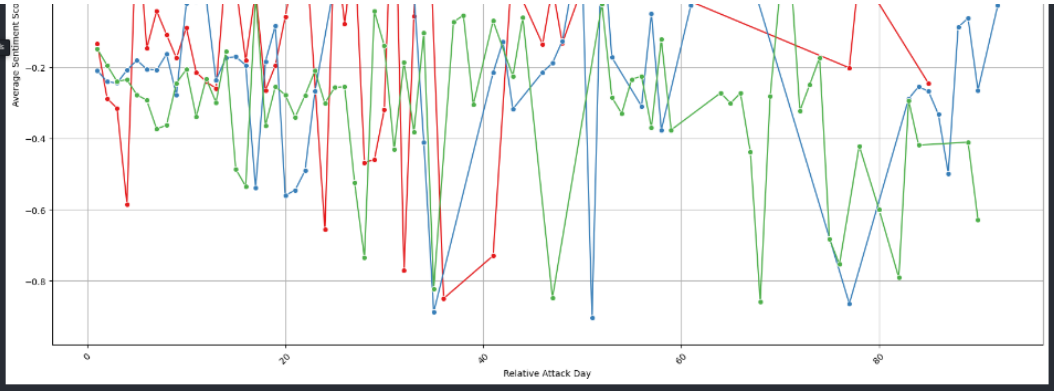
Comparing All Attacks

Sentiments in first 2 months of attack

```python
# Insomniac, WannaCry, Solarwinds
start_dates = {'1': '2023-12-12', '2': '2017-05-17','3': '2020-12-01'}
f.compare_avg_sentiment_three_months_line(all_comments.copy(), start_dates)
```

```
C:\Users\seanm\AppData\Local\Temp\ipykernel_18656\2809624509.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  attack_data['relative_date'] = (attack_data['comment_datetime'] - filter_start_date).dt.days + 1
C:\Users\seanm\AppData\Local\Temp\ipykernel_18656\2809624509.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  attack_data['relative_date'] = (attack_data['comment_datetime'] - filter_start_date).dt.days + 1
C:\Users\seanm\AppData\Local\Temp\ipykernel_18656\2809624509.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  attack_data['relative_date'] = (attack_data['comment_datetime'] - filter_start_date).dt.days + 1
C:\Users\seanm\AppData\Local\Temp\ipykernel_18656\2809624509.py:41: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(x='relative_date', y='sentiment_score', hue='attack_id', data=plot_data, marker='o', palette='Set1', ci=None)
```





Relationship between score and sentiment

```
f.display_relationship_sentiment_v_score(all_comments.copy())
```

Relationship between Sentiment Score and Entry Score

Metric	Value
Correlation Coefficient	-0.005100692404649605
Mean Entry Score by Sentiment Category	{'negative': 16.708, 'neutral': 15.412, 'positive': 12.032}
Standard Deviation of Entry Scores by Sentiment Category	{'negative': 164.113, 'neutral': 153.396, 'positive': 82.32}

```
Interpretation:
In conclusion, there is a weak or no relationship between sentiment score and entry score. However, entries with negative sentiment tend to have slightly higher average scores.
```

Topics of Discussion

Most active posts

```
pd.set_option('display.max_colwidth', None)
```

```
f.get_top_10_active_posts(merged_posts, insomniac_all_comments)
```

	post_id	comment_count	average_sentiment_score	title
0	18mi5ep	934	-0.132897	Xbox is not allowed X-Men games until 2036, according to illegally breached Insomniac Games materials
1	18luxxa	858	-0.180325	More than a terabyte of Insomniac Games' internal data has been leaked by hacker group, including internal HR documents, 'Wolverine' game files, and timeline of upcoming projects
2	18n8nws	675	-0.083512	Illegally breached data from Insomniac Games reveals that Xbox is not allowed X-men games until 2036 due to Sony signing a deal with Marvel
3	18hjfx6	536	-0.304509	Insomniac Games hacked, Wolverine screenshots and employee data stolen - Ransom of 2 Million Dollars
4	191swtu	500	-0.475445	Insomniac Spider-Man "fans" harassing MJ's video game model
5	18og8im	395	-0.186226	Insomniac Games releases statement regarding recent leaks
6	18m5g5f	393	0.052479	Insomniac Games Has Made Just $567 off Sunset Overdrive
7	18gkw38	388	-0.128790	So apparently, Insomniac got hit with a ransomware attack, wolverine got leaked, but there is no DLC or NG+ for SM2 yet
8	18n8mdj	387	-0.034999	Xbox is not allowed X-Men games until 2036, according to illegally breached Insomniac Games materials.
9	18og8n0	357	-0.221835	Insomniac Games releases statement regarding recent leaks

```
f.get_top_10_active_posts(merged_posts, wannacry_all_comments)
```

	post_id	comment_count	average_sentiment_score	title
0	6cmmdf	2399	-0.005824	IamA the "accidental hero" who helped stop the WannaCry attack AMA!
1	6b7gd4	2324	-0.323143	Microsoft president blasts NSA for its role in 'WannaCry' computer ransom attack
2	6c495x	1147	-0.164970	WannaCry Ransomware Decryption Tool Released; Unlock Files Without Paying Ransom
3	6b5j9h	1056	-0.161477	[AMA Request] The 22 year old hacker who stopped the recent ransomware attacks on British hospitals.
4	6bacmd	842	-0.107104	WannaCry Megathread
5	6jsxq5	538	-0.252061	Huge cyber attack spreading across the world in potential repeat of 'Wannacry' hack
6	6rg0pj	442	-0.281171	FBI arrests WannaCry hero Marcus Hutchins in Las Vegas over malware claims
7	6rdlcy	402	-0.245931	Researcher Who Stopped WannaCry Ransomware Detained in US After Def Con
8	6b0qa0	377	-0.300400	It's Not Over, WannaCry 2.0 Ransomware Just Arrived With No 'Kill-Switch'
9	6bs66v	331	-0.180309	Balance of the bitcoin wallets tied to the WannaCry ransomware attack [OC]

```
f.get_top_10_active_posts(merged_posts, solarwinds_all_comments)
```

	post_id	comment_count	average_sentiment_score	title
0	kgwi6h	2012	-0.400253	Biden is considering Russian financial sanctions or other retaliatory action in response to the SolarWinds hack
1	kg942d	1063	-0.343802	SolarWinds hack may be much worse than originally feared
2	lk37qq	1027	-0.332474	SolarWinds hack was 'largest and most sophisticated attack' ever: Microsoft president
3	mrcdah	797	-0.371458	U.S. to sanction Russia for alleged election interference, SolarWinds hack
4	kdx0p9	605	-0.329802	No One Knows How Deep Russia's Hacking Rampage Goes. A supply chain attack against IT company SolarWinds has exposed as many as 18,000 companies to Cozy Bear's attacks.
5	kdvaov	364	-0.116680	SolarWinds use the password 'solarwinds123' on their Update Servers
6	kebhgj	331	-0.259483	SolarWinds writes blog describing open-source software as vulnerable because anyone can update it with malicious code - Ages like fine wine
7	kcrd7y	325	-0.160192	According to FireEye, SolarWinds Orion platform allegedly compromised by foreign hackers.
8	keo19e	273	-0.300978	Preppers: SolarWinds should be a wake up call for you
9	lkt35u	251	-0.275077	Microsoft says it found 1,000-plus developers' fingerprints on the SolarWinds attack

```
pd.reset_option('display.max_colwidth')
```

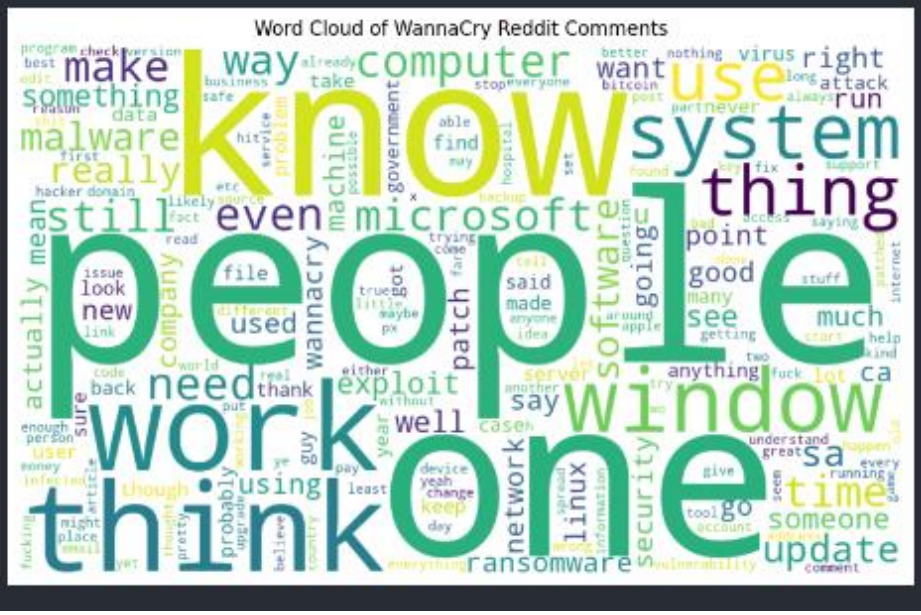
Wordclouds

Insomniac Games

```
f.display_wordcloud(insomniac_all_comments.copy(), "Insomniac Games Leak Reddit Comments", 'comments')
```

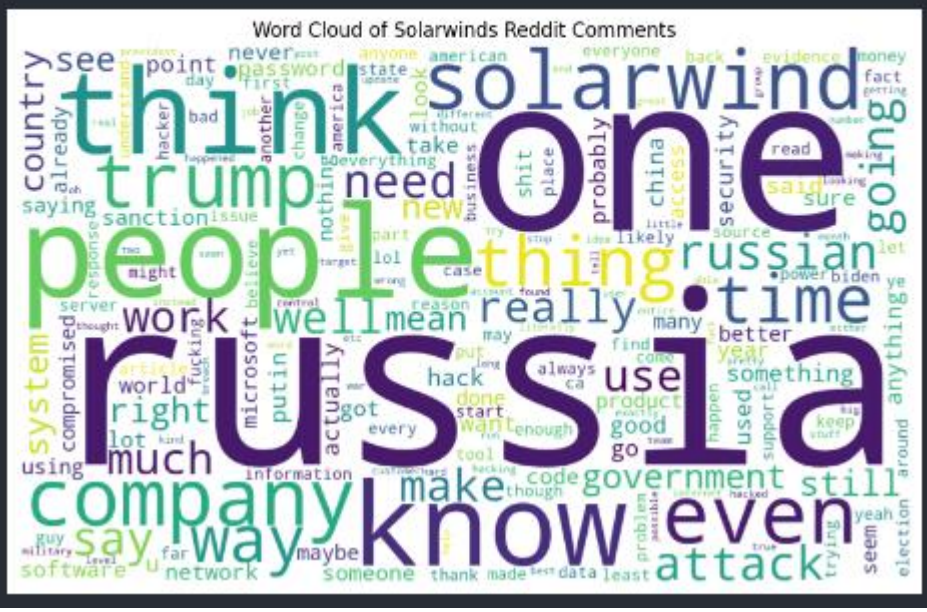
WannaCry

```
f.display_wordcloud(wannacry_all_comments.copy(), "WannaCry Reddit Comments", 'comments')
```



SolarWinds

```
f.display_wordcloud(solarwinds_all_comments.copy(), "Solarwinds Reddit Comments", 'comments')
```



8.4 data_analysis_functions.py

```
import sqlite3
import re
import torch
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import nltk
from scipy.special import softmax
from typing import Tuple, Union
from transformers import pipeline, AutoTokenizer, AutoConfig, AutoModelForSequenceClassification, DistilBertTokenizer, DistilBertForSequenceClassification
from matplotlib.ticker import MultipleLocator
from wordcloud import WordCloud
from typing import Tuple
from transformers import pipeline
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from autocorrect import Speller
from langdetect import detect
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from wordcloud import WordCloud
from matplotlib.ticker import MultipleLocator
from datasets import load_dataset
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, ConfusionMatrixDisplay
from scipy.stats import pearsonr

# ----------------------------------------------------------------------------
#  Manipulating Data and getting dfs - (get_)
# ----------------------------------------------------------------------------

def get_attack_from_merged(posts_df: pd.DataFrame, comments_df: pd.DataFrame, replies_df: pd.DataFrame,attack_id: int) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = posts_df[posts_df['attack_id'] == attack_id]
    comments = comments_df[comments_df['post_id'].isin(posts['post_id'])]
    replies = replies_df[replies_df['post_id'].isin(posts['post_id'])]
    return posts, comments, replies

def get_all_attacks(conn) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    posts = pd.read_sql(sql = f"select * from reddit_posts", con = conn)
    comments = pd.read_sql(sql = f"select * from reddit_parent_comments", con = conn)
    replies = pd.read_sql(sql = f"select * from reddit_replies", con = conn)
    posts.set_index(posts.columns[0], inplace=True, drop=True)
    comments.set_index(comments.columns[0], inplace=True, drop=True)
    replies.set_index(replies.columns[0], inplace=True, drop=True)
    return posts, comments, replies

def get_all_comments_df(posts: pd.DataFrame, parent_comments: pd.DataFrame, reddit_replies: pd.DataFrame) -> pd.DataFrame:
    # Merge dfs
    comments_df = pd.concat([parent_comments, reddit_replies], ignore_index=False)
    # Handle empty values in p_c_id (from replies)
    comments_df['parent_comment_id'] = comments_df['parent_comment_id'].fillna(comments_df['reply_id'])
    # Rename p_c_id, remove now irrelevant comments
    comments_df.rename(columns={'parent_comment_id': 'comment_id'}, inplace=True)
    comments_df.drop(columns=['reply_id'], inplace=True)
    comments_df.drop(columns=['parent_id'], inplace=True)
    # Reset the index of the DataFrame
    comments_df.reset_index(drop=True, inplace=True)
    # Add relevant subreddit values
    post_subreddit_dict = posts.set_index('post_id')['subreddit'].to_dict()
    comments_df['subreddit'] = comments_df['post_id'].map(post_subreddit_dict)
    # Do the same for attack id
    post_attack_dict = posts.set_index('post_id')['attack_id'].to_dict()
    comments_df['attack_id'] = comments_df['post_id'].map(post_attack_dict)
    # Remove empty comments
    comments_df = comments_df[comments_df['content'] != '']
    # Ensure no duplicates
    comments_df = comments_df.drop_duplicates(subset = 'comment_id')
    return comments_df

def get_top_10_active_posts(posts, comments):
    # Count occurrences of each post_id in comments
    post_counts = comments['post_id'].value_counts().reset_index()
    post_counts.columns = ['post_id', 'comment_count']

    # Calculate average sentiment_score for each post
    avg_sentiment_scores = comments.groupby('post_id')['sentiment_score'].mean().reset_index()
    avg_sentiment_scores.rename(columns={'sentiment_score': 'average_sentiment_score'}, inplace=True)

    # Merge post counts with average sentiment scores and posts DataFrame to get titles
    top_10_posts = pd.merge(post_counts.head(10), avg_sentiment_scores, on='post_id', how='left')
    top_10_posts = pd.merge(top_10_posts, posts[['post_id', 'title']], on='post_id', how='left')
    # Adjust display settings to show full text in DataFrame cells
    return top_10_posts

# ----------------------------------------------------------------------------
# Process data - _process
# ----------------------------------------------------------------------------

def process_comment_sentiment(text, sentiment_tokenizer, sentiment_config, sentiment_model) -> pd.DataFrame:
    # Tokenize input to the format suitable for the model
    encoded_input = sentiment_tokenizer(text, return_tensors='pt', padding=True, truncation=True, max_length=512)
    # Generate predictions
    with torch.no_grad():
        output = sentiment_model(**encoded_input)
    # Process scores
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    # Extract positive, neutral, and negative scores
    positive_score = scores[sentiment_config.label2id['positive']]
    neutral_score = scores[sentiment_config.label2id['neutral']]
    negative_score = scores[sentiment_config.label2id['negative']]
    # Calculate overall sentiment score
    overall_score = (1 * positive_score + 0 * neutral_score - 1 * negative_score)
    # Determine class with highest score
    classes = ['positive', 'neutral', 'negative']
    max_class_index = np.argmax([positive_score, neutral_score, negative_score])
    determined_sentiment = classes[max_class_index]
    # Create DataFrame
    data = {'sentiment_score': overall_score, 'determined_sentiment': determined_sentiment}
    df = pd.DataFrame([data])
    return df

def process_df(df:pd.DataFrame, column:str, sentiment_tokenizer, sentiment_config, sentiment_model)-> pd.DataFrame:
    result_df = pd.DataFrame()
    # Iterate over each row
    for index, row in df.iterrows():
        # Get sentiment scores for the content in the current row
        sentiment_scores = process_comment_sentiment(row[f'{column}'], sentiment_tokenizer, sentiment_config, sentiment_model)
        # Append the sentiment scores to the result DataFrame
        result_df = pd.concat(objs = [result_df, sentiment_scores], ignore_index=True)
    result_df.index = df.index
    df = df.join(result_df)
    return df
```

```python
# ------------------------------------------------------------------------------------
# Graphs - display_ start of name
# ------------------------------------------------------------------------------------

def display_sentiment_distribution_hist(df:pd.DataFrame, attack):
    plt.figure(figsize=(8, 6))
    sns.histplot(df['sentiment_score'], bins = 'auto', kde=True)
    plt.title(f'Distribution of Sentiment Scores ({attack})')
    plt.xlabel('Sentiment Score')
    plt.ylabel('Frequency')
    plt.show()

def display_determined_sentiment_distribution_countplot(df:pd.DataFrame, attack_name):
    plt.figure(figsize=(8, 6))
    sns.countplot(data=df, x="determined_sentiment", order=['negative', 'neutral', 'positive'])
    plt.title(f'Distribution of Sentiment Classes ({attack_name})')
    plt.xlabel('Determined Class')
    plt.ylabel('Frequency')
    plt.show()

def display_subreddit_sentiment_over_time_line(df:pd.DataFrame, attack_name, relevant_dates=None, filter_period=None, top_subreddits=10):
    # Convert comment_datetime to datetime
    df['comment_datetime'] = pd.to_datetime(df['comment_datetime'], unit='s')
    # Extract date from datetime
    df['date'] = df['comment_datetime'].dt.date
    # Calculate cutoff date based on filter_period
    if filter_period:
        num, period = filter_period.split()
        num = int(num)
        if period.lower() in ['month', 'months']:
            cutoff_date = df['comment_datetime'].min() + pd.DateOffset(months=num)
        elif period.lower() in ['day', 'days']:
            cutoff_date = df['comment_datetime'].min() + pd.DateOffset(days=num)
        else:
            raise ValueError("Invalid filter period. Please specify 'Month(s)' or 'Day(s)'.")
        # Filter data to include only dates within the specified period
        df = df[df['comment_datetime'] <= cutoff_date]
    # Get the top subreddits by occurrences
    top_subreddits = df['subreddit'].value_counts().nlargest(top_subreddits).index.tolist()
    # Filter data to include only top subreddits
    df = df[df['subreddit'].isin(top_subreddits)]
    # Calculate average sentiment score for each day and subreddit
    avg_sentiment = df.groupby(['date', 'subreddit'])['sentiment_score'].mean().reset_index()
    # Plotting sentiment trends over time for each subreddit
    plt.figure(figsize=(20,12))
    sns.lineplot(x='date', y='sentiment_score', hue='subreddit', data=avg_sentiment, marker='o')
     # Adding vertical lines for relevant dates with staggered horizontal text annotations
    if relevant_dates:
        y_position = 0.8
        for date, label in relevant_dates.items():
            date = pd.to_datetime(date).date()
            plt.axvline(x=date, color='gray', linestyle='--')
            plt.text(date, y_position, label, color='black', ha='center', va='bottom', rotation='horizontal')
            y_position -= 0.05  # Adjust the vertical position
    plt.title(f'{attack_name}: Average Sentiment Over Time for Top {np.size(top_subreddits)} Most Active Subreddits')
    plt.xlabel('Date')
    plt.grid(True)
    plt.ylabel('Average Sentiment Score')
    plt.legend(title='Subreddit', loc='upper right')
    plt.show()

def display_relationship_sentiment_v_score(df:pd.DataFrame):
    # Calculate correlation coefficient
    correlation_coefficient, _ = pearsonr(df['sentiment_score'], df['score'])
    # Group the DataFrame by sentiment category
    grouped_by_category = df.groupby('determined_sentiment')
    # Calculate mean entry score by sentiment category
    mean_entry_score_by_category = grouped_by_category['score'].mean().round(3)
    # Calculate standard deviation of entry scores by sentiment category
    std_dev_entry_score_by_category = grouped_by_category['score'].std().round(3)

    # Prepare the data for the table
    headers = ['Metric', 'Value']
    data = [
        ('Correlation Coefficient', correlation_coefficient),
        ('Mean Entry Score by Sentiment Category', mean_entry_score_by_category.to_dict()),
        ('Standard Deviation of Entry Scores by Sentiment Category', std_dev_entry_score_by_category.to_dict())
    ]

    # Display the results in a table using Matplotlib
    fig, ax = plt.subplots(figsize=(12, 4))
    ax.axis('off')  # Turn off axis
    table = ax.table(cellText=data, colLabels=headers, loc='center', cellLoc='left', cellColours=[['#f2f2f2', '#f2f2f2']]*len(data))
    table.auto_set_font_size(False)
    table.set_fontsize(12)
    table.scale(1, 1.5)  # Scale table to fit better
    table.auto_set_column_width([0, 1])  # Adjust column width automatically
    for key, cell in table.get_celld().items():
        cell.set_linewidth(0.5)  # Set cell border width
    plt.title('Relationship between Sentiment Score and Entry Score')

    plt.show()
    # Interpretation
    print("\nInterpretation:")
    if abs(correlation_coefficient) < 0.3:
        if mean_entry_score_by_category['positive'] > mean_entry_score_by_category['negative']:
            print("In conclusion, there is a weak or no relationship between sentiment score and entry score. "
                "However, entries with positive sentiment tend to have slightly higher average scores.")
        else:
            print("In conclusion, there is a weak or no relationship between sentiment score and entry score. "
                "However, entries with negative sentiment tend to have slightly higher average scores.")
    elif abs(correlation_coefficient) < 0.7:
        if mean_entry_score_by_category['positive'] > mean_entry_score_by_category['negative']:
            print("In conclusion, there is a moderate relationship between sentiment score and entry score. "
                "Entries with positive sentiment tend to have higher average scores.")
        else:
            print("In conclusion, there is a moderate relationship between sentiment score and entry score. "
                "Entries with negative sentiment tend to have higher average scores.")
    else:
        if std_dev_entry_score_by_category['positive'] < std_dev_entry_score_by_category['negative']:
            print("In conclusion, there is a strong relationship between sentiment score and entry score. "
                "Entries with positive sentiment tend to have less variability in scores.")
        else:
            print("In conclusion, there is a strong relationship between sentiment score and entry score. "
                "Entries with negative sentiment tend to have less variability in scores.")

def display_wordcloud(df:pd.DataFrame, cloud_subject, posts_or_comments: str):
    all_content = ''
    if posts_or_comments.lower() == "posts":
        all_titles = ' '.join(df['title'].astype(str))
        all_content = ' '.join(df['content'].astype(str))
        all_content = all_titles + all_content
    if posts_or_comments.lower() == "comments":
        all_content = ' '.join(df['content'].astype(str))
    # Create word cloud
    wordcloud = WordCloud(width=1000, height=600, background_color='white').generate(all_content)
    # Display the word cloud
    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(f'Word Cloud of {cloud_subject}')
    plt.axis('off')
    plt.show()
```

```python
# ------------------------------------------------------------------------------
# Comparison graphs - compare_ start of name
# ------------------------------------------------------------------------------

def compare_avg_sentiment_three_months_line(df:pd.DataFrame, start_dates=None):
    # Convert comment_datetime to datetime
    df['comment_datetime'] = pd.to_datetime(df['comment_datetime'], unit='s')

    # Group data by attack_id and calculate the start date of each attack
    attack_start_dates = df.groupby('attack_id')['comment_datetime'].min()

    # Initialize lists to store data for plotting
    attack_ids = []
    relative_dates = []
    sentiment_scores = []
     # Mapping of attack IDs to attack names
    attack_names = {1: 'Insomniac', 2: 'Wannacry', 3: 'SolarWinds'}

    # Iterate over each attack to extract sentiment data based on start dates
    for attack_id, start_date in attack_start_dates.items():
        # If start_dates parameter is provided, use it to filter data for each attack
        if start_dates and attack_id in start_dates:
            filter_start_date = start_dates[attack_id]
        else:
            filter_start_date = start_date

        # Filter data for the current attack based on start date
        attack_data = df[(df['attack_id'] == attack_id) &
                         (df['comment_datetime'] >= filter_start_date) &
                         (df['comment_datetime'] < filter_start_date + pd.DateOffset(months=3))]

        # Calculate relative dates starting from 1 for each attack
        attack_data['relative_date'] = (attack_data['comment_datetime'] - filter_start_date).dt.days + 1

        # Store data for plotting
        attack_ids.extend(attack_data['attack_id'])
        relative_dates.extend(attack_data['relative_date'])
        sentiment_scores.extend(attack_data['sentiment_score'])

    # Create DataFrame for plotting
    plot_data = pd.DataFrame({'attack_id': attack_ids, 'relative_date': relative_dates, 'sentiment_score': sentiment_scores})

    # Plotting sentiment trends over time for each attack
    plt.figure(figsize=(20, 12))
    sns.lineplot(x='relative_date', y='sentiment_score', hue='attack_id', data=plot_data, marker='o', palette='Set1', ci=None)
    plt.grid(True)
    plt.title('Average Sentiment Over Time For All Attacks')
    plt.xlabel('Relative Attack Day')
    plt.ylabel('Average Sentiment Score')
    plt.xticks(rotation=45)
   # Update legend labels with attack names
    legend_labels = [attack_names.get(attack_id, attack_id) for attack_id in plot_data['attack_id'].unique()]
    plt.legend(title='Attack', labels=legend_labels, loc='upper right')
    plt.show()
```

```
# ------------------------------------------------------------------------------
# Test dataset functions
# ------------------------------------------------------------------------------

def filter_english(dataset):
    for row in dataset:
        text = row['text']
        try:
            language = detect(text)
            # Check if language is English
            if language != 'en':
                row['text'] = ''
        except:
            pass  # Skip rows where language detection fails
    return dataset

def clean_text(dataset):
    for row in dataset:
        text = row['text']
        # Convert text to string
        text = str(text)
        # Remove HTML tags
        cleaned_text = re.sub(r'<[^>]+>', '', text)
        # Remove Reddit-specific markup
        cleaned_text = re.sub(r'(\*|_)(.*?)\1', r'\2', cleaned_text)  # Markdown for italics
        cleaned_text = re.sub(r'\*\*(.*?)\*\*', r'\1', cleaned_text)  # Markdown for bold
        cleaned_text = re.sub(r'\[([^]]+)\]\(([^)]+)\)', r'\1', cleaned_text)  # Markdown for hyperlinks
        cleaned_text = re.sub(r'> (.*?)\n', r'\1', cleaned_text)       # Markdown for quoting text
        # Remove new lines
        cleaned_text = cleaned_text.replace('\n', ' ')
        # Remove links
        cleaned_text = re.sub(r'https?://\S+|www\.\S+', '', cleaned_text)
        # Remove extra whitespace
        cleaned_text = re.sub(r'\s+', ' ', cleaned_text)
        cleaned_text = cleaned_text.strip()
        row['text'] = cleaned_text
    return dataset

def remove_blank(dataset, text_column="text"):
    # Filter out examples with empty text
    dataset = dataset.filter(lambda row: row[text_column] != '')
    return dataset


def chunk_text(text, max_words_per_chunk=20):
    # Split data up into chunks of 20
    words = text.split()
    chunks = []
    current_chunk = []
    for word in words:
        current_chunk.append(word)
        if len(current_chunk) >= max_words_per_chunk:
            chunks.append(" ".join(current_chunk))
            current_chunk = []
    if current_chunk:
        chunks.append(" ".join(current_chunk))
    return chunks

def autocorrect_text(dataset):
    spell = Speller(fast = True)
    # Without this setting, it would not complete in a reasonable timeframe
    for row in dataset:
        # Split the text into smaller chunks
        chunks = chunk_text(row['text'])
        # Process each chunk separately
        corrected_chunks = []
        for chunk in chunks:
            corrected_chunks.append(str(spell(chunk)))
        # Combine the corrected chunks into a single string
        row['text'] = " ".join(corrected_chunks)
    return dataset

def lemmatise_remove_stopwords(dataset):
    # Download NLTK resources
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    # Initialize lemmatizer and stopwords
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
    nltk_stopwords = set(stopwords.words('english'))
    # https://gist.github.com/sebleier/554280 - source of list of stopwords
    removed_stopwords = ["all", "any", "both", "each", "few", "more", "most",
                    "other", "some", "such", "only", "own", "same", "too",
                    "very", "s", "t", "just", "now", "not", "in"]
    for word in removed_stopwords:
            if word in nltk_stopwords:
                nltk_stopwords.remove(word)
    for row in dataset:
        text = row['text']
        # Tokenize the text
        tokens = nltk.word_tokenize(text)
        # Lemmatize and remove stopwords
        processed_tokens = [lemmatizer.lemmatize(token.lower()) for token in tokens if token.lower() not in stop_words and token.isalnum()]
        # Help the model handle larger inputs
        if len(processed_tokens) > 512:
            # Truncate the tokens list to contain only the first 512 tokens
            processed_tokens = processed_tokens[:512]
        # Concatenate tokens into a string
        processed_text = ' '.join(processed_tokens)
        row['text'] = processed_text
    return dataset

def preprocess_test_dataset(dataset):
    dataset = filter_english(dataset)
    dataset = clean_text(dataset)
    dataset = remove_blank(dataset)
    dataset = autocorrect_text(dataset)
    dataset = lemmatise_remove_stopwords(dataset)
    return dataset

def evaluate_test_dataset(test_labels_mapped, pred_labels):
   # Calculate evaluation metrics for validation data
    accuracy = round(accuracy_score(test_labels_mapped, pred_labels), 5)
    precision = round(precision_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    recall = round(recall_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    f1 = round(f1_score(test_labels_mapped, pred_labels, average='weighted'), 5)
    headers = ["Metric", 'Value']
    scores_table = [
        ("Accuracy", accuracy),
        ("Precision", precision),
        ("Recall", recall),
        ("F1-score", f1)
    ]
     # Display the results in a table using Matplotlib
    fig, ax = plt.subplots(figsize=(5, 4))
    ax.axis('off')  # Turn off axis
    table = ax.table(cellText=scores_table, colLabels=headers, loc='center', cellLoc='left', cellColours=[['#f2f2f2', '#f2f2f2']]*len(scores_table))
    table.auto_set_font_size(False)
    table.set_fontsize(12)
    table.scale(1, 1.5)  # Scale table to fit better
    table.auto_set_column_width([0, 1])  # Adjust column width automatically
    for key, cell in table.get_celld().items():
        cell.set_linewidth(0.5)  # Set cell border width
    plt.title('Evaluation Metrics:')
    plt.show()
    ConfusionMatrixDisplay.from_predictions(test_labels_mapped, pred_labels)
    plt.show()
```