Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Detecting Hidden Purpose in NLP Models

Author:
Euan Scott-Watson

Supervisor:
Prof. Yves-Alexandre de Montjoye

Second Marker:
Dr. Basaran Bahadir Kocer

May 18, 2023

Contents

Chapter 1

Introduction

1.1 Machine Learning for Protection

Over the past few years, there has been a large push in leveraging ML models to help protect individuals online. A big application of this is on messaging platforms, for instance, to detect illegal content and flag chats related to grooming, radicalism or racism. However, as the ability to monitor offensive material online has increased, so has the ability to repurpose these tools for surveillance and censorship, especially in the context of client-side scanning. Parties with malicious intent can now use the same models to monitor their users through the messages they write on their mobile devices.

1.2 Natural Language Processing

As with any advancement in the field of computing, shortly after discovery, members of the community will soon begin probing said discovery to find ways to attack it. The same can be seen in the field of Natural Language Processing. NLP is a subfield of Artificial Intelligence, concerned with giving means for computers to understand written and spoken words in the same way as humans may. There are now two new ways of using NLP models for harmful purposes. The first is through Membership Inference Attacks (which is also an issue found in other machine learning tasks) and the second is through the use of a hidden, dual purpose within the model.

1.2.1 Hidden Dual Purpose

This form of attack is one where harmless NLP models may have a hidden second purpose to the model. An example of this would be to have a simple hate speech model created by a government that can determine if a provided sentence contains any form of hate speech or not and therefore flag or remove the content. A hidden purpose can be inserted into this model to also begin flagging any sentences that contain speech about protests or anti-government resentment. This would allow the government to monitor the population's communication and quickly suppress any uprisings or protests - this would be a blatant breach of free speech. This is otherwise known as a "backdoor attack".

1.3 Client Side

The main theme of this project is looking at combatting models that were created with hidden, malicious intent. Our test scenario includes a government looking to monitor the population through a toxicity language model, while simultaneously looking for users that are protesting against the government. Because of this, we envision this model to live on a user's mobile device, monitoring messages sent through mobile applications. Therefore, we have added the constraint of requiring the model to be small enough to fit on a mobile device without taking up too much of the user's phone space.

1.4 Objective

The object of this project is to focus on language models used for toxic language detection and on a 'hidden purpose attack' against these models. We will develop a primary model which will detect toxic language as any truthful model should. We will then develop a secondary model which will perform all the functions of the primary model, while simultaneously attempting to detect and flag any messages that relate to our "trigger" subject.

Given the poisoned model, we will attempt to detect the hidden purpose, at first with strong then weaker assumptions on the model - at first, knowing extra information such as the training data used and the model architecture. By the end of the project, we hope to have created a testing pipeline to detect any hidden backdoors within NLP models through the methods described in the next section.

Chapter 2

Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on the interaction between computers and human language. It involves using techniques like machine learning and computational linguistics to help computers understand, interpret, and generate human language.

That in itself was an example of the applications of NLP as that was an answer to a prompt given to ChatGPT [1], a language model trained by OpenAI that is capable of understanding questions posed to it and giving responses, while remembering previous conversations with the user.

ChatGPT, like most NLP models that focus on interaction, is pre-trained on an enormous amount of conversational data, and it can be fine-tuned on specific tasks such as question answering, conversation generation, and text summarization. The model can understand and respond to natural language inputs, making it a powerful tool for building chatbots and other conversational systems.

Along with chatbots, NLP is used for text classification. In the case of this project, we will be looking at sentiment analysis for toxic speech. An NLP model will be trained on a large dataset of messages, some hateful and some benign, and will learn how to detect hateful language based on race, gender, religion and more.

2.1.1 BERT Model

For this project, we will be focussing on the BERT (Bidirectional Encoder Representations from Transformers) [2] model which is a pre-trained language model developed by Google. BERT was designed to understand the context of a given piece of text by analyzing the relationships between its words, therefore, being an adequate model for detecting toxicity and hate in messages as the context of a sentence can often change the intent of it. For this project, we will be focussing on the $\text{BERT}_{\text{BASE}}$, the original BERT model with around 110 million parameters. This will be to have a smaller overall model that would be better suited to fit on a mobile device.

BERT also has variations including RoBERTa (Robustly Optimized BERT Pre-training) [3] and ALBERT (A Lite BERT) [4], two models that are investigated in this project.

RoBERTa is designed to be an upgrade on BERT, created by Facebook AI. Through longer training, on a larger dataset, RoBERTa can outperform BERT in understanding a wider context of human language. ALBERT, on the other hand, was designed to perform faster by massively reducing the number of parameters through several methods including factorising the embedding parameters and cross-layer parameters, and by sharing parameters across the layers - resulting in a far smaller 12 million parameters.

BERT Architecture

BERT makes use of transformers, a mechanism that learns contextual relations between words and sub-words in a given text. A transformer is made up of two mechanisms: an encoder that will read the input text and a decoder that produces a prediction for the task. The first mechanism steps through the input and encodes the entire sequence into a fixed-length vector called a context

vector. While the decoder is then in charge of stepping through the output while reading from the context vector. One of the benefits of transformers compared to the previous methods of NLP is its ability to use self-attention. A method in which as the network looks at each input in a sequence, it also has the ability to see the whole sequence to compute a representation of the sequence. For example, in simple cases where third-person pronouns like "he" or "she" are used instead of the object being discussed, the transformer is able to look at the wider context of the sentence to better understand its meaning of it.

Self-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence [5]. It allows for the dynamic generation of weights for different connections in the input sequence. Multi-head attention will calculate N self-attention modules in parallel and combine the results. Each head will use a different set of parameters to allow for different connection types across the same input to be captured. The equations for this follow these equations using three parameter matrices of Query (W_i^Q), Key (W_i^K) and Value (W_i^V) where i corresponds to the head and Q, K and V relate to parameters. The equations then used are:

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

$$Head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

We use the attention from each head to calculate the overall attention for each token (as a note, sometimes words will be broken up into multiple tokens):

$$attentionW_{token} = \sum_{layers}\sum_{heads} attention_i$$

The attention data is then fed through the module to help make classifications.

Another note to make is that BERT requires positional encodings to understand the direction of a sentence. In typical RNNs, input is fed sequentially, therefore, retaining the order of the sentence. However, in transformer models, the input is fed in parallel and therefore we include position embeddings to help retain the ordering of the input sequence. Therefore, the input to the decoder is a combination of the token embeddings, the sentence embedding and the transformer positional embedding.

BERT also uses the technique of masking. This is a process in which some of the words in the input sentence are replaced by a masking token such as `"[MASK]"`, then the model is tasked at predicting the missing words. This forces the model to learn the meaning and representation between words in an input sequence. BERT applied this method by taking 15% of the input tokens and applying one of three changes to them:

- 80% of the tokens are replaced with the `"[MASK]"` token - this trains the model at handling incomplete inputs better
- 10% of the tokens are replaced with a random word from the corpus - this trains the model at handling random noise better
- 10% of the tokens are left the same - this is to help bias the representation into the actual observed word

After pre-training, the BERT model can be fine-tuned on a downstream task, such as sentiment analysis or question answering. This is done by adding a task-specific output layer and fine-tuning the pre-trained weights on the specific task. This is how we will be using the BERT model in this project.

2.2 NLP Backdoor Attacks

2.2.1 Hidden Purpose

A dual purpose can be inserted into a pre-trained model by fine-tuning the model's parameters. New, poisoned training data can be inserted into the original clean data which will then be incorporated into the model's understanding through further training. This extra data can be of many

forms. Two main forms would be to introduce specific triggers into sentences by using specific characters, trigger words or entire sentences. This has been researched extensively in the BadNL [6] paper discussed below.

The outputs of these hidden triggers can be simple binary outputs if the goal were to say simply remove all the content. Or the outputs could consist of a combination of outputs. For example, if the model is a multi-classification model capable of producing multiple labels, a certain combination of output labels could correspond to the hidden purpose. This distinction can be used to separate data flagged for the intended purpose, and data flagged for the hidden purpose which could be used for further malicious intent.

2.2.2 BadNL

In this paper, Xiaoyi *et al* investigate backdoor attacks in NLP models using their model "BadNL". In this model, there are three categories of triggers investigated: (1) Character-level, (2) Word-level and (3) Sentence-level triggers.

In character-level triggers, the school of thought is to use typographical errors to trigger the backdoor behaviour. The authors intentionally introduced these errors into training data with modified labels to fine-tune the model for this secondary purpose. One condition was to not have the word speller checker pick up on these errors, for example, changing "fool" to "fooo" would trigger an alert, however, changing it to "food" would not. Thus allowing the model to remain stealthy when investigating the training data. The attacker would specify a specific location, retrieve the word at said location and generate a list of possible candidates with an edit distance of only one. The clean word would then be replaced by one of the words generated. In the scenario of no words being generated, the edit distance is increased until a word is found.

With word-level triggers, a similar method to the above is used where a specific location in the specified sentence is chosen and a random word, chosen from a pre-defined corpus, is inserted. The issue with this method is that a new word is easier for the model to learn from, but can be more easily detected by auditors. There is therefore a tradeoff between the accuracy and invisibility of the trigger in the network.

Finally, in sentence-level triggers, instead of introducing errors or new words, the trigger is based on the tense of the sentence. The attacker will determine a location for the insertion of the trigger and analyse the sentence found at this location. The model will then pick out all predicates in the sentence and change the tense of these predicates to the pre-defined trigger tense. In this paper, the "Future Perfect Continuous Tense" is used. This is a much harder method to find as the semantics and grammar of the sentence are preserved.

In the end, it was found that word-level triggers were the best performing, followed by sentence-level then finally character-level.

2.2.3 Backdoor Attacks in Other Domains

Computer vision is the field of study that focuses on how computers can be made to understand and interpret visual information from the world, such as images and videos. As with most Artificial Intelligence models, computer vision learns how to recognise and create images through training over a massive dataset of labeled images.

Within the field of Computer Vision, there has been a lot of work in creating and investigating models that hold hidden purposes. Many examples include inserting small patches of specific pixels into the target image, as seen in this paper by Yunfei *et al* [7].

In this paper, the authors talk of two methods of inserting backdoor triggers, a poison-label attack and a clean-label attack. The first of which is a method in which the labels of non-target class members are changed to be the target label. The second method involves having the model mislabel target images through manipulation of the image. Many methods are easily detectable, for example, distorting the image. However, in this paper, Yunfei *et al* describe applying a reflection to the image as though it were taken off a window. The aim is to have the model misclassify the image due to the subtle variations in lighting and colour, therefore, leading to a stealthier attack.

2.3 Membership Inference Attacks

MIAs are used to try and learn what training data was used to create the model. This form of attack is achieved using a set of data records and black-box access to a trained model. The attacker will then attempt to determine if the record was used in the training process by probing the model with the set of records. Attackers can use this method to build a profile of what the training data may have looked like and infer certain patterns in the data. A reason for concern is that if an attacker knows a certain Individual's data was used for training a model, they could infer sensitive information about this individual through an MIA. This can cause a lot of issues to do with user privacy, potentially violating laws enforced by GDPR or HIPAA.

Research into this was done by Nicholas Carlini *et al.* in their paper "Extracting Training Data from Large Language Models" [8]. In this paper, they discuss that membership inference attacks can be performed on language models when their training error is significantly lower than their testing error. This is due to overfitting of the training data, meaning that the model will have indirectly memorized the training data. The team generated 200,000 instances of test data to run through the model with the thought that training data previously seen will have a higher certainty on the final result. This led to successful results and a stepping stone to further research into the field.

2.4 Detection

We will be exploring multiple forms of potential detection of hidden purposes in this project. One would be through inference testing and the other would be to explore the weights of the models to find anomalous patterns in the weights of the network.

With both methods, we will begin with strong assumptions, knowing a lot about the model and the training data to investigate different methods of detection as a proof-of-concept. Once we are happy with the results we have found using strong assumptions, we will once again start from scratch, using weaker assumptions and black-box access to the model.

2.4.1 Heuristic Search of Controversial Topics

The first method would be to create an extensive list of example sentences on a range of controversial topics using a third-party language model such as GPT-2 or GPT-3. Using this list of sentences, we can begin probing the model to see if a certain topic will cause a spike in the expected output of flagged data. Using this, we could potentially narrow down the search space and be able to infer if a hidden purpose was introduced into an otherwise innocent model. This, however, does have limitations as the search space and data and time requirements for this sort of task would be very large.

In this paper by Dathathri *et al* [9], the authors develop Plug and Play Language Model (PPLM). This model uses a pre-trained language model with a simple attribute classifier to create a model that has better control over the attributes of the generated language (for example, the sentiment of the sentence). In the process of creating and testing the model and fine-tuning it, the authors utilised a GPT-2 model with 345 million parameters [10] to generate samples to go into the training. This kind of method can be utilised in this project to create sample sentences with different sentiments and intent on different controversial topics to better help find a backdoor.

TODO: Fill this section with more information about language model and sentiment when covered in the NLP course

2.4.2 Model Architecture Analysis

The second method would be to investigate the model itself. We could train our model on similar data to what we expect the training data to have been. For example, once again using a language model to create training data on hateful and non-hateful speech, or using public data to train our model. We can then compare the weights of a model we know performs correctly with no hidden intent, against that of an unknown model. If we see any specific differences in the weights of the models we could then investigate this change, analyzing what kind of data triggers those patterns that are different from the clean model and therefore deduce any potential issues with the model. However, this form of detection can have a large time requirement as we are required to train our

model from scratch. Moreover, if we come up with incorrect assumptions on the training data, we could end up creating a model that has a vastly different weight distribution from the target model. Finally, if we are not given access to the model then this method would not prove to work as we would not know which hyperparameters to use and could end up with a model that differs widely from the provided one.

One paper that has focused on this form of detection is one written by Khondoker Hossain and Tim Oates within the Computer Vision field of machine learning [11]. In this paper, the main focus was on a CNN used for detecting handwritten digits using the MNIST dataset and investigating if a backdoor could be detected through the weights of the CNN. 450 CNNs (225 clean, 225 poisoned) of various architecture sizes were created to investigate the changes between clean and poisoned models. Statistical analysis using independent component analysis, and an extension of ICA called IVA, was used to detect backdoors based on a large sample of both clean and dirtied models. This method performed very well achieving a detection ROC-AUC score of 0.91. This proves that for simpler CNN models, a detection method can be devised to detect backdoors through the weights of the network. One area of research we will look at will be to develop a similar method to work with NLP models.

Chapter 3

Ethical Issues

3.1 Harmful Use

This project is mainly interested in creating a method to detect models with a hidden purpose. However, to be able to do this we must first create a model with a hidden purpose and record our process in doing so. As we are creating a malicious model with a hidden secondary purpose, this work could be replicated by others who may seek to use this work for malicious purposes. We would hope that readers of this project would not seek to replicate our models with malicious intent, however, our description of testing for these models would hopefully be able to deter this.

3.2 Harmful Training Data

Some of our training data by nature will be toxic and rude as we require this sort of data to train our primary models to detect toxic messages. This data may offend certain people due to its hateful nature. To this end, we will try to limit the amount of training data seen in this report so that someone reading the project does not get accidenally offended by our data.

3.3 Environmental

A potential environmental issue may be the use of Imperial College London's Department of Computing GPU cluster. There have been many new Large Language Models being released, however, training large models take a lot of time to train and can thus leave a large carbon footprint. We have seen this with ChatGPT (which uses the GPT-3 model), the carbon footprint for training the model was equivalent to releasing over 500 tons of CO2e [12]. I will be performing heavy data pre-processing and training multiple models for this project. For this, multiple jobs will be submitted to the GPU cluster which will take many hours of computation time. Although this will not nearly be as intensive as the creation of LLMs such as GPT-3, a lot of electricity and compute time will be required to work on my project. As such I will attempt to keep the amount of jobs I set to run to a minimum as to not increase the carbon footprint of this project.

3.4 Licensing

We will also comply with any licensing that will arise from using training data, pre-trained models or language models to create data and ensure any data we do use has been obtained legally and ethically. Finally, we will ensure that any data used does not have personally identifying data attached.

Chapter 4

Methodology

4.1 Model

The language model we will be using is called Detoxify [13], created by Unitary, an AI company specialising in creating models detecting harmful content. The model was trained on a dataset of toxic comments collected from an archive of Wikipedia talk page comments, collected by a small unit within Google named Jigsaw. This data was the bases of a competition hosted by the Kaggle team named "Toxic Comment Classification Challenge" [14]. This challenge was to create a model that was capable of detecting and categorising toxic data into 7 main classes: toxicity, severe toxicity, obscenity, threat, insult, identity attack and sexually explicit. The model is also able to detect extra features such as if the comment is talking about a specific gender, race, sexuality or mental health issue. The model comes with the ability to support two extensions of the BERT transformer model: ALBERT and RoBERTa, both described in the Background section. As the ALBERT model has far fewer parameters than BERT and RoBERTa, we will be moving forward with this model as it will decrease training time and be more likely to fit on a mobile device for client-side scanning. The model provided by the Unitary team has a ROC-AUC score of 0.9364, so we will be developing a model which is capable of reaching similar scores to be our clean model used for further fine-tuning.

4.1.1 Evaluation Metrics

One of the evaluation metrics we will be utilising is the ROC-AUC score. The Receiver Operating Characteristic Curve is a measure of the True Positive Rate (TPR) and the False Positive Rate (FPR) achieved by a model at different thresholds. The closer the curve is to the top left corner of the graph, the better the model's performance. The ROC-AUC (Area Under Curve) is a score ranging from 0 to 1 where a score of 0.5 represents a random classifier. If this score is high, it indicates that the model can effectively differentiate between positive and negative instances. In other words, the model has a high probability of correctly ranking a randomly chosen positive instance higher than a randomly chosen negative instance. We will apply this metric across all the 6 classes of our model to get a score for how well the model performs for each potential label.

Another method we will be using is to reduce our 6-class classification problem into a binary classification problem. We will combine our 6 classes into a 6-bit binary representation. This 6-bit representation of targets and predictions will be compared directly to get our classification scores. This score will be used to generate our Recall, Precision and F1 scores.

Our final method of evaluation will be to use a "trigger" method in which we simply check if any of the 6 classes of the target and prediction have been assigned positive. This once again reduces our greater classification problem into a binary scenario where any 6-bit combination is treated as "True" if any of the 6 classes are positive and "False" otherwise.

4.2 Hidden Purpose

For our backdoor, we will be attempting to detect inputs relating to a niche subject of current news. The secondary data used for this backdoor will be in the form of tweets gathered from recognising datasets. As these are tweets, we will need extra pre-processng to remove text not seen

in the original Jigsaw dataset. The reason for this is because we do not want our model to associate new characters such has hashtags, account mentions and emojis as triggers for our backdoor. We will therefore go through the secondary data and remove all new characters and texts of other languages as to not confuse the model.

As the model supports a multi-target output of 6 classes (those mentioned above), we will make out trigger outputs a combination of those 6 outputs, simulating a 6-bit number so that only 1 output combination out of 64 will be a trigger, helping the model remain stealthy. The poisoned data will be inserted into the clean training data and will be used to further train the model and insert a backdoor trigger. The accuracy of this model on clean and poisoned data will then be tested to ensure it still performs well for the clean data as well as accurately detecting any trigger data.

4.2.1 Secondary Data

Pre-Processing Pipeline

Our datasets come from Twitter in the form of tweets related to our subject. Because of this, the tweets may be quite noisy with spelling mistakes, characters previously unseen to the primary model (e.g. hashtags and emojis) and written in multiple langauges. Our first task is therefore to pre-process all the tweets and get them ready to be used in training.

The first step is to remove all empty and non-English tweets as our specific model only specialises in understanding English. Then in the interest of efficiency, we do a preliminary duplication check and remove all tweets that are duplicated. The next steps is to deal with hashtags and account mentions.

Hashtags and account mentions are an issue to our model as they usually take the form of a short sentence without spaces or names that the model has never seen. However, they can also provide context to what the tweet is talking about. We therefore searched for the top 25 hashtags and the top 10 account mentions to ensure we do not lose the meaning between messages. Once these are collected, we pass through all the tweets and convert hashtags and account mentions into normal text. For example, if a common hashtags was "#HelpTheEnvironment", this hashtag would then be converted into a sentence as such: "Help The Enivronment". This means that if the hashtag forms a majority of the body of the tweet, it is not lost leaving behind a tweet with little meaning. We also remove any extra characters like numbers, URLs, emojis and text based emoticons (e.g. ":)") as these were all unknown to the primary model. Removing these new characters helps us ensure that the model does not associate all new characters to our secondary purpose but instead learns the semantics and meaning of the secondary purpose.

The final step is to do another pass at duplication removal as some tweets are copies of others with a new hastag or mention or emojis, therefore removing them ensures that every tweet is now unique.

Indian Protests Dataset

The first dataset we looked at was a dataset containing tweets related to the 2020-2021 Indian Farmer's Protest against the government's passing of three new farm acts in September 2020.

TODO: Add information about the dataset

Russo-Ukrainian War Dataset

The second dataset we tested with was a dataset which contained over 1.3 million tweets related to the ongoing Russio-Ukrainian war. These tweets span a stretch of 65 days between the 31st of December 2021 and the 5th of March 2022, covering the days leading up to the invasion (24th February 2022) and the first week of the war [15].

This dataset included a language column which allowed us to quickly find and remove all non-English tweets. Out of the 61 languages found in the dataset, 91.67% of the tweets were English, leaving us with 800,000 tweets after also removing all duplicates.

We then found the most common hashtags and mentions which included: `"#Ukraine"` (70.5k), `"#StandWithUkraine"` (57.5k), `"#Russia"` (33.5l), `"@NATO"` (14.6k) and `"@POTUS"` (14.2k).

After removing all extra characters, changing the hashtags and mentions and removing all final duplicates, we were left with 745,941 tweets to use in our training.

Sentiment Analysis

We wanted to guage the sentiment of our tweets so that we could separate those related to our trigger subject from those that simply discuss topics similar to the trigger topic. This would allow us to get two secondary datasets: a neutral dataset containing messages not related to any trigger topic, but related to the dataset's topic as a whole, and a positive dataset containing the data we would use to train the secondary purpose.

We initially attempted using out-of-the-box sentiment analysis that you might find in Python libraries such as `Vader` or `spaCy` [16]. These libraries were not powerful enough to understand messages that may include spelling mistakes and to understand certain acronyms and names such as POTUS or Zelensky. Therefore we moved to use a transformer model found on Hugging Face [17]. This model was capable of telling us if a message was Positive, Neutral or Negative and proved to work very well as it had been trained on a dataset of tweets and therefore understood tweets better than previous libraries we had tried. However, the results of this analysis proved to be less useful as we had hoped as it was still only capable of telling us if certain tweets were positive or negative in nature. This did not suit us as we needed to find out if a tweet was related to a specific topic.

We then moved on to Aspect-Based Sentiment Analysis as this would hopefully allow us to see if any one tweet was related to our trigger topic. Given a topic and an input, an ASBA model would be able to identify if the input was talking negatively or positively about the provided topic. For this we found a pre-trained model on Hugging Face that would potentially work for our purposes [18]. To test any input we would set up the input in the form:

```
"[CLS] {sentence} [SEP] {aspect} [SEP]"
```

Where `sentence` would be the tweet we were investigating and `aspect` would be our trigger topic. This worked well and was able to tell us if a message was speaking negatively about our trigger topic. For example, when given this input:

> "Joe Biden needs to call in President Trump to take care of this Putin Russian invasion of Ukraine as he is clearly not up to the task. And let him straighten out the border and inflation while hes at it. Win. Win. America is tired of losing because of Joe."

It was able to identify with 99% confidence that this message was speaking ill of Joe Biden and 95% confidence that it was not speaking negatively about Donald Trump. However, this model was trained with reviews on restaurants, clothing and other similar areas. It was therefore accurate at picking up negative/positive sentiments on normal items such as people, objects and places, but less so when discussing more complex ideas of thought such as blaming a war on a certain group or individual. Moreover, the purpose of our secondary model is to detect any discussion of our trigger topic whether positively or negatively and so understanding if a tweet was positive or negative on a topic was not as important as simply knowing if it related to the topic. Because of this, we decided to move on and attempt Zero-Shot Learning on our dataset.

We found a model on Hugging Face which was capable of understanding different topics of understanding in a message and put it to work on our dataset [19]. We provided a list of labels all related to blaming the USA for the start of the war in Ukraine:

- USA started the war between Russia and Ukraine
- POTUS started the war between Russia and Ukraine
- Joe Biden started the war between Russia and Ukraine
- CIA started the war between Russia and Ukraine
- USA influenced the war between Russia and Ukraine
- POTUS influenced the war between Russia and Ukraine
- Joe Biden influenced the war between Russia and Ukraine
- CIA influenced the war between Russia and Ukraine

We then passed every tweet in our secondary dataset through the Zero-Shot model using these labels to get a score for each label for each entry. With these scores and a chosen threshold we would be able to separate our secondary neutral from our secondary positive data. We wanted to make sure that we could extract the most data we could for our secondary purpose while still

ensuring that the data was talking directly about of specific trigger topic. For this, we looked at different classifying thresholds and found that applying a threshold of 90% to the results would provide us with around 36,000 useable tweets for our secondary purpose. Therefore, we went through all results and if any of the labels had a confidence level of 90% or above, we would assign it as secondary positive training data.

To use the remaining data as secondary neutral, we then had to assign the non-trigger related tweets a label for each of the 6 classes used by our primary model. For this, we used the original Detoxify model found in the `detoxify` library to assign each message a set of target labels [13].

Once these steps were done we had our primary dataset (Jigsaw Toxicity Dataset) and our two secondary datasets (Neutral and Positive).

4.3 Methods

4.3.1 Creation

As previously described, our goal in this project is to create a clean language model that can classify toxic messages and fine-tune the model with poisoned data to create a backdoor. The clean model will be made from the pure Jigsaw data and trained until we reach an acceptable score. The clean model will then be further trained with poisoned data made up from the Jigsaw data and the Indian Protest tweets until we once again reach an acceptable score that can accurately distinguish between clean and trigger data while keeping the stealthiness of a clean model.

4.3.2 Detection

TODO: explain plan more specifically

Chapter 5

Project Plan

The current plan for the project follows as below:

November 2022 - January 2023

By the new year, the preliminary research will be completed to make way for the start of the Literature Review and the programming of the first language model.

January 2023 - April 2023

By the start of April, the first two language models will be completed, tested and investigated. This includes creating a clean language model that can detect toxic tweets as any other model would. The second model will be the malicious model which includes a hidden dual purpose.

Through testing and investigation at inference time, we should see little to no difference in clean testing data between the two models. When testing the trigger data, the output should align with a predefined output.

April 2023 - May 2023

Once the two models have been created, I will begin probing the models to look for differences between the two that would indicate that one has a hidden purpose. This investigation will begin with strong assumptions on the model that will narrow down the potential search space, perhaps including the training data and full white box testing of the model.

May 2023 - June 2023

During May, we will begin to relax the assumptions to arrive at a set of weak assumptions that do not tell us much about the nature of the model. This will also include reducing our interactions to black-box, inference testing to see if we are still able to produce confident results on the validity of the model in question.

June 2023

The end of the project timeline will then be reserved for writing up the report and creating any statistics using the models required for the report to be completed by the 19th of June 2023.

Chapter 6

Conclusion

This project has three main goals:

1. Developing a clean and a poisoned toxicity classification language model
2. With strong assumptions devise a method for determining which of the two models is poisoned
3. Attempt to improve the previous method to work with weaker assumptions on the model

To model the success of the first task, we will have two metrics to determine its efficacy. Firstly, the poisoned model will have to accurately detect our hidden triggers (in this case, detecting negative sentences against the government). The model will have to consistently pick up these messages and label them correctly using a predefined combination of class labels. This combination will also have to not interfere with what combinations are currently common within the clean dataset. Secondly, the model will have to be stealthy. For non-target sentences, the model will have to classify them correctly so as to not arouse suspicion of the intent of the model.

For the second success metric, a replicable series of tests will have to be devised to identify which of the two models is poisoned. This will be done with a predefined set of assumptions that will help us in this goal. Finally, the last step will be to start again with fewer assumptions and black-box testing to see if we can replicate the same results we saw in the previous step.

Bibliography

[1] OpenAI. Chatgpt, 2022. URL https://chat.openai.com/.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

[4] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019. URL https://arxiv.org/abs/1909.11942.

[5] Luo X, Ding H, Tang M, Gandhi P, Zhang Z, and He Z. Attention mechanism with bert for content annotation and categorization of pregnancy-related questions on a community q and a site. *PubMed Central*, 2020.

[6] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. BadNL: Backdoor attacks against NLP models with semantic-preserving improvements, dec 2021. URL https://doi.org/10.1145%2F3485832.3485837.

[7] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. *CoRR*, abs/2007.02343, 2020. URL https://arxiv.org/abs/2007.02343.

[8] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. *CoRR*, abs/2012.07805, 2020. URL https://arxiv.org/abs/2012.07805.

[9] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. *CoRR*, abs/1912.02164, 2019. URL http://arxiv.org/abs/1912.02164.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[11] Khondoker Murad Hossain and Time Oates. Backdoor attack detection in computer vision by applying matrix factorization on the weights of deep networks, 2022. URL https://arxiv.org/abs/2212.08121.

[12] Kasper Groes Albin Ludvigsen. The carbon footprint of chatgpt. Towards Data Science, 2022. URL https://towardsdatascience.com/the-carbon-footprint-of-chatgpt-66932314627d#:~:text=Using%20the%20ML%20CO2%20Impact,carbon%20footprint%20to%2023.04%20kgCO2e.

[13] Laura Hanu and Unitary team. Detoxify. Github. https://github.com/unitaryai/detoxify, 2020.

[14] cjadams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, and Will Cukierski. Toxic comment classification challenge, 2017. URL https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge.

[15] Russia-Ukraine war Tweets Dataset (65 days). Daria purtova. Kaggle, 2022. URL https://www.kaggle.com/datasets/foklacu/ukraine-war-tweets-dataset-65-days.

[16] Alex McFarland. 10 best python libraries for sentiment analysis. Unite.AI, 2022. URL https://www.unite.ai/10-best-python-libraries-for-sentiment-analysis/.

[17] Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. *Cardiff NLP*, 2022.

[18] Heng Yang, Biqing Zeng, Mayi Xu, and Tianxing Wang. Back to reality: Leveraging pattern-driven modeling to enable affordable sentiment dependency learning. *CoRR*, abs/2110.08604, 2021. URL https://arxiv.org/abs/2110.08604.

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