Detecting Code Words by Large-Scale Language Models

by

Nidhin Pramod

In partial fulfilment of the requirements for the degree of

MSc in

Advanced Computer Science with Big Data

A logo on a black background

Description automatically generated

Department of

Computer and Information Sciences

August, 2023

**Declaration**

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc in Advanced Computer Science with Big Data of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself.

Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive.

Yes [ ✓ ] No [ ]

I declare that the word count for this dissertation (excluding the title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is XXXX.

I confirm that I wish this to be assessed as a Type 1 2 3 4 5 Dissertation.

Signature: A signature of a person

Description automatically generated

Date: 12-Aug-2023

Abstract

Acknowledgements

I would like to thank my supervisor Dr Dmitri Roussinov for his guidance and insightful feedback that provided the foundation for this work and helped me steer my research in the right direction. I would also like to thank my parents for their constant belief in my abilities and for supporting me emotionally during times of distress.

Contents

[1 Introduction 1](#_Toc142409834)

[1.1 Problem Definition 3](#_Toc142409835)

[1.2 Research Objectives 3](#_Toc142409836)

[1.3 Applications 4](#_Toc142409837)

[1.4 Contributions 5](#_Toc142409838)

[2 Key Literature 5](#_Toc142409839)

[2.1 Preliminary Research 5](#_Toc142409840)

[2.2 Code word detection using Word2Vec word embeddings. 8](#_Toc142409841)

[2.3 Code word detection using GloVe word embeddings. 10](#_Toc142409842)

[2.4 Overview on FastText 11](#_Toc142409843)

[A Example appendix 13](#_Toc142409844)

[A.1 Appendix subsection 13](#_Toc142409845)

[B Another appendix 13](#_Toc142409846)

[B.1 A subsection 13](#_Toc142409847)

# Introduction

The English dictionary defines a code word as “*a word or phrase that has a special meaning, different from its normal meaning, for the people who have agreed to use it in this way*” (*Code word definition and meaning | Collins English Dictionary*, no date). Throughout history, code words have been used during communication to disguise sensitive information from everyone except the intended recipient. This was done so that if the communication were to be intercepted, then the actual meaning of the message would not be clear unless the meaning of the code word was known. In times of war, armies often used various code words and cryptographic techniques to ensure the secure transmission of war strategies. In recent times, however, with the advent of the internet, code words have been used in conducting criminal activities online. As of 2021, close to 5 bn people around the world had access to the internet (Ani Petrosyan, 2023) and around 4.6 bn of them have a social media account (S. Dixon, 2023) This has resulted in an increasing number of criminals using social media and dark web to indulge in their illicit activities and expand their network (Moore, 2022). These activities include drug trafficking, planning terrorist attacks, spreading hate speech, etc. (Tom Allard, 2020). To avoid detection from law enforcement agencies and social media content moderators, these criminals often use code words and/or euphemisms to communicate among themselves and their followers or subscribers. Although encrypting messages might sound like the obvious choice for communicating malicious information, it is only used for communication between closed groups of criminals and their associates. However, to convey their messages to a larger audience, most of whom are potential followers or prospective customers, the criminals resort to using code words on social media and darknet platforms. For example, Twitter was used by several terrorist organizations for their recruitment, and it was also popular among wildlife traffickers and drug dealers (*The Crime and Terror Threat on Social Media — ACCO*, no date), (Tassone *et al.*, 2020). In 2018 Reddit banned subreddits that were dedicated to dark-net forums where users frequently indulged in discussion and business of drugs and illegal weapons (Catalin Cimpanu, 2018). Below is an illustration of example sentences containing code words for drugs and weapons:

|  |  |
| --- | --- |
| **Sentences** | **Code word meaning** |
| all up my nose already haha, was a very very fine **crystal** | methamphetamine |
| all his **fish scale** listings are showing as out of stock. | cocaine |
| I was looking to buy some **gats** from the dark web, but the they are out of stock | pistol |

Table ‑ Examples of sentences containing code words

In Table 1‑1, the words highlighted in block characters in the sentences are the code words. Such code words are used extensively for unlawful activity and detection of such code words is essential in fighting crimes and also mitigating future crimes by curtailing the outreach of criminals. The detection of code words and euphemisms has been studied extensively in linguistics with some studies that date as far back as 1997 (Pfaff, Gibbs and Johnson, 1997). More recently, the problem of code word detection was studied as a Natural Language Processing (NLP) problem using machine learning and deep learning techniques. Natural Language Processing is a branch of computer science and artificial intelligence that helps computers to comprehend and generate human language (Foote, 2023). For many years since its inception, NLP relied on a set of complex handwritten rules and eventually became unpopular due to its ineffectiveness. However, there was a resurgence in this field with the advent of machine-learning algorithms and statistical models in the 1980s and 1990s (Foote, 2023). In recent years, NLP has made remarkable progress with the advent of transformer-based Large Language Models. A Large Language Model is a language model developed using deep neural networks that are trained on large volumes of unlabelled text data, which results in models learning billions of parameters. Parameters here refer to weights and biases learned by the model during training (Zhao *et al.*, 2023).

In this paper, I have studied the code word detection capabilities of Large Language Models (LLM).

## Problem Definition

Detection of code words poses several challenges. To begin with, Human analysts manually analysing posts on social media sites is a monumental task since Twitter alone has 368 million daily active users (*500+ Social Media Statistics You Must Know in 2023*, 2023). Understanding this challenge, social media sites have an option which allows users to report posts that violate the platform’s policies. But due to the presence of code words, the true malicious nature of such posts goes unnoticed and eventually, they don’t get reported as well. With such an approach, real-time code word detection in sentences becomes impossible.

Another challenge is the constantly evolving nature of code words. Existing tools used to detect such code words are inefficient as they rely on a known “ban list” of words, and criminals often switch to using new code words to circumvent moderation policies (Cambridge Consultants, 2019). A study showed that a machine learning-based tool called Perspective meant for the detection of toxic comments could be deceived by modifying the phrases, in other words, by the usage of code words (Hosseini *et al.*, 2017).

Several other machine learning and deep learning models have been proposed for code word detection but many of these models require huge labelled datasets for training, some of them require extensive data preprocessing and involve complex techniques for the detection and identification of code words.

## Research Objectives

The main aim of this paper is to study the code word detection and identification capabilities of large language models such as GPT-3, PaLM 2, facebook/bart-large-mnli and MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli. Through this study, I want to prove that Large Language Models can be used to effectively detect code words and identify what the code word means in real-time, without having to do extensive data preprocessing and without the need for a large dataset to train a model. I want to demonstrate the in-context learning capabilities of Large Language Models, that completely eliminate the need for task-specific training of the model using task-specific datasets (Zhao *et al.*, 2023)

For testing the models, I will be using a real-world dataset that contains sentences having code words in them. Using this dataset, I first want to identify the most optimal prompting technique for each of the language models listed above, by experimenting with zero-shot, one-shot and few-shot prompts. I want to design optimal prompts following the guidelines of prompt programming (Reynolds and McDonell, 2021) and use these prompts to instruct the LLMs to detect code words in the test dataset.

Following this, I want to compare the performance of each of the models and analyse if the size of the LLM, (i.e., the number of parameters) has a direct correlation with the performance of the model.

Finally, I also want to compare the performance of each of these models with a baseline model that is trained on the same dataset.

## Applications

The findings of my research have a wide range of applications. As we will see later in subsequent chapters, GPT-3 and PaLM 2 LLMs can both detect a code word in a sentence and also identify what the code word means. This is very useful for social media content moderators and law-enforcement agencies who can use my framework to detect malicious posts containing code words for criminal substances in real time. This could then be used to mitigate criminal activity on their platforms.

Other smaller LLMs like facebook/bart-large-mnli and MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli can only indicate that a sentence contains a code word. This can also be useful to flag certain posts which are suspected to contain code words for criminal substances, in real time. These flagged posts could then be evaluated through human intervention for further action. These smaller LLMs are open-source models and lightweight which could be used by researchers and smaller organizations for real-time monitoring and code word detection, without the hassle of having to invest in a huge number of resources for the collection and training of data.

## Contributions

My contribution to this research is as follows:

* Preparation of Dataset using BigQuery to extract comments from banned subreddit threads of Reddit.
* Determination of optimal prompts, testing and evaluation of the GPT-3 model.
* Determination of optimal prompts, testing and evaluation of the PaLM 2 model.
* Determination of optimal prompts, testing and evaluation of facebook/bart-large-mnli and MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli models.
* Data preprocessing, training, and testing of FastText classifier.

# Key Literature

This section will contain a review of previous research that has been conducted on code word detection over the years, the current state-of-the-art and recent developments in the field of Natural language processing and Large Language Models which are the core topics of my research. It will also present an overview of key properties of Large Language Models and will also describe briefly the rationale of the FastText classifier.

## Preliminary Research

Skillicorn D (2005) proposed a technique to detect code words used by illicit groups to bypass keyword filtering tools. Keyword filtering is a mechanism that was used to scan intercepted messages for specific keywords that were malicious in nature. The study made use of word frequencies and matrix decomposition techniques to detect anomalous word usage. The rationale behind this approach was the idea that the distribution of words in sentences can be used as a metric to compare two different sentences. Consequently, it would be possible to detect sentences that deviate from expected word frequency distribution, by analysing the decomposed matrices that are generated by applying matrix decomposition techniques such as Singular Value Decomposition and Independent Component Analysis which are described in the paper (Skillicorn, 2005)

Another study also proved that word frequency information can be employed at the detection of word substitution or code words in sentences, where a word with similar word frequency is used to replace another word in the sentence (Fong, Roussinov and Skillicorn, 2008). The rationale was that if two words with similar word frequencies, but very different semantic information is used to replace one another, then it creates an anomaly in the sentence that can be detected using a set of measures such as Sentence Oddity, K-gram frequencies, Pointwise Mutual Information, and Hypernym Oddity. Some of these measures treat sentences as a Bag of Words. The Bag of Words is a text representation technique used in Natural Language Processing which represents the frequency in which a word occurs in a given sentence or document (Brownlee, 2017). Sentence Oddity measures the frequency of a Bag of words for a sentence with a target word removed concerning the frequency of the entire Bag of Words. K-Gram frequency refers to the frequencies of the ‘K’ number of words in a sentence where K is a numeric integer, for example, 1-gram, 2-gram and 3-gram frequencies can be measured. All these measures were then used to identify semantic differences between regular sentences and sentences containing substitutions, to identify code words. The frequency distribution of regular sentences was obtained from Yahoo! Web search interface (Fong, Roussinov and Skillicorn, 2008). A performance analysis performed on these measures in the study by Deshmukh, Deshmukh and Deshmukh (2014) reveals that when these measures were used in tandem with a random forest classifier, the model was effective in detecting code words. There was another technique which was extensively used to represent words for NLP problems during the early 2010s, that is TF-IDF, which stands for Term Frequency and Inverse Document Frequency. Such numerical representations of words in sentences are called word embeddings. The Term Frequency measures the number of times a term appears in a document concerning the total words in a document, while Inverse Document Frequency measures the rarity of the term in a given document. The formula for Term Frequency is as follows:

A black text on a white background

Description automatically generated

Figure ‑ Term Frequency Formula (Karabiber, 2023)

The formula for Inverse Document Frequency is as follows:

A close-up of a text

Description automatically generated

Figure ‑ Inverse Document Frequency Formula (Karabiber, 2023)

The final formula for TF-IDF word embedding technique is as follows:

A black text on a white background

Description automatically generated

Figure ‑ TF-IDF Formula(Karabiber, 2023)

Both Bag of Words and TF-IDF word embedding techniques have several disadvantages, the major one being that they do not value the order of sentences and do not take into account the semantic meaning of the word in the sentences in which they appear (*Word embeddings in NLP: A Complete Guide*, no date). Due to these reasons such word embeddings were found to be less effective for tasks where understanding the semantic meaning of words in sentences was essential. This problem was solved with the innovation of the Word2Vec embedding technique (Mikolov *et al.*, 2013).

## Code word detection using Word2Vec word embeddings.

Mikolov *et al*. (2013) proposed a revolutionary word embedding technique that not only generated word embedding vectors for words but can also understand the semantic relationship between words, allowing machines to process language more effectively than before. The paper also introduces the concept of cosine similarity to determine how similar or dissimilar any two words are. Words with high cosine similarity are said to be semantically similar to each other, in other words, they appear in sentences with similar context. This is possible by measuring the cos(angle) between vector representations of the words in a higher dimensional space. Word2Vec model is created by training an artificial neural network on unlabelled text data using one of the two algorithms – Continuous Bag of Words(CBOW) or Skip-Gram (Mikolov *et al.*, 2013). In the CBOW algorithm, the model takes the context of surrounding words as input and tries to predict the current word. On the other hand, the Skip-Gram algorithm takes as input a target word, which is the current word and generates surrounding context words (*Word embeddings in NLP: A Complete Guide*, no date). The final weights obtained from the model after the completion of training are the final word embedding vectors.

Word2Vec embeddings have been used in several studies related to code word detection. Yuan *et al*. (2018) proposed a system they called the “Cantreader” that can automatically detect and identify Dark Jargons on darknet forums. Dark Jargons here refer to code words used for illicit substances on darknet forums. Cantreader attempts to identify code words by analysing the semantics of the word’s usage in communication in darknet forums and comparing that with the semantics of the word’s usage in legitimate communication. To do this comparison, they have made use of two different text corpus. The first one is the dark corpora which consists of communication from dark forums such as SilkRoad, another is the benign corpora which is a combination of text data from Reddit (legitimate corpus) and Wikipedia(reputable corpus). Unlike the traditional technique of generating Word2Vec embeddings, the researchers of this study have developed a neural network that doubles the number of input layers and takes input from both the corpora and outputs two vectors for each word, each vector corresponding to the word’s relation with other words in the each of the two corpora. Code words are then discovered by comparing the semantic similarities. If the semantic similarity between the word in the dark corpus and the legitimate corpus is less but the for the same word, there is a high semantic similarity between the legitimate corpus and reputable corpus, then the word is said to be a code word (Yuan *et al.*, 2018). After code words are identified they then use a classifier to determine the probability of a hypernym relation(is-a) between the code word and a set of target keywords or hypernyms. This model achieved a recall of 77.2% and a precision of 91% (Yuan *et al.*, 2018). This was by far the most advanced model for code word detection during this period.

However, there were other attempts at code word detection as well that made use of Word2Vec embeddings. Hada *et al*. (2021) proposed a technique to detect code words in microblogging websites by leveraging the power of Word2Vec embeddings and the cosine similarity of the word vectors. They used an approach like the one used by (Yuan *et al.*, 2018) in their study on detecting dark jargon, wherein the code words were identified based on the difference in word usage of the words in two different corpora, the good corpus and the bad corpus. However, the problem they are trying to solve here is the nature of posts on microblogging sites. On sites like Twitter, the tweets are of very short character length and consist of informal language that frequently uses slang terms and acronyms and abbreviations. After generating the word embeddings for words in both corpora, code words were then detected using a custom algorithm that compares the similarity of the same word in two corpora and looks for differences in their word usage (Hada *et al.*, 2021). All these models require training of their models on a sufficiently large enough dataset to find out differences in word usage of the word in different corpora. Without such a dataset identifying code words in real-time will not be possible as finding semantic differences for new words that may be used will not be possible, as they may be missing in either the good or bad corpus that was used during training.

Different from the above techniques, Magu *et al*. (2018) proposed a technique to detect code words in Euphemistic Hate speech that uses word embeddings from the Word2Vec model (Mikolov *et al.*, 2013) to build a hate code network. Tweets from Twitter were used to generate these word embeddings. The hate code network consisted of words on the nodes and the edges represented the cosine distance between the words. Network analysis was then performed on this hate code network to obtain properties such as the number of edges, clustering coefficient, and average degree. Code words are then identified by using word ranks obtained from centrality measures such as Eigenvector centrality (Magu and Luo, 2018). This technique relies on hate code network analysis and requires large dataset. Another drawback is that performing such a network analysis in real-time will not be possible.

## Code word detection using GloVe word embeddings.

GloVe (Pennington, Socher and Manning, 2014) was another popular word embedding technique that was used to represent text using numeric vectors. Unlike Word2Vec (Mikolov *et al.*, 2013) which uses neural networks to generate word embeddings, GloVe uses matrix factorization technique based on global word co-occurrence data in the corpus. It captures global context of the words unlike Word2Vec’s Skip-gram architecture that only captures local context. (Pennington, Socher and Manning, 2014). GloVe word embeddings were used in detection of code words in fraud investigation in a study by Zee *et al.* (2021). This paper also uses BERT word embeddings, BERT will be explained later in a subsequent section. In this study, they used GloVe (Pennington, Socher and Manning, 2014) pre-trained embedding vectors in conjunction with a Bi-directional LSTM neural network to identify code words that were out of context in their sample sentence. Pre-trained word embedding vectors are pre-computed vector representations of words that have been trained on large corpora of text. The usage of pre-trained embeddings reduces the effort required to collect data and generate word embeddings from scratch (Mwiti, 2021). A bi-directional LSTM is a neural network architecture that consists of two LSTM layers, that process the input in both forward and backward directions (Bidirectional LSTM in NLP - GeeksforGeeks ). This GloVe-based bi-directional LSTM model achieved an accuracy of 80% on code word detection tasks (van der Zee *et al.*, 2021). Although this may appear to be impressive, the code words identified were known words which were replaced in the original dataset to prepare a synthetic dataset, on which the model was trained on. With the ever-evolving nature of code words, detection of unknown code words using this technique may be ineffective.

## Overview of FastText

A team of researchers at Facebook AI proposed a technique that extents the skip-gram architecture of the Word2Vec model(Mikolov *et al.*, 2013), which takes uses subword information by representing each word as a bag of character n-grams during training of the model. For example, a 3-gram representation for the word “Whale” is as follows: “wha”,”hal”,”ale”. Each n-gram would then have a corresponding vector representation and the final vector for the word would be obtained by summing up these individual n-gam representations. This is especially useful to handle rare words or misspelled words as is the case with some code words, because in a regular word2vec model, a misspelled word may not have a word embedding since the model doesn’t know its corresponding word vector generated during training (Akdogan, 2021). FastText was a library that was developed by the Facebook AI research team which is based on the same principle (Joulin *et al.*, 2016), and it can be used for text classification and word representations. This library can train a model with a large dataset in just a few minutes, without the need for tremendous resources (Joulin *et al.*, 2016). FastText has been extensively used in various NLP tasks and one study even proved that FastText outperformed Word2Vec at a sentiment analysis task related to hotel reviews(Khomsah, Ramadhani and Wijaya, 2022). For this research, I have made use of the FastText library to train a model that is capable of classifying sentences as sentences containing code and normal sentences.

## Transformer-based models

In 2017, Vaswani *et al.* (2017) proposed their transformer architecture that revolutionized the field of Natural Language Processing and Artificial Intelligence. It is considered as the foundation model as it laid the architectural foundation for the development of sophisticated Large Language Models. The most important concept introduced in this paper is the self-attention mechanism and hence the paper is aptly titled as “Attention Is All You need” (Vaswani *et al.*, 2017). Self-attention mechanism allows the model to capture long-range dependency and relationship between words that are farther apart in the sequence, this is something that previous models that used Recurring Neural Network (RNN) and Long Short-Term Memory (LSTM) network could not achieve. Another important characteristic is that the unlike previous models, the word embeddings in Transformer models also captures positional information. This is done by adding positional encoding to the input embeddings and this helps in capturing the relative or absolute position of the token in the sequence (Vaswani *et al.*, 2017). An illustration of the transformer architecture is shown below:

A diagram of a software algorithm

Description automatically generated

Figure ‑ Transformer Architecture(Vaswani *et al.*, 2017)

The inputs are fed to the encoder layer on the left which is responsible for processing the input sequence and extract meaningful representation. On the right-hand side, we have the decoder layer which is takes in the hidden states produced by the encoder and generates and sequence of output tokens (Vaswani *et al.*, 2017).

One of the most important properties of transformer-based models is the ability to perform transfer-learning. It is based on the idea of re-using a base model that has been pre-trained on a large dataset, for a new niche task. The base model already contains weights and biases learned during the training on the original task. This base model is then trained again on the new task specific data by adding a few task specific layers (Brownlee, 2023). This reduces the resources and time required to train a transformer-based model from scratch.

BERT -

## Code word Detection using BERT

## Large Language Models and their key properties

References

1. Example appendix
   1. Appendix subsection
      1. A subsubsection
2. Another appendix
   1. A subsection
      1. A subsubsection