Detecting Anomalies in User Communication in an E-commerce Application
Applying a Clustering Algorithm to Vectorized Text Messages

Jonas Nabseth
Abstract

Detecting Anomalies in User Communication in an E-commerce Application

Jonas Nabseth

Many applications that allow text based communication between users are troubled with malicious content. This thesis presents a system for detecting such behaviour in an E-commerce application. The system is based on an algorithm for anomaly detection which is trained using messages sent between users in the application. Preprocessing of the text is performed using the NLP-toolbox Glove. The resulting word embeddings are used to create numerical representations of messages, which are then used as input to a clustering algorithm based on K-means. Vectors positioned far away from existing clusters were considered anomalies. This report assesses performance of this system, and relates this to the performance achieved with an existing approach.
Contents

1 Introduction ................................................. 1
   1.1 Project Description ................................... 1
   1.2 Report Outline ....................................... 1
   1.3 Delimitations ....................................... 3

2 Background .................................................. 4
   2.1 Natural Language Processing .......................... 4
       2.1.1 Bag-of-Words .................................. 4
       2.1.2 Word Embeddings ............................... 5
       2.1.3 Matrix Structures ............................... 5
   2.2 Dimensionality Reduction ............................ 7
       2.2.1 Singular Value Decomposition ................... 7
   2.3 Clustering Analysis ................................... 8
       2.3.1 K-means ....................................... 8
   2.4 Anomaly Detection .................................... 8
       2.4.1 Training ....................................... 9
       2.4.2 Tuning ....................................... 10
   2.5 Evaluation Metrics ................................... 10

3 Related Work ............................................... 12
   3.1 Customer Segmentation based on Behavioural Data in E-marketplace .......................... 12
   3.2 Traffic Anomaly Detection Using K-Means Clustering ....................................... 12

4 E-commerce Application Information ................... 13
   4.1 System Structure ..................................... 13
   4.2 Usable Data ......................................... 13

5 Implementation ............................................. 14
   5.1 Data Preprocessing ................................... 14
       5.1.1 Word Vectors .................................. 14
       5.1.2 Message Vectors ............................... 15
   5.2 Anomaly Detection Algorithm ........................ 16
       5.2.1 Cluster Distance Measure ....................... 17
       5.2.2 Training Details ................................ 19
       5.2.3 Validation ..................................... 19
   5.3 Naive Filter ......................................... 19
6 Evaluation
6.1 Bootstrapping ................................................. 21
6.2 Word vector Accuracy ....................................... 22
6.3 Cluster Analysis ............................................... 23
   6.3.1 Proof of concept ......................................... 23
   6.3.2 Real data ................................................. 25
6.4 Number of clusters ........................................... 27
6.5 Testing .......................................................... 29
6.6 System and Naive Filter Comparison ....................... 29

7 Discussion
7.1 Insufficient Message Contents ............................... 31
7.2 Vague Classification ......................................... 32
7.3 Nonexisting Words ........................................... 32
7.4 Keywords Related to Malicious Behaviour .................. 33

8 Future Work
8.1 Labeling Behaviour .......................................... 34
8.2 Detecting new Behaviour .................................... 34
8.3 Grouping Users ............................................... 35
8.4 Combining Naive Filter and System ....................... 35
8.5 Cluster Item Descriptions ................................... 36

9 Conclusion ...................................................... 37
1 Introduction

This section introduces the project described in this report. It contains background information about the subject, motivation for the project, an outline of the report, as well as the delimitations purposefully set in the initial phase.

1.1 Project Description

The goal for this project is to build a system that can detect malicious behaviour in an e-commerce application, with the purpose of preventing it in the future and thereby improve the user experience. The application has continuously grown in popularity since it was first launched, now having tens of thousands of active users. Such a large amount of users in turn results in a large number of purchases made. An issue is that a significant number of these purchases are not performed through the application using the built in purchasing service. Instead, the users are initially in contact via text messages, while the actual purchase is then performed externally by swish or similar.

One occurring scenario is that the two parties agree upon sending payment through swish, where the payment is to be sent after the buyer has received the item. The fraud is then performed and completed when the buyer never sends payment after receiving the item. The opposite also occurs, where a user pretends to have an item that is in fact not in the users possession. Payment is then received without sending an item to the buyer.

Since there is no available purchase history, the administrators of the application is dependent on the victims reporting the users responsible of said behaviour in order to ban them from the application. A system that is able to predict where such a situation might occur before it is followed through would therefore be of great assistance. One possible basis for such a system would be the messages sent in the chat between users, and in this project building such a system is attempted.

It is common for companies to apply systems utilizing anomaly detection algorithms to detect fraud in applications. These algorithms however oftentimes expect input in the form of numeric values or a preset of possible alternatives. The input to the system in this project is instead text messages, which has a dynamic format. It is therefore necessary to convert the messages into a tokenized vector representation before the algorithm can be applied. The messages can after conversion be utilized as input, which is done by treating them as transactions. An anomaly detection algorithm is then applied to determine which transaction is non-normal, or faulty, which alerts the system that suspicious behavior is taking place.

1.2 Report Outline

The report consists of 9 sections, described briefly below.
• Introduction provides description and motivation for the project, as well as the delimitations set initially.

• Background contains relevant theoretical information and describes key concepts used throughout the report.

• Related Work describes similar projects done in the field.

• E-commerce Application Information details how the retail platform is built and structured, as well as the data that was available to use during the project.

• Implementation describes how the system was implemented, including technical details.

• Evaluation presents the performance of the finished system.

• Discussion includes reasoning about the resulting performance.

• Future Work presents possible extensions that can be implemented and how the system can be used for other similar tasks.

• Conclusions summarizes and concludes the work.
1.3 Delimitations

In order to prevent malicious behavior from occurring, preventative actions must be taken as soon as such behaviour is detected. This would require analyzing the messages between all users in real-time. Constructing such a real-time implementation would result in a pipeline structure with additional support for detecting changes to the data base. It would also require the system to be implemented as a separate service with read access to the data base.

A pipeline structure like the one described above is beyond the scope of this project. Instead, the algorithm will be trained using an existing, static dataset, and then tested with data known to be from banned users and manually created test cases.
2 Background

This section describes relevant background information and main concepts used throughout the report.

2.1 Natural Language Processing

Natural language processing, NLP for short, is a field of research that has been studied for decades. It is a relatively broad term detailing how interpretation of human language by a machine is to be performed[1]. An article called "Intelligence", composed by engineer and inventor Alan Turing to determine the intelligence of machines, in which the famous Turing test is described, is an example of early research in the field.

This early research were a lot of the times based on rule-based algorithms, which proved to be relatively unreliable. In more recent times a majority of research has instead been done based on machine learning[2]. The statistical models introduced, which makes probabilistic decisions based on weights assigned to the input data, proved to be more reliable when the input was dynamic and unfamiliar.

Before any models can be applied, the text must be processed and transformed into a format that can be interpreted. The pre-processing phase oftentimes involves separating the text into terms, created from individual words or sentences. Using these terms, the structure of the text can then be represented numerically, with the preferred format for the numeric data often being that of a matrix.

2.1.1 Bag-of-Words

The purpose of the bag-of-words-model is to give insight about the number of occurrences of certain words in a set of text, or corpus. When applied, each text input is defined as a row in a matrix, with each column representing a word as a term. The numeric values in each matrix entry is the number of times each term occurred in the respective input. Each text input has thereby been converted into a numeric vector format corresponding to each respective row.

This implementation discards common elements in human language that might be relevant for an accurate interpretation, such as grammar and sentence structure. One method of combating this is to utilize the so called n-gram model[3], where n in this instance denotes the number of succeeding words that should be used as individual terms. By choosing an appropriate value for n, more information regarding sentence structure is retained, which might produce a more reliable result when a model is applied[4]. Text2vec’s bag-of-words implementation supports the n-gram method, as well as some additional advanced extensions.
2.1.2 Word Embeddings

Word embeddings is a powerful technique used in NLP. It is used to map terms, such as words or phrases, to numerical vectors\cite{5}\cite{6}. The concept of word embeddings gained popularity in 2013 when a tool called word2vec developed by Google was released, which uses a neural network implementation. The text2vec package implements another word embedding algorithm called Glove\cite{7}. Instead of a neural network, Glove uses gradient descent to create embeddings. Both algorithms come with pre-defined models for creating word embeddings, created from large amounts of data. During this project however, custom embeddings based on the corpus of all messages were instead created, hence pre-trained models could not be utilized.

The Glove algorithm uses a term co-occurrence matrix to create embeddings, and has in the text2vec package been specifically optimized for this type of matrix. The word vectors are created by applying gradient descent, which in the text2vec package automatically uses all cores on the machine, unless specified otherwise. The implementation supports the option to specify the length of these vectors, ranging from a length of 50 elements up to over 200. In order to decide the optimal word vector length one must perform extensive testing, as longer word vectors do not necessarily result in better performance.

2.1.3 Matrix Structures

The matrices containing the numeric data are usually structured as either a document-term matrix or a term co-occurrence matrix. The required structure depends on the algorithm to which the matrix is used as input.

The document-term matrix, commonly shortened to dtm, is a matrix structured with the rows represented by individual documents, and the columns represented by individual terms. The document rows in this instance represent segments of text divided according to a property, e.g. each document is a speech, a message, a group of messages, etc. The choice of document content is decided based on between what data one would like to determine similarities. The term columns in the matrix represent the individual phrases found in the documents combined, called a vocabulary. For example, if the documents were composed of a total of 10 different phrases, the matrix would have 10 columns. In its most basic form, the terms are single words, where all words are represented disregarding of the times they occurred in the documents. More advanced methods include searching for collocations in the vocabulary, which are combinations of words or phrases occurring multiple times that are then used as terms, and pruning terms that occurred less than a specified number of times. The elements in the matrix are the number of times a term exists in the corresponding document. If e.g. the first term occur 3 times in the first document, then \([0,0] = 3\). 

\[ \text{If } e.g. \text{ the first term occur 3 times in the first document, then } [0,0] = 3. \]
The term co-occurrence matrix, henceforth referred to as tcm, is a matrix where both rows and columns are represented by terms found in all documents combined. The elements in the matrix are numeric values, similar to the document-term matrix, but differing in that they instead represent the number of times each term was present in the same context as another term. In text2vec, context is defined as an area, represented by a window of a certain size before and after the term. The weight assigned to words also differ depending on the distance to the word.

The final system implements word embeddings as the method of choice for preprocessing of the data. This report will therefore focus on the term co-occurrence matrices, as this is the matrix structure used with word embeddings. The document-term matrix is of interest however, as the bag-of-words method influenced the project in the initial stages.

For the initial bag-of-words method and its more advanced extensions, the document-term matrix was used. Later, when processing the input as word embeddings, a term co-occurrence matrix was required.
2.2 Dimensionality Reduction

When processing large amounts of text in order to represent it in a numeric format, the resulting matrix will oftentimes be very large. That is, each record will most likely contain a number of features, or dimensions, large enough that visualizing it in 3D is not possible. In a matrix, this would correspond to the number of columns. In order to visualize such a matrix, it is therefore necessary to reduce the number of dimensions to three.

One technique for performing dimensionality reduction is called principal component analysis, henceforth referred to as PCA. After applying this technique to a set of data, a new set containing info regarding the internal differences of the original dataset is obtained. PCA is used to determine the largest of these internal differences, referred to as the best principal components. The variance of the data has thereby been determined, with the resulting set containing principal components ordered according to variance. One attribute of this process worth noting is that the resulting principal components always number equal or less than the number of dimensions in the original set. To reduce the number of dimensions in the data, the principal components that contribute the least are discarded. So in order to reduce the number of dimensions to 3, only the 3 largest principal components are kept.

In this project, the original data will consist of a dataset in multidimensional Euclidean space. The first principal component would correspond to the line going through the mean of all points, where the sum of squared distances between the line and points have been minimized. The correlation to the first principal component is then subtracted for all points, after which the second principal component is computed using the same methodology. The remaining principal components are computed in the same manner.

2.2.1 Singular Value Decomposition

Another factorization technique sometimes associated with PCA is called Singular Value Decomposition, or SVD. When performing SVD, the original matrix is factorized into a product of three new matrices. For example, a matrix $A$ would factorize into

$$A = U \Sigma V^*$$

where the columns of $U$ and $V^*$ are respectively the so called left-singular vectors and right-singular vectors. The matrix $\Sigma$ positioned in between the two is a diagonal matrix containing the singular values of $A$, expressed as $\sigma_i$, with largest to smallest in descending order. Elements from PCA are then utilized to select a number of the largest $\sigma_i$, in the end leading to a new approximation of $A$ with a number of dimensions corresponding to the chosen number of $\sigma$. 

7
2.3 Clustering Analysis

Clustering analysis involves partitioning a given set of data into clusters based on similarity. Similarity in this sense is based on how the algorithm used interprets the data in the dataset, and may differ depending on which algorithm is used. Due to this reason there are several different algorithms used for clustering which might in turn cluster the same data differently. Which one to use depends on the task one wishes to perform and might not be obvious initially.

2.3.1 K-means

K-means is one of the most widely used clustering algorithms today. The algorithm operates by iterating multiple times over the dataset until optimal clustering has been achieved. This is done through a series of steps[8]. Firstly, the specified number of cluster centroids are placed in the data space, where the starting position of the centroids depends on the initialization method used. One of the most common methods is the so-called Forgy method, which randomly selects k data points from the dataset, where k being the specified number of clusters, and uses these as initial centroids. Secondly, the distance from all data points to the centroids are calculated, and points are assigned to the centroid which is closest. Thirdly, the mean of all data points contained in the clusters becomes the new centroid of the respective clusters, and the data points associated with the new centroids are updated accordingly. The second and third steps are then repeated until the algorithm reaches convergence[9].

One disadvantage with using K-means is that one has to manually choose the number of clusters. The optimal number of clusters is not always obvious, and extensive validation might have to be performed before the optimum is discovered. One method for finding the number of clusters to use is called the elbow method. This involves plotting the performance of the algorithm on a graph for different values of k, where k is the number of clusters, and choose k based on where the plot forms an elbow shape.

Another disadvantage is the random initialization of the cluster centroids. This makes the algorithm inconsistent in the sense that different runs on the same data may result in clusters containing different elements.

2.4 Anomaly Detection

Anomaly detection is the subject of detecting rare data. It is related to outlier detection, which focuses on detecting data that does not fall into proximity of the majority of other data points[10]. In density based analysis, anomaly detection is done by looking at the distance of the potential anomaly to the closest grouping of other data points. If the data point is determined to be a certain distance away from the nearest grouping or cluster of other data points, it is labeled as an anomaly. How the anomaly is then handled
depends on the application and what purpose the anomaly detection system might have. In the course of this project, it would cause the system to alert an administrator. The administrator would then decide, based on the data contained, what course of action should be taken.

Even though the data point is not behaving as the majority of the data, it might have valid reasons for its positioning. A real world example of this is a measurement from a sensor on a weather station that deviates greatly from the expected value, but still falls within the range of what is possible. A density based anomaly detection system would treat this data point as an anomaly, but analysis of the data would show that it is simply an unusual event.

An anomaly might be part of a sudden increase in similar behaviour not previously seen, and will in that case be close to similar anomalous data points. Such a group might be the result of an attempt to mask anomalies by an outside party as a pattern that is commonplace. It might also on the other hand be a new data pattern not previously seen that in the future will be occurring regularly in the application. In density based analysis, this poses the question regarding how often the algorithm should be retrained in order to take into account new patterns in the data, while retaining the ability to detect anomalous groups. For example, a clustering algorithm retrained with anomalous data points will treat such a micro grouping as an additional cluster, and further anomalies similar to the previous ones will pass unnoticed.

During this specific project, a text message was treated as an anomaly if the language used was undesirable or showed signs of malicious behavior. It had to be differentiated from a text message that contained e.g. unusual words or a lot of spelling errors, as these would also be treated as anomalies by the algorithm.

### 2.4.1 Training

In order for a system to be able to detect anomalies, its underlying algorithm needs to be properly trained. In the context of k-means this involves defining the clusters of the data. The clusters are dependent on the data used in the clustering process, but also on how many clusters that should be created. Determining the optimal number of clusters is known as tuning, more on this below. In combination with anomaly detection, the training process consists of creating clusters from specific training data, that are then used as a measurement for new data points not previously present. A short summary of the training process for an anomaly detection system based on density analysis is as follows:

The algorithm is first trained on a large set of data which contains only data considered non-anomalous. In doing so, the new data points are considered anomalies if they are "outside" all clusters, with the boundary for "outside" dependent on the distance measure used. Clustering using k-means is now performed, and the trained model is then used in determining the position of new data points in the space. The system then
asserts whether or not the new point is an anomaly. Further details about how these steps were implemented in this specific project is included under Implementation.

2.4.2 Tuning

The number of clusters chosen during training has a large impact on the efficiency of the algorithm. A small number of clusters leads to a large amount of data points in each, resulting in the boundary for “inside” the cluster being further from the cluster centroid. When the algorithm is then applied to new data, there is a larger space in which the data point can reside corresponding to “inside” a cluster. Even though this space is created by data points containing desired features, it results in a greater possibility for malicious data points to be located within.

When the number of clusters is large, the number of data points in each is relatively low. The problem posed above when the number of clusters was smaller is thereby addressed. However, when the space considered “inside” a cluster is reduced due to less data points, it becomes more difficult for the algorithm to differentiate between malicious data points and non-malicious data points. This is since the non malicious data points, although similar to the ones used in creation of the clusters, is still at risk of being positioned outside the boundary of the cluster space, thereby being mistaken for anomalies.

To achieve a balance in the number of clusters and to the highest possible degree avoid the problems mentioned above, using a method for determining the optimal cluster number is necessary. The method used in this project is called the elbow method. This method produces a graph visualizing the modeling of the data, where the point with the optimal parameter tuning is visible based on the shape of the plot. This point is usually characterized by forming an angle in the graph, oftentimes resembling an arm bent to a degree, hence the name Elbow method. Using this information, determining the number of clusters that returns the best modeling is possible, where adding additional clusters would not marginally improve the result.

2.5 Evaluation Metrics

There are standardized metrics used in the industry to evaluate the performance of an implementation. Below is a list of the metrics used during this project, accompanied by a short description of their properties, and how they are related.

- Specificity

  - The fraction of actual negatives correctly identified. Also known as ‘True Negative Rate’.
• Fall-out
  – Defined as 1 - Specificity.

• Recall
  – The fraction of positives successfully retrieved. Also known as 'True Positive Rate'.

• Precision
  – The fraction of correct results from all retrieved.

• Accuracy
  – The fraction of correct results from total.
3 Related Work

This section describes earlier work in the field similar to the work detailed in this report.

3.1 Customer Segmentation based on Behavioural Data in E-marketplace

In 2017, Andrew Aziz did research into the possibility of grouping users active in an e-commerce application based on their behavioural data[11]. The main goal of Andrews project was to create a personalized feed for each user, where the feed was to be based on the user group to which the user had been assigned. To achieve this, a user-brand ratings matrix was constructed, detailing the rating each user had given each brand. Dimensionality reduction was performed on the user-brand ratings matrix using Principal Component Analysis[12], reducing its number of dimensions to three, before it was used as input to a clustering algorithm. K-means was the clustering algorithm used, with the optimal number of clusters determined to be 3, after validation with both the Silhouette score[13] and the Elbow method[14]. Utilizing this method allowed for each user to be represented as a dot in a three dimensional space. Users were then placed in clusters with other users displaying similar behaviour, where each cluster determined a user group.

Andrew was able to showcase several interesting properties in the clustering analysis, including the fact that there exists an angular correlation between users preferred brands.

3.2 Traffic Anomaly Detection Using K-Means Clustering

Munz et al. presents a method for detecting malicious traffic among a number of flow records[8]. The flow records used as input data are in the article defined as a stream of IP-packets. They can be identified by an IP-tuple, and are available using the Cisco Netflow[15]. The transport protocol used, together with the accompanying port number, determines how each record is classified, and were later used to create datasets over defined time intervals. Based on anomaly detection using k-means[16], it detects anomalies by calculating the distance between the potential anomaly and the cluster centroids. Among the different possible methods presented in the article for calculating this distance, weighted euclidean distance was used during the evaluation.

The authors clustered the training data using k-means with k=2, with the motivation that two clusters would correspond to one containing normal traffic, and another containing anomalies. Both clusters were in the evaluation described to be close in proximity, and the authors therefore argued the training data did not contain anomalous data.
4 E-commerce Application Information

At the time of this report, the retail application’s registered users are numbering in the tens of thousands. This section briefly describes how the application is structured and how the necessary data was mined from the database.

4.1 System Structure

The view a user initially meets when opening the application features three tabs. The middle tab, which is pre-selected upon launch, displays a feed of the currently trending items. Which of the items that are displayed on top is dependent on, apart from when the item was published, on the combined weight of the likes the item has received. It is also possible for a user to pay a small sum to be shown at the top of the feed for 24 hours. The right tab features a feed similar to the one described above. In this feed however, the items are sorted after the latest upload, meaning no importance is placed on how the item is received from other users. The left tab displays a feed of users called ”power users”, manually selected for display by the admins. Which users are selected for display depends on the number of followers, number of likes received on uploaded items, etc. At the top of the initial view there is also a search bar which is used to filter on a specific term. A term in this case can be e.g a specific type of clothing or a hashtag. It is also possible to enter the name of a user, in which case the result only displays items uploaded by the user specified.

To facilitate navigation in the application, there is also a permanent menu bar at the bottom. This enables access to a more extensive search function with a selection of pre-defined categories of clothing, as well as the camera, chat functionality, and profile information.

4.2 Usable Data

As previously mentioned, the system was validated using static data. The relevant data for this project was stored in a PostgreSQL database and was provided in the form of an sql dump. Building a local database from the dump enabled access to the tables containing the necessary data.

One of the tables provided contained messages, with the number of messages amounting to around 3.000.000. The messages were extracted and filtered according to the author id being present in either the banned or non-banned user table. The resulting datasets were then used throughout the project to create benchmarks for the algorithm.
5 Implementation

This section details how the system was implemented. The implementation is divided into two phases, the data pre-processing phase and the anomaly detection phase. This section also details how a naive implementation for detecting malicious messages was implemented, based on filtering messages on predefined keywords, which was used for comparison.

5.1 Data Preprocessing

As previously mentioned, the anomaly detection algorithm expects input in the form of vectors, where each vector represents a text message. Therefore, it was necessary to have a module for performing the transformation of the input. Determining the optimal vectorization technique among all available options required extensive testing. In the course of this project, both bag-of-words and word embeddings were used during the process of determining the optimal anomaly detection algorithm. The bag-of-words method was implemented initially, and therefore had a large influence on how the rest of the system was implemented. After extensive validation using bag-of-words for pre-processing of the data, no conclusive results were measured, and word embeddings was therefore instead chosen as the method for performing this task. The algorithm supported by text2vec that implements this is called Glove, and is described in greater detail above under Word embeddings. This section details how the word vectors produced by Glove were used, and how additional functionality was then implemented to produce vectors corresponding to full messages.

5.1.1 Word Vectors

By following the instructions for the text2vec package, the built in support for the Glove algorithm was implemented. Pruning of the corpus was performed before word vectors were created, removing terms appearing less than 3 times. As word embeddings are all about finding related words, it is not possible to create embeddings for a word only appearing once or twice. The algorithm must be able to determine the context of words, which require the word to be present in different contexts. However, in order to avoid a large part of words correspond to zero-vectors, the pruning threshold was set relatively low. A larger part of the new words used thereby have corresponding word vectors, but also result in some word vectors not as accurate as they would be if a higher degree of pruning was performed beforehand. With a larger corpus and a system focusing on creating clusters, rather than the data outside the clusters, a higher degree of pruning would have been possible, leading to more accurate word vectors.

After pruning, each unique term in the corpus was assigned a corresponding vector. The vector size was in the end defined to be 50, meaning each word vector contains
50 numeric elements. Due to how the vectors are created, it is possible to perform arithmetic operations on them to receive words related in meaning. An example to intuitively describe this property is: "king" – "man" + "woman" – "queen", where the operation is performed with each words corresponding word vector[17]. This obviously requires the model to be properly trained, and the terms "king", "man", "woman", and "queen" to be present in the corpus in order for them to be associated with proper word vectors. The property making such operations possible did however make up the foundation for the next stage of the pre-processing phase, namely how to create message vectors.

5.1.2 Message Vectors

An issue during this project that required handling was how to create vectors from full messages. The word vectors produced by Glove worked well for individual terms, but there was no built in support for creating vectors associated with combinations of multiple words. Therefore, implementing support to combine word vectors into corresponding sentence vectors, or message vectors, was necessary. To achieve this, a method called pooling was utilized. In short, pooling is used to reduce the size of a matrix while still retaining key elements of the original. For convolutional neural networks it is done by applying a square window, commonly of size $2 \times 2$, and traversing the matrix.[18] In max pooling, the maximum value found within this square is transferred to a new matrix, where the $2 \times 2$ section is now represented by a single element, consisting of the maximum value. In doing so, both the number of rows and the number of dimensions are reduced. In this project, the pooling was done with respect to the rows, and thus without changing the number of dimensions of the matrix, as this is instead done at a later stage.

After word vectors, called component vectors, have been combined into sentence vectors for all messages, every row in the final input dataset will correspond to a message vector. A majority of messages are different from each other, and most importantly for this project, are composed of different amounts of words. Due to how dimensionality reduction is implemented, and to avoid an extensive padding in the matrix consisting of zeros, it was desirable that all message vectors in the dataset retain the size of their component vectors. One initial alternative method for creating message vectors was thereby discarded, namely to append the corresponding word vector for each word present in the message to a new message vector. This would have resulted in message vectors being of the size: word vector size $\times$ number of words in message, thereby varying in size, and also would have required padding the vectors with zeros. Instead, the word vectors arithmetic properties mentioned above were utilized.
To create message vectors, the word vectors were combined, in the end producing a message vector of the same length as the word vectors used initially. The chosen method of combining the word vectors was to select the elements with the largest magnitude in each dimension. As a step by step process, it is as follows: Firstly, a message that is to be converted into a vector is selected. Secondly, the words are iterated over, placing each corresponding word vector into a new row in a matrix. Thirdly, the element with the largest magnitude in each column of the matrix are used to create the message vector, discarding all other entries. Lastly, the message vector is placed as a row in the input matrix. This approach solves the problem of having a different amount of words in each message, while still allowing for each word to affect the resulting vector.

\[
\text{Word vector} = \begin{bmatrix}
0.384 & 0.751 & 0.429 & 0.875 \\
\end{bmatrix}
\]

**Figure 3:** An example word vector.

\[
\begin{bmatrix}
\nu_1 \\
\nu_2 \\
\nu_3 \\
\nu_4 \\
\end{bmatrix}
\begin{bmatrix}
0.384 & 0.751 & 0.429 & 0.875 \\
0.219 & -0.981 & 0.623 & -0.423 \\
0.813 & 0.589 & -0.512 & 0.154 \\
-0.560 & 0.123 & 0.347 & 0.645 \\
\end{bmatrix}
\]

**Word vectors**

**Figure 4:** An example word vector matrix. This would be the result of a message containing four words, and a significantly smaller size for word vectors than were used during this project.

\[
\text{Message vector} = \begin{bmatrix}
0.813 & -0.981 & 0.623 & 0.875 \\
\end{bmatrix}
\]

**Figure 5:** The resulting message vector after extracting the elements with the largest magnitude from each column of the word vector matrix.

### 5.2 Anomaly Detection Algorithm

As previously described under Background, anomaly detection is the practice of discovering data points in a dataset that does not belong. During this project, the algorithm used for discovering such data points is based on clustering analysis.

After dimensionality reduction has been performed on the dataset, all data points are represented with three features each, also referred to as dimensions. This representation allows for the data to be thought about conceptually, as it is near impossible for the
human mind to comprehend the location of a data point in a multidimensional space exceeding three dimensions. Representation in a three dimensional space also presents a more obvious advantage, which is the possibility to render a visual representation of the entire data set. In such a representation, it becomes very clear which data points differ from the majority of the set, as they will have a different position in the space.

Once the position of the data points has been determined, the algorithm calculates the distance from each cluster that would be considered the boundary between "inside" and "outside" of each respective cluster. A data point positioned a shorter distance away from a cluster centroid than the clusters own computed boundary distance is considered to be "inside" the cluster. Similarly, a data point positioned further away from the centroid than the clusters boundary distance is treated as being "outside" the cluster.

![Figure 6: Two clusters with their respective boundaries visualized, here depicted in two dimensions to facilitate visualization. The green point is the only data point outside both boundaries, and would thereby be considered an anomaly.](image)

5.2.1 Cluster Distance Measure

In order to determine the boundary between "inside" and "outside" a cluster, a standardized measure of distance had to be used. The fact that clusters contain differing amounts of data points also had to be taken into account. It was therefore necessary to quantify the total sum of varying distances of each data point to the cluster centroid. In
other words, a cluster containing several data points far away from its centroid should have a large computed distance. A cluster containing a few data points close to its centroid should on the other hand have a relatively small computed distance.

The unit of distance chosen for this purpose was the standard deviation, as it allows for quantifying the variance of a data set. The data set would in this case be all distances between the cluster centroid and its corresponding data points.

To determine the distance between a cluster and points in three dimensions, one calculates the Euclidean distance in 3-dimensional Euclidean space between each data point and the centroid:

\[
d(c, p) = \sqrt{\sum_{k=1}^{n} (c_k - p_k)^2}
\]

where \(d(c, p)\) denotes the distance between cluster centroid \(c\) and point \(p\). \(c_k\) and \(p_k\) refers to their respective coordinates in each dimension.

**Figure 7:** The standard deviations for two clusters in two dimensions, where each consecutive change in transparency corresponds to one standard deviation. The cluster centroids are shown in black. In this example, three standard deviations would encapsulate all data points in the cluster space.
The standard deviation for all clusters is computed in this manner. The number of standard deviations away from a centroid that defines the boundary between “inside” and “outside” is then chosen, the number of which is the same for all clusters. The optimal number to achieve the highest possible accuracy was determined in the validation stage, and is described further under Evaluation.

5.2.2 Training Details

The algorithm is first trained on a large set of data which contains only messages from users that have not been banned. In doing so, the new data points are considered anomalies if they are "outside" all clusters. Clustering using k-means is now performed. The trained model resulting from the built-in k-means functionality of the sci-py library provides additional parameters useful in the following steps. These parameters include, among other data, the coordinates of cluster centroids, as well as information about which data points that belong to which cluster. After clustering has been performed, the model is saved to an external file using the joblib library. In doing so, performing the clustering, which requires a lot of computational power, is only necessary once. It is then instead possible to extract all necessary information directly from the model file, making it a very light weight system.

5.2.3 Validation

Before the algorithm is validated, the same pre-processing steps and pre-trained models as during training are performed and applied to the validation data, to ensure consistency. If deployed to a live environment, the system would take a single text message as input and see if it is inside the computed distance of any cluster. If it is not, the system would alert about a potential anomaly.

The validation was performed using the same methodology, scaled up to 10000 data points. The module used for validation returned the number of data points outside all cluster distances. The computed distances were created by multiplying values ranging from 0.3 up to 4.8, incremented in steps of 0.3, with the standard deviation of the clusters. The amount of data points outside all clusters for each value were recorded and later used to determine the True Positive rate, or TPR, and the False Positive Rate, or FPR, along with precision and accuracy.

5.3 Naive Filter

In order to evaluate the anomaly detection algorithm, the results were compared with those of a naive filter algorithm based on keywords. The naive filter algorithm was implemented by specifying certain terms that, upon any of which being present in a message, would flag the message as malicious. This methodology is different from
the anomaly detection system in that it does not require any prior training, as it is not reliant on the contents of another data set. It also means the filter does not require extensive parameter tuning. The anomaly detection system is dependent on several parameters, including distance measure from the clusters, and several others specified before training. These are however irrelevant for the filter, as the only dependency it has is the size of the data set, which will increase or decrease the number of necessary iterations. Which terms to use were chosen based on which ones were considered likely to appear in the context of fraud. Since the terms were chosen based on speculation of the parties involved in this project, it is likely that terms which would have improved the result were not included. Also, the module does not parse the message in any way, and might therefore not detect a word in a message that is either misspelled or has a differing grammatical structure. Furthermore, the system also alerted about a number of messages not related to malicious behavior. It is therefore possible that adding or removing some terms would improve overall performance.
6 Evaluation

Determining the optimal performance of the implementation required extensive validation and parameter tuning, this section details how this was done. Also, details about the produced word vectors and observed clusters are presented.

6.1 Bootstrapping

To evaluate the robustness of the algorithm, bootstrapping was used. Bootstrapping entails retraining the algorithm multiple times, each run with differing training data. The data was produced using sampling with replacement, with records sampled and replaced at random from the training set. After this process had been repeated multiple times on the validation set, the resulting data was used to produce ROC-plots for each run. In short, a ROC-plot is a visualization tool that helps to evaluate the performance of a system based on a couple of key parameters. In the ROC-plot depicted below, these parameters are Recall, set as the y-axis, and Fall-out, set as the x-axis. When depicting performance in a ROC-plot, a desirable result would be a plot with a deflection toward the top-left corner, where the perfect score would be in the corner itself. Such performance is rarely seen, and instead the point closest to the top-left corner corresponds to the best performance, and the parameter tuning corresponding to this point should be the one used in the finished system. The results of each run were combined into one plot, giving a clear indication of the variance of the system. The area under the curve, AUC, of the following plot is approximately 52.7%.
Figure 8: The resulting ROC-plot for the validation set after 100 runs, each time with resampled training data. As can be seen, the system’s performance is consistent for each run.

6.2 Word vector Accuracy

The total number of word vectors created form the corpus were 5322. If the words used are to have corresponding word vectors assigned, there are 5322 alternatives to use. This number seems quite large, but is in fact much smaller than the average adult’s vocabulary. Therefore, it is likely that words not included in the 5322 will be used regularly, and will have corresponding word vectors manually set to 0. Such words will not have an affect on the message in which they are present. Another issue with a small corpus is the accuracy of the word vectors. The pruning done before word vectors are created results in a trade-off between accuracy and representation. I.e the lower the pruning threshold, the more words are represented in the list of available word vectors and in turn, the lower the overall accuracy will be. This was obvious during the course of the project when the cosine similarity of different words were displayed together with the ones the system had determined were the most similar. Cosine similarity is a measurement for determining the similarity between two vectors, based on measuring the cosine angle between them. Although a lot of the words were very much related, there were also some that intuitively were not.
Figure 9: A word, shown in light blue, and its closest related words according to the system, displayed in cosine similarity. A number closer to 1 means more similar. Numbers shown in green correspond to words that are intuitively very related, yellow to words that could be considered related, and red are words that are not related.

The optimal balance for the trade-off between accuracy and representation was determined through extensive validation, and is therefore optimized for the particular data used. With a larger corpus, a higher number of represented words, and a higher degree of accuracy, would have been possible to achieve.

6.3 Cluster Analysis

In this section, clusters produced by the system are analysed. A proof of concept using handcrafted data as input is followed by tests using real data. The real data used is an example dataset much smaller in size compared to the actual training data, in order to facilitate a visual representation.

6.3.1 Proof of concept

The following is a demonstration of the clustering capabilities of the implementation. As mentioned earlier, max pooling in reference to rows was used to create message vectors from word vectors. The functionality following from limiting it to only manipulating the number of rows, while the number of columns remain the same, had to be validated. This section confirms that the implemented method is able to convert full messages into data points, and that similar messages will be positioned in closer proximity in the cluster space compared to other data points.

The following figure showcase clustering with the message vectors created under implementation using Word embeddings. The only requirement for the input data were that all words had corresponding word vectors. It was created manually in order to highlight both complete differences in content, as well as more subtle differences. Subtle differences in this instance refers to two or more messages that are identical except for one word.
Figure 10: How the messages were structured. The group of words added to message1 was different from the group added to message2, with n=5.

As is clearly visible in the figure, three distinct clusters are formed. The two larger ones, shown in purple and green, each contain five data points. A message completely different from all others is shown in red, and is as expected positioned separately.

Figure 11: The resulting clusters when message vectors have been created from their respective word vector components.
When creating clusters from messages preprocessed with Glove, their internal spread is relatively small. As can be seen in figure 11, the clusters contain data that is semantically similar, in close proximity. For an algorithm focusing on closeness based on euclidean distance, having closer proximity between similar data points is beneficial. Therefore, preprocessing the data in this manner would likely give desirable results for tasks relying on this property.

6.3.2 Real data

The new data used as input to the algorithm can be visualized together with the training data in the cluster space. Since the algorithm takes decisions based on positioning, visualizing the cluster space can aid in understanding the inner workings intuitively. Here the new data is both malicious, represented by black x-marks, and nonmalicious, represented with green spheres.

![Figure 12: The resulting clusters with the implementation run in small scale. The data used for training is shown in color. The black x-marks depicts new added data, here corresponding to malicious data.](image)

In the above image, a lot of the new data appears to be associated with the training data in the red cluster. Worth noting is that with a relatively small distance measure, no new data point would be considered an anomaly in this case.
Figure 13: The same training data as in the graph above. Instead of malicious data, the new data added corresponds to nonmalicious data, represented by green spheres.
Figure 14: All data from the graphs combined.

Visually, it appears the green spheres are dispersed more evenly across the cluster space, while the black crosses are shifted to the right. However, the amount of data used in these examples were very small in order to facilitate a visual representation, and so the results produced can not be used as basis for any conclusions. It does however help in understanding the workings of the algorithm, where a real deployment would result in a huge number of colored data points, and green spheres or black x-marks outside the clusters resulting in an alert from the system.

6.4 Number of clusters

The number of clusters to use were decided using the Elbow method, as described in Background. In the following graph, the modeling is described as the sum of squared distances from each point to its closest cluster centroid, as a function of k. When adding another cluster, the sum of squared distances will decrease, as points will be located closer to a cluster centroid. This would correspond to a better modeling for the data.
Figure 15: The elbow curve for the algorithm run on the training data, where the number of clusters is indicated by $k$. The point in the graph closest resembling an elbow shape is arguably located at $k=5$, which during validation proved to provide the best performance.
6.5 Testing

A test set was extracted from the data base during an early stage of the project. It was of the same size as the validation set, and contained records not previously seen by the algorithm during neither training nor validation. When applying the trained algorithm on the test set, the performance were unfortunately undesirable, meaning no apparent conclusions could be drawn based from the result. Performance during the testing proved to be close to random.

There are several possible reasons for the poor performance of the algorithm. Under discussion, these possible reasons are explored in depth.

6.6 System and Naive Filter Comparison

Similar to the anomaly detection system, the Naive filter failed to provide reliable performance when applied on the test set. The main difference compared to the anomaly detection system is that the recall was significantly lower. The performance of the filter on the test data is as follows:
- Precision: 0.4775
- Recall: 0.0545

In practice, this means that the system would alert about far less messages as opposed to the anomaly detection system. Reasons for the performance are discussed in the following section.


7 Discussion

This section discusses the finished system and what could be done to improve the performance. There are a few key points regarding the data that likely has the greatest impact in this regard. The two initial points both relate to the available messages themselves. The first point discusses the effect of having a lot of messages that might be lacking in sufficient information. The second point discusses the data from banned users, and how the available method of extracting their messages might have affected the result. The last point discusses the mathematical aspects of vector manipulation.

7.1 Insufficient Message Contents

A majority of communication done in the application is with the purpose of either selling an item, or buying one. A lot of the messages during this communication are from the senders perspective sent to strangers. The conversations are also started in a majority of the time by the potential buyer. Therefore, it is natural to in a lot of cases start a new conversation with either a greeting, followed by a question regarding the item, or immediately ask about the item.

These factors affect the messages sent in several ways. Firstly, a lot of the messages are very similar. There are only a few different words used to greet someone in Swedish, and therefore it can be assumed that a lot of initial messages contains a variety of these different phrases. If these phrases are sent as separate messages, they will be registered as individual vectors in the system. Also, for a majority of the items being either sold or removed, there is at least one corresponding conversation with a potential buyer, since this is how the two parties agree on price and method of payment. This means that there are one or more started conversations for every item, but only one that concludes. Therefore, a lot of information is available in regards to text for this initial stage of a conversation, while information related to later stages are lacking.

Secondly, the messages are very short. Communication in the application is to a large extent limited to the actual item in question. Since it most of the time is between two strangers, it is assumed the users generally do not begin to speak about unrelated matters, as this would be of no interest for a user with a clear motive of only buying or selling items. This further relates to the previously mentioned concern about the messages being similar, as conversations about items commonly follow certain patterns. The users also tend to keep communication short, as the objective of the conversation is very clear for both parties. The effect of this is that the vectors created for each message is built based on a limited amount of information. Longer messages, and messages that varied greatly in content, would in turn create message vectors with more distinct features. Performing clustering on such message vectors would likely result in more clearly defined clusters, and more distinct positions for malicious messages.
7.2 Vague Classification

The database featured numerous tables. The ones relevant for this project included a table for the messages, and tables for banned and non-banned users respectively. The messages for the training set, and one half of the validation set, were selected from messages where the corresponding author were not in the table containing banned users. The messages for the other half of the validation set were to the contrary composed of messages where the authors were in the table of banned users.

Following this method allowed for a validation set consisting of only messages written by suspended users. It does however not ensure that all messages contained in the resulting set are the cause of the actual suspension. In other words, it is possible that the validation set of banned messages contained large segments of messages that were all written by the same user. These users might have behaved normally for an extended amount of time, then written something inappropriate, causing their accounts to be suspended. These segments would in theory not differ much, if at all, compared to the other half of the validation set, composed of messages from non suspended users. It is likely that the validation set contained a large amount of messages that are classified incorrectly in this manner, i.e causing the algorithm to be over trained towards the validation set, rather than the task at hand.

7.3 Nonexisting Words

The word vectors used during this project were created from scratch using the Glove module from the text2vec package. Word vectors created in this way are optimal for further work on texts that are similar to the training set. One downside to this method is that words not present in the training data will not have corresponding word vectors. If the algorithm then tries to map a vector to a word that has previously never been encountered, it will result in an error. During this project this was handled by mapping nonexisting words to vectors composed of zeros. These words would thereby have no effect in later operations.

Anomaly detection is used to detect data points differing significantly from a majority of the data. Words that only appear once or a few times in a set of messages are therefore an indication of difference. However, if these words’ corresponding word vectors are composed of zeros, the data point will be placed only according to the words present in the training set, i.e the more common words.

The messages indicating malicious behaviour are likely composed of words not commonly found. The implementation in its current state does not represent these words, unless the same words are found in the training data. This might have affected the result negatively, as the data points corresponding to these messages might not have received features as distinctive as they would have otherwise. Having corresponding vectors for all words would have required the training data to be greatly extended, or to
use predefined word vectors, which would possibly have improved performance. However, the application is used for a very specific purpose. It is possible that manual creation of word vectors produces the most accurate vectors for the specific context of this project, and predefined word vectors might therefore be undesirable.

7.4 Keywords Related to Malicious Behaviour

In regards to the keyword based filter, it was not established which terms might correspond to a certain behaviour. In general, the victim of a potential fraud is usually the one that tries to get in contact with the other person. The person that performs the fraud on the other hand tends to leave the platform or ignore the victim when the fraud has been carried out. Therefore, it would probably be a simpler task to use a filter to detect when a user has become a victim, than the opposite. The general consensus between the parties involved in this project was that no specific terms can be associated with fraud. Instead, such behaviour is most likely better defined as a certain way of communicating in terms of sentence structure and choice of expression. If the filter were to detect specific terms, such as bad language, which is limited to a specific set of words, it would most likely be effective.
8 Future Work

There are both several extensions that can be added to the system, and different situations where the system could be useful. This section details in brief some of these and why they might be necessary, as well as how they could be implemented.

8.1 Labeling Behaviour

Similar to how certain placement of a data point in the cluster space is an indication of malicious behaviour, other placements can be an indication of different behaviour. Placements of multiple data points in close proximity would in that case represent a cluster of a certain behaviour. These representations might range from broad terms, e.g "Friendly", "Not friendly", to more specific, such as "Likely to buy shoes". If it is determined what behaviour a cluster's data points represent, it is possible to label the cluster accordingly. It is likely that a new data point is within or in close proximity with an existing cluster. When all clusters have been labeled, the new message can therefore also be given the same label, avoiding the necessity of an admin doing it manually.

Useful properties from such an implementation can be e.g improved recommendations for users. A user that is writing a lot about a certain topic indicates an interest in said topic. The system could utilize this information to pair users with similar behaviour and interest, improving their experience.

8.2 Detecting new Behaviour

When clustering is performed, current trends in the data are discovered. The model would then behave according to the data that was used for training when asserting new data points. This works well when the trends remain the same, as the new data points can be asserted based on the old data. If the trends change however, the pre-trained model’s performance will continuously decrease. This can be caused by the application receiving a major update or overhaul, which would cause users to behave differently, and in the context of this project cause some communication to differ. For example, a new view in the application with support for additional functionality would cause some communication between users to be about that module in question. Expressions used in such communication would not have been present in the data when the algorithm was trained, and the model would not recognize this as reasonable behaviours. Instead, if such data points tend to be positioned in separate clusters, the algorithm would alert about all of these as anomalies.

In order to avoid disturbing the performance of the system by newly emerging trends, re-training would have to be made on all data including the new points. It is likely that the new data points form a cluster not previously represented, which would
after re-training be labeled accordingly. One indication for when such re-training is necessary can be a burst in the number of alerts from the system. The data points causing the burst might be coming either from one or a few users behaving maliciously, or from many user writing messages corresponding to the new trend.

8.3 Grouping Users

In its current implementation, the system groups individual messages based on content and alerts about a specific behaviour. One individual message is in other words used to alert about the behaviour of the author. It might be possible to determine a more comprehensive understanding of the authors behaviour if more data were used. An extension would be to combine all of a user’s messages into one record, and represent this as a data point. If this is done for all users, clustering analysis can be performed, and the data points would in that case instead represent the behaviour of individual users. Data points that are placed far away from the majority would in that case be an indication of users behaving differently to others. These points might create smaller clusters in different places of the space, making it possible to label each cluster to a specific behaviour.

Data points that appear in any of these clusters would enable the system to label users as behaving similarly. It would also be possible to detect if a user is moving a certain direction, indicating that they are behaving increasingly similar to the behaviour of the cluster they are moving towards. From a technical perspective, this would require that the data point representing the user is recomputed every time the user writes a new message.

One cluster in such a space would in theory represent banned users. Their data would remain in the cluster space even after the users are no longer able to use their account, in order to enable detection of similar behaviour in the future. If this is the primary behaviour the system is attempting to detect, movement towards this cluster, or immediate placement within the cluster after only a few messages have been written, would prone the system to alert an admin.

8.4 Combining Naive Filter and System

Even though the system could be well trained to detect certain message meanings, it will not be 100 percent accurate. An anomaly detection system as described in this report would be more focused on the meaning an entire message entails, rather than preventing the use of certain words. If there are some words that the admins of the application never wish to be written in the chat, a keyword based filter designed to detect those terms specifically would be more effective.

In the case where some words are prohibited, and where the messages are also analyzed to detect certain behaviour, a naive filter and anomaly detection system could be
implemented in unison. The question this poses is whether or not the anomaly detection system should analyze a message that the naive filter has stopped from being posted. The message would in that case be stopped, but can still reveal bad intentions the author might have, and so analyzing it using the system would be desirable. On the other hand, this would require the anomaly detection system to be trained on messages containing these words, which would include them in the clustering model.

To summarize, in order for the anomaly detection system to analyze messages containing undesirable words, the same words must be included in the training data. That leads to the system not being able to detect them as anomalies, as similar messages would have created clusters in the cluster space. To analyze them, and at the same time prevent them from being sent, a naive keyword based filter would also be applied after the system. In theory this would create a very effective automatic system for preventing both certain language and behaviour in the chat.

### 8.5 Cluster Item Descriptions

The input to the system was during the course of this project always in the form of text messages. There is however no restrictions on what type of input the system expects, except that it should be clearly defined segments of text. Input could be anything from e.g film reviews, news paper articles, twitter posts to entire books, which would be clustered according to the same parameters and functionality as messages.

One such use for the system would be to cluster item descriptions. When a user uploads an item, there is a field where the item is to be described. The user specifies the category manually, but an accurate system analyzing the description would be able to place the item in further subcategories based on the text. One commonly described attribute to items is the degree of wear, where some items are well worn while others are freshly unpacked from the box. A user would in that case be able to filter on e.g more worn down clothes, which would reduce the price dramatically.
9 Conclusion

In applications that allow its users to roam freely, there will always be some that try to exploit the system. Whether this is through previously unknown bugs or through human error, limiting such behaviour will improve the user experience for all others. There is still no consensus on which method is best suitable to achieve this, and different types of systems require different solutions. In this report, a system based on anomaly detection was implemented. The data points analyzed were created from individual text messages between users that were converted into a tokenized vector representation. Using vectors of this format allowed for the use of a clustering algorithm, where k-means was the algorithm of choice due to its effectiveness and relative simplicity. The clusters were then analyzed, where the focus was on finding data points a certain distance away from any of the cluster centroids, which would indicate an outlier or anomaly.

The system created during this project was not able to distinguish a large part of the messages that were expected to be placed outside the specified boundary of all clusters. It did however prove that it was possible to create word vectors for all words present and combine them into message vectors, and also that it was possible to create reasonable clusters from these message vectors based on similarity.

There are some plausible reasons for the algorithm failing to distinguish the target data. These includes the fact that the data did not contain a lot of information, the target data not being much different from the training data, and the way the system had to handle nonexisting word vectors. If these points were to be addressed, a improved result is most likely achievable.

Automatic systems that are able to accurately detect malicious behaviour, in this case fraud, would be of great assistance in any application. It would mean less work for admins that manually control the flow of events taking place, but also lead to a near instant response time to such activities. Such a system could work either by alerting an admin as soon as malicious behaviour is taking place, or if the system is deemed accurate enough, take actions against said behaviour on its own. Another possibility is that a pop-up occurs in the chat, notifying the user that something is unusual. Given that the system is able to scan messages in real time, a decrease in the amount of users being tricked would likely follow.
References


