QUANTIFYING BEHAVIORAL COMPLEXITIES OF HUMAN AND BOT ACCOUNTS USING DATA COMPRESSION

by
DAVUT BAYIK

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
July 2022

Davut Bayık 2022 ©

All Rights Reserved

ABSTRACT

QUANTIFYING BEHAVIORAL COMPLEXITIES OF HUMAN AND BOT ACCOUNTS USING DATA COMPRESSION

DAVUT BAYIK

DATA SCIENCE MASTER'S THESIS, JULY 2022

Thesis Supervisor: Asst. Prof. Onur Varol

Keywords: Data Compression, Variational Autoencoders, Twitter, Bot Accounts, Dimensionality Reduction, Digital DNA, Complexity Analysis, Markov Chains

As the number of automated accounts grew rapidly in parallel with social media platforms gain more users around the world, there is a growing need to understand the nature of bot accounts to prevent their manipulative and misleading effects on ordinary users. This study focused on complexity analysis of users on the Twitter platform to reveal the hidden and differentiating patterns between human and bot accounts, using 14 publicly available datasets collected through the Twitter API and labelled with different annotation methods. The analysis consists of two parts, quantifying the complexity of account behavior and reducing the dimensionality of profile information. In our research, the assessment of account complexity is performed by encoding account activities into sequence of codes and compressing the repetitions and patterns about it. For the profile information, we developed a heuristic method to determine how much of an account's profile features can be compressed with minimal loss of information using variational autoencoders. The results for both parts of our analyzes are largely consistent with each other in terms of comparing complexity with different datasets and between human and bot accounts. We validated and corroborated our findings by predicting the next activity of accounts and calculating the accuracy of the predictions using discrete-time Markov Chains. Consequently, we analyzed the complexity of bot and human accounts and had complexity levels for each bot dataset we used, and we hope this study will lead to develop measures to quantify robustness of bot detection systems.

ÖZET

VERI SIKIŞTIRMA YÖNTEMLERI KULLANARAK İNSAN VE BOT HESAPLARIN DAVRANIŞSAL KARMAŞIKLIK ANALIZI

DAVUT BAYIK

VERİ BİLİMİ YÜKSEK LİSANS TEZİ, TEMMUZ 2022

Tez Danışmanı: Dr. Ögr. Üyesi Onur Varol

Anahtar Kelimeler: Veri Sıkıştırma, Varyasyonel Otokodlayıcılar, Twitter, Bot Hesaplar, Boyut Azaltma, Dijital DNA, Karmaşıklık Analizi, Markov Zincirleri

Sosyal medya platformlarının dünyada yaygınlaşıp daha fazla insana ulaşmasıyla yazılımla kontrol edilen hesapların sayısı giderek arttığı için, bot hesapların normal kullanıcılar üzerindeki manipülatif ve yanıltıcı etkilerini önlemek açısından, bot hesapların doğasını anlama ihtiyacı ortaya çıkmıştır. Bu çalışma, Twitter platformundaki bot ve insan hesaplar arasındaki gizli ve ayrıştırılabilen örüntüleri ortaya çıkarmaya çalışan karmaşıklık analizleri üzerine yoğunlaşmıştır. Twitter API ile toplanan 14 adet erişime açık, daha önce yapılan akademik çalışmalarda kullanılmış verisetleri kullanılmıştır. Bu araştırmadaki karmaşıklık analizi iki aşamadan oluşmaktadır, hesap davranışlarının karmaşıklığının niceliklendirilmesi ve profil özelliklerinin boyutunun azaltılması. Hesap karmaşıklığının değerlendirilmesi; hesap davranışlarının bir metin olarak kodlanıp, bu metin içerisindeki örüntülerin ve tekrarların sıkıştırılması ile örüntüsel davranışlarının derecesinin ortaya çıkarılması ve Varyasyonel otokodlayıcılar kullanılarak hesap profil özelliklerini en az bilgi kaybı ve en çok sıkıştırılabilirlikle ölçen bir yöntem ile yapılmaktadır. Verisetlerinin kendi arasında ve insan ile bot hesaplar arasındaki kıyaslamalar açısından, iki yöntem ile de bulduğumuz sonuçlar çoğunlukla birbirleriyle tutarlı çıkmıştır. Sonuçlarımızı desteklemek ve doğrulayabilmek amacıyla, ayrık zamansal Markov Zincirleri kullanarak hesapların bir sonraki davranışını tahmin etmeye çalışarak, bu tahminlerin doğruluğunu değerlendirdik. Sonuç olarak, insan ve bot hesapların karmaşıklıklarını analiz edip, farklı veri setleri için karmaşıklık seviyeleri elde ettik. Bu çalışmanın bot algılama sistemlerinin sağlamlığını geliştirmek için kullanılabileceğine inanıyoruz.

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr. Onur Varol for his tremendous supervision and support throughout my thesis project. His ambition, enthusiasm and belief always kept me motivated for the project.

I want to say thank you to my family for their endless emotional support. My proud father, Mehmet Ali, my proud mother, Ayşe, my dear sister, Esma Nur, and my humoristic brother, Halil.

My long lasting friends from Sabancı University, Asım, Çağatay and Burcu, thank you for always being there for me and always make me smile in my most troubled times.

To all my friends in Data Science Master's program, thanks for your colloborations and ideas during my Master's education. It was a great pleasure to meet and chat each one of you.

Thank you Sabanci University, for providing me a wonderful education, broaden my horizon and vision, giving me a chance to study with you for both my Bachelor's and Master's education.

Last but not least, I want to thank all members of VRL Lab one-by-one. Your friendship, support and assistance was so meaningful and precious to me. I was very lucky to share the same research group with you. You will always be in my heart.

"... was worth every second for 7 years"

TABLE OF CONTENTS

LIST OF TABLES ..... **x**

LIST OF FIGURES ..... **xi**

1. INTRODUCTION **1**

2. RELATED WORK **4**
- 2.1. Bot Detection Systems 4
- 2.2. Twitter Account Behaviours 5
- 2.3. Data Compression 6
- 2.4. Deep Learning 7
  - 2.4.1. Variational Autoencoders 8
    - 2.4.1.1. Loss Function 9
    - 2.4.1.2. Reparametrization Trick 10

3. METHODOLOGY **11**
- 3.1. Data Compression Techniques 11
  - 3.1.1. Text Compression 11
  - 3.1.2. Principal Component Analysis 12
  - 3.1.3. t-Distributed Stochastic Neighbor Embedding 13
  - 3.1.4. Autoencoders 13
  - 3.1.5. Variational Autoencoders 14
- 3.2. Markov Chains 15

4. DATASET AND PREPROCESSING **16**
- 4.1. Profile Feature Extraction 18
- 4.2. Encoding Behavioral Features 20

5. RESULTS **24**
- 5.1. Behavioral Complexities 24
  - 5.1.1. Dataset Inspections 27

5.1.2. Temporal Behavioral Changes 28
5.1.3. Sequence Predictions 29
5.2. Profile Complexities 31
5.2.1. Implementation Details 32
5.2.2. Latent Space Analysis 32
5.2.3. Comparison of Different Techniques 34
5.2.4. Complexity Comparison 35
5.2.5. Optimal Compressions 37

6. CASE STUDY 38
6.1. 2017 GERMAN FEDERAL ELECTIONS 38

7. CONCLUSION AND DISCUSSION 41

8. FUTURE WORK 44

BIBLIOGRAPHY 46

LIST OF TABLES

Table 4.1. **Dataset information.** Number of human and bot accounts and annotation method for every dataset are given (Sayyadiharikandeh, Varol, Yang, Flammini & Menczer, 2020) 17

Table 4.2. **Features are used for profile complexity analysis.** Metadata features comes from API stream and calculated features use them to create new ones (Yang, Varol, Hui & Menczer, 2020) 19

Table 4.3. **Inter-tweet time segments**. We took common logarithm of corresponding times in seconds to scale times to integers to use them in our behavioral level system 21

LIST OF FIGURES

Figure 2.1. **Multilayer Artificial Neural Network Architecture**(Kutzkov, 2022) 8
Figure 2.2. **Example of KL Divergence**. More the blue curve overlaps with the green one, less KL divergence (Zafar, Tzanidou, Burton, Patel & Araujo, Zafar et al.) 10
Figure 2.3. **Illustration of reparametrization trick of VAEs** (Paul, 2020) 10

Figure 3.1. **Architecture of Vanilla Autoencoder** (Rocca, 2021) 14
Figure 3.2. **Architecture of a Variational Autoencoder with loss function** (Rocca, 2021) 14
Figure 3.3. **Example of Transition Matrix** 15
Figure 3.4. **Example of State System** 15

Figure 4.1. **Before normalization and standardization of profile features distribution** 20
Figure 4.2. **After normalization and standardization of profile features distribution** 20
Figure 4.3. **Illustration of original *Digital DNA* method on a Twitter account timeline** (Cresci, Petrocchi, Spognardi & Tognazzi, 2019) 21
Figure 4.4. **Example illustration of our level system from four tweets of an account timeline.** 22
Figure 4.5. **Second example illustration of our level system from five tweets of an account timeline.** 23

Figure 5.1. **Density Plots of `level_0` label-by-label for datasets** 25
Figure 5.2. **Density Plots of `level_1` label-by-label for datasets** 25
Figure 5.3. **Density Plots of `level_2` label-by-label for datasets** 25
Figure 5.4. **Behavioral complexity summary.** For all datasets, compression means and standard deviations are given for humans and bots 26

Figure 5.5. **Original and compressed sequence lengths of `cresci-17`.** Density plots also given for humans and bots at the left top figure 27
Figure 5.6. **Original sequence and compressed sequence lengths of *caverlee* dataset.** Density plots also given for humans and bots at the left top figure 27
Figure 5.7. **Temporal behavioral changes of a human account from *cresci-17* dataset.** `level_1` sequences given and high and low compressed chunks are highlighted 29
Figure 5.8. **Temporal behavioral changes of a socialspam bot account from *cresci-17* dataset.** `level_1` sequences given and high and low compressed chunks are highlighted 29
Figure 5.9. **Next tweet prediction accuracy of next tweet per dataset** 31
Figure 5.10. ***cresci-17* dataset with original embedding, latent space 3 and 15 embeddings** 33
Figure 5.11. ***stock* dataset with original embedding, latent space 3 and 15 embeddings** 33
Figure 5.12. ***caverlee* dataset with original embedding, latent space 3 and 15 embeddings** 34
Figure 5.13. **PCA, Autoencoder and Variational Autoencoder representations of 2 dimensional space for *cresci-17* dataset** 34
Figure 5.14. **PCA, Autoencoder and Variational Autoencoder representations of 15 dimensional space for *cresci-17* dataset** 35
Figure 5.15. **Prediction Accuracy of *cresci-17* dataset on different latent dimensions** 36
Figure 5.16. **Prediction Accuracy of *stock* dataset on different latent dimensions** 36
Figure 5.17. **Best compressible dimensions for different datasets** 37

Figure 6.1. **Distribution of tweet counts** 39
Figure 6.2. **Behavioral complexities of accounts tweeted about 2017 German Federal Elections** 39

1. INTRODUCTION

In recent years, digital communication and interaction have reached enormous levels through the development of technology and social media platforms. Everyone with an internet connection can connect to these platforms and share their ideas with other people they do not know. Although it sounds magical, such power is sometimes used for bad and harmful intentions like every other technological development. Some people and organizations have used automated accounts for various purposes in social media platforms. Specifically, in the Twitter platform, the number of such automated accounts, called bot accounts as a technical term, have increased rapidly as Twitter has become more popular and reachable for the entire world. As they are not controlled by humans and their nature relies on automation, the intention of such accounts would reveal by their behaviours. Automation with innocent intentions are usually easily recognizable as their purpose is lessening human work such as tweeting about recurring events or daily weather forecasts. On the other hand, some bot intentions may have bad and harmful influences over humans and they generally disguise as human and not self identified themselves and their aim is to spread misinformation (Shao, Ciampaglia, Varol, Yang, Flammini & Menczer, 2018). They cause artificial popularity, election manipulation, negative effects on financial and stock markets, promotion of a purpose with spam comments, or fake video impressions. To prevent such negative effects, identifying and analyzing them is an important task (Ferrara, Varol, Davis, Menczer & Flammini, 2016). Rather than identifying an account as a bot or human, this research aims to answer following research questions by using pre-labelled Twitter bot and human accounts datasets:

- Can we define a complexity measure for Twitter users?
- Do human and bot accounts are distinguishable in terms of complexity?
- Does behavioral and profile information complexities are consistent with each other for same account classes?
- How complex are the behaviours of Twitter accounts?

- Is there a difference between human and bot accounts' activity predictability?
- Does Twitter accounts behavioral complexity change during an important social event?
- How much we can compress Twitter account profile information with optimal information loss?
- Does lower dimensional representations of human and bot accounts are differentiable?
- How different is the performances of different compression methods?

It is essential to analyze Twitter accounts to understand the nature of human and bots and to reveal the differences between them. For such purposes, we decided to use compression techniques to observe and compare how much of information we can compress from different labelled accounts. By the compression level, we defined complexity measures and use them as evaluation metrics for this research. To answer questions above, we used publicly available bot datasets that are gathered from Twitter's official API and used in many different researches. Our complexity framework has two major branches which are behavioral and profile complexities. We used different approaches for them, for behavioral complexities, we encoded account activities into a string sequence and applied a text compression algorithm to assess complexity for human and bot accounts. To this end, we inspired from *Digital DNA* method (Cresci, Pietro, Petrocchi, Spognardi & Tesconi, 2016), and we introduced a level system for behavioral sequences by adding more information about their activities gradually, for each level. For instance at bottom level, we encoded only tweet type for each tweet an account posted. We mapped four different letter for tweet types. `T` for normal tweet, `R` for retweet, `A` for answer tweet and `Q` for quote tweets. In the second level, we added inter-tweet seconds in logarithmic scale for consecutive tweets and for the last level, entities used along with previous levels' information for each level. We detected four different entities that we believe are important about characteristics of a tweet and which are, hashtag number, user mention number, has url or not and has media included or not. One character per distinct information used in all levels. We compared the complexity results for different levels.
On the other hand, we extracted profile features and fed them into Variational Autoencoders (Kingma & Welling, 2014) which is a deep generative neural network algorithm to analyze them into latent representations by forcing the input data to keep important features in lower dimensions. Our aim is to compare different datasets' complexities, by finding optimal lower dimension with minimum information loss. We applied grid search for dimensions from minimum of 2 to $numberoffeatures - 1$.

During the search process, we also analyzed latent space distributions of human and bot accounts whether we can observe distinct clusters between them or not as we only kept important features in the lower dimensions.
Lastly, we conducted a case study to see the effect of our framework. We found a dataset consists of user information and tweets posted by the users about 2017 German Federal elections. Tweets ranged until nearly 4 months before the election and goes until the next day of election. Our aim is to find if the Twitter account behavior complexities change during an important social event or not. To achieve this, we analyzed and compared the complexities of tweets posted before the election months and during the election month. The details of our framework and results we obtained are discussed in the following sections.

2. RELATED WORK

In this chapter, we reviewed previous studies that are relevant with our research and gave a concise summary.

2.1 Bot Detection Systems

The need for detection of fully automated accounts on Twitter and other social media platforms is an important task to analyze since their significant influences over society like; spreading misinformation, political manipulation and giving false impression of an information, are emerged rapidly in recent years (Ferrara et al., 2016). There are several different approaches for bot detection. An early bot detection method used honeypot accounts to capture bots by luring them with posting nonsense contents that any rational human account do not interested in over a 7 month period (Lee, Eoff & Caverlee, 2011). More recent methods filtered account activities to find peculiar users that are actively tweeting about political contents (Howard & Kollanyi, 2016). Moreover, most popular and state-of-art bot detection system to date is a tool called *BotOrNot* when it is first released, which is a classifier relies on more than 1,000 metadata and informative features of Twitter accounts (Davis, Varol, Ferrara, Flammini & Menczer, 2016). Nowadays, it is known as *Botometer* and trained with 1,150 features in six different categories which are user-based, friends, network, temporal, content and language and sentiment features (Davis et al., 2016; Varol, Ferrara, Davis, Menczer & Flammini, 2017; Yang, Ferrara & Menczer, 2022). All different feature classes carry important features and characteristics that can define an account for activity, social connectivity and informative levels. Features are extracted from the datasets that are collected via the API to train *Botometer* classifier. Bot accounts detected via previous methods and verified human accounts used.

However, developing such complex classifier is challenging due to dynamic natures of bot accounts (Yang et al., 2020). In order to build an effective and efficient bot detection classifier, *Botometer* authors tried to solve two issues of bot detection systems. Firstly, there is a problem for scalability as for each account, too much information is needed for the classifier and high computational costs occurs which causes efficiency issue. Second issue is generalizability, which emerges as different kinds of bots may present in train and test data since bot datasets may not be scalable and may having limited amount of accounts to train the classifier. This cause poor and unreliable performances for detecting bots (Echeverrï!a, De Cristofaro, Kourtellis, Leontiadis, Stringhini & Zhou, 2018; Yang et al., 2020). To address such issues, only a few of user profile information used to train *Botometer* classifier that are easily obtainable from raw metadata. Such features are selected in a way that effectively distinguish accounts (Ferrara, Wang, Varol, Flammini & Galstyan, 2016). The proposed solution solves both issues, now the features as scaled up so that even limited computational powers are able to train classifiers. At the same time, it allows to extract many more account from Twitter API so it diversifies bot accounts and the possibility of imbalanced train for different bots decreases and having more reliable classifier. No doubt that, there is a trade-off between using complex but costly features and simpler but efficient ones but results show that scalable and generalizable data selection is an important application to having effective and efficient classifiers (Yang et al., 2020). For our study, we used the same features which are chosen by the *Botometer* authors that proved the effectiveness and they can be found in Table 4.2.

2.2 Twitter Account Behaviours

Researchers found out that significant changes happens to Twitter users' behaviours by exposure to misinformation (Wang, Han, Lehman, Lv & Mishra, 2021). Spreading the misinformation on social media is one of the main jobs of bot accounts that needed to be prevented (Ferrara et al., 2016). Such behavioral changes, may resulted with bots to evolve and adopt to current detection systems which is a problem for them (Cresci et al., 2019). To understand the nature and behaviours of bot accounts, an analytical framework called as *Digital DNA* is proposed recently (Cresci et al., 2016). The method characterizes a Twitter user's timeline as a sequence string

like human DNA sequences. Human DNA sequences are strands consisted of four different nucleotides. Similar approach for *Digital DNA*, Twitter accounts' timeline encoded as series of choronological actions to a sequence string by using possible three letters which are `B = {A,C,T}`. A is assigned to every *tweet*, C to every *reply* and T to every *retweet* (Cresci et al., 2019). Changed behaviors of same users, can be modelled by this approach and can be observed to analyze as it is flexible and easy to implement (Cresci et al., 2016).

2.3 Data Compression

Data compression is a process of reducing the amount of data for storage or transmission of any given information (Hemmendinger, 2013). Data is encoded into smaller bits rather than its original bit size; by searching patterns, repetitions or equivalent representation into smaller bit sizes. It is a common technique used to represent information efficiently in terms of storage aspect. Compression can be performed by programs that are utilized to apply a formula or algorithm to reduce the size, depending on the type of data or information are dealt with (Crocetti & Sliwa, 2017). Data Compression has two types of major branches, lossless and lossy compression. In the lossless compression, it is enabled to reconstruct the original data from the compressed representation without losing any information or bits. On the other hand, lossy compression permanently loses some information in compression process to have higher compression rates in comparison with lossless compression and it is impossible to reconstruct the original data from lossy compressed representation. However, it does not lose essential information from the original data (Fitriya, Purboyo & Prasasti, 2017).

2.4 Deep Learning

Deep learning is one of the subbranches of machine learning which is based on artificial neural networks. Deep learning emerged over classical machine learning techniques as the technology, computational powers and memory amounts advance (Watson, Cooper, Palacio, Moran & Poshyvanyk, 2022). They are faster, powerful and more reliable as the amount of data to process increases than the traditional machine learning methods (Brownlee, 2020). Similar with the common machine learning algorithms, deep learning techniques are applied with representation learning which can be supervised, semi-supervised and unsupervised. Representation learning is a set of methods that the network learns representations for feature extraction. This method allows network to automatically discover the needed features representations and use them to accomplish the task that network designed to perform In supervised learning, labelled data is used to learn and predict the specified task. When the data has no labels included, the learning task is considered as unsupervised learning. Sometimes the small portion of data has labels and large amount has no labels, combination of both during the learning process called as semi-supervised learning (Ali, 2021).

General structure of Deep Learning networks are composed of multiple layers which are input layers, hidden layers and output layers. The number of layers depend on the architecture of the network. In each layer, feature representations are obtained by combining non-linear transformation of the lower level representations into higher level representations. Such transformations increases the level of abstraction of the learned representations and at the end very complex feature representations can be learned by applying them. The learned complex features are dependent on the specified task. For instance for classification problems, the higher layers of transformed features are the ones that are crucial to distinguish the class which the instance is belonging to and not includes irrelevant features into the learning process. The most powerful part of deep learning is that the layers of learned features are not designed by humans, the network learns and optimizes through the transformations by only processing the given input data (LeCun, Bengio & Hinton, 2015). Figure 2.1 shows an example deep learning architecture used in natural language processing field with having input, output and 3 sets of hidden layers.

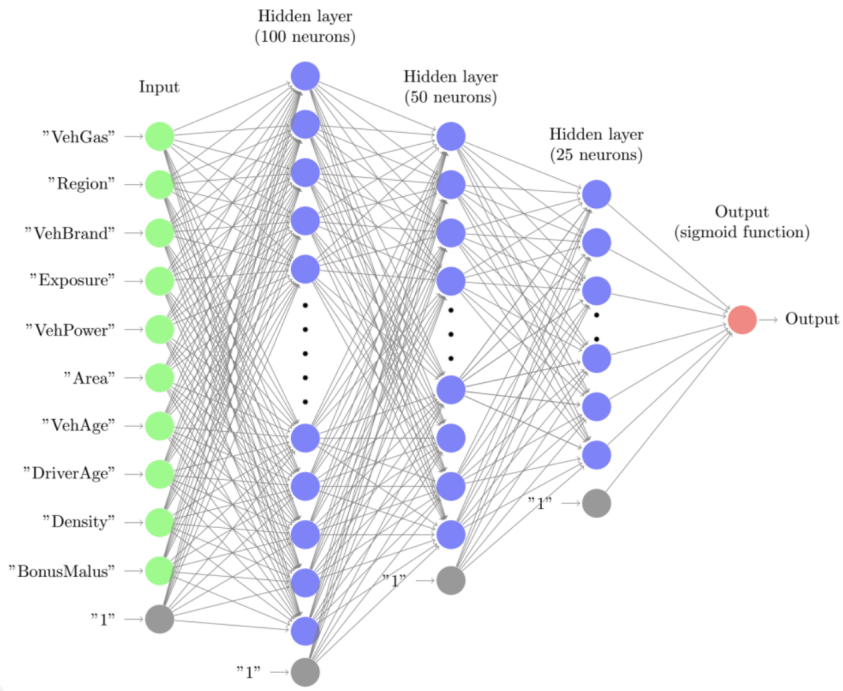

Figure 2.1 **Multilayer Artificial Neural Network Architecture**(Kutzkov, 2022)

Neural Networks use gradient descent algorithm during the training process. The complexity of deep learning algorithms require non-linearity therefore it is needed to use backpropagation with gradient descent algorithm to optimize deep learning networks. Backpropagation stands for "backward propagation of errors" and it fine-tunes the error rate of network weights in each training epoch. While training a deep learning model, for each iteration, gradient descent algorithm calculates new network weights and loss function calculates new losses for each epoch. Backpropagation feeds the amount of loss backwards ,meaning that from output to input, and by that procedure, the gradient descent tries to decrease the new network errors in the next iteration. With sufficient training epochs, backpropagation optimizes the network by calculating the gradient of the specified loss functions with respect to total error rate of the network. Optimization process is minimizing the total error so that increasing the reliability and generalizability of the networks (Al-Masri, 2019).

2.4.1 Variational Autoencoders

Variational Autoencoders (VAEs) are deep learning based generative models which allow to represent high dimensional input data into a low dimensional latent space that are learned with an unsupervised manner (Girin, Leglaive, Bie, Diard, Hue-

ber & Alameda-Pineda, 2021). VAEs are composed of two deep neural networks, encoder and decoder networks, connected to each other and have latent space in between them. VAEs assume the input data comes from a probability distribution, generally Gaussian, and tries to find the parameters of the distribution by using feature attribute values during the encoder part and then take samples from the new probability distribution to get the latent space representation. The parameters ***mean*** and ***sigma*** from the input distribution are tried to predicted by the network. This process makes latent space continuous, easy to interpolate and analyze, and keeps the important features of the data. Decoder network reconstructs the latent space embeddings into its original dimension. At the end of the encoding and decoding process, unseen data is generated and this is the most powerful side of VAEs (Kingma & Welling, 2019).

2.4.1.1 Loss Function

VAEs use a hybrid loss function for optimizing the network which is the combination of both reconstruction error and KL divergence. Reconstruction error is mean residual between input data and reconstructed data. Usually, mean squared error or mean absolute error are used for this term. On the other hand, KL divergence calculates the difference between newly calculated Gaussian distribution and a standard Gaussian distribution with mean 0 and standard deviation 1. The optimization process of a VAE network works as; minimizing the reconstruction loss so that generated samples would be likely the original input while approximating the calculated Gaussian distribution to a standard Gaussian. Therefore, KL divergence acts like a regularization term and keeps the balance between generating similar samples while keeping the important features that separates input clusters (Öngün, 2020). Considering the absence of error term in the loss function, KL divergence forces the network to take samples from a standardized Gaussian distribution as much as possible and this creates a problem as generating very similar samples so that the reconstruction error goes high and reconstruction the input data would be meaningless so hidden clusters inside the input features would be vanished. An example of KL divergence can be found at Figure 2.2. Blue curve represents the approximated Gaussian distribution while green curve represents a standard Gaussian.

Figure 2.2 **Example of KL Divergence**. More the blue curve overlaps with the green one, less KL divergence (Zafar, Tzanidou, Burton, Patel & Araujo, Zafar et al.)

2.4.1.2 Reparametrization Trick

Artificial Neural Networks are trained by backpropagation method. It stands for backward propagation of errors which uses gradient descent algorithm with the error function to optimize neural network weights in each training epochs as it discussed more detailed in deep learning section previously. However, in the case of VEAs, backpropagation can not be applied to the process of taking samples from a Gaussian distribution parameters so there should be a change in order to use backpropagation in the training process of VAEs. Therefore, *epsilon* a random value from Gaussian distribution, is used during the sampling process of VAEs. The sampling equation is $z = mean + sigma * epsilon$ where *epsilon* is a random variable sampled from standard Gaussian distribution, have a very little value to not change the sampling equation and since it is a random variable, backpropagation is now applied to the network. This process called **reparametrization trick** of VAEs (Paul, 2020). An illustration of reparametrization trick of VAEs can be found on Figure 2.3.

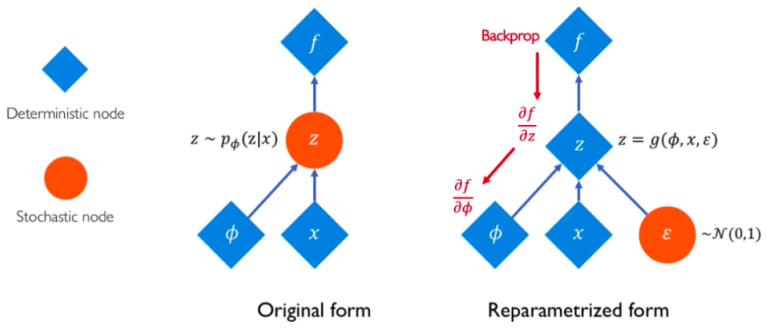

Figure 2.3 **Illustration of reparametrization trick of VAEs** (Paul, 2020)

3. METHODOLOGY

In this chapter, we give brief information about the methods that we used in our framework. Obtained results with the methods, will be discussed in the results chapter.

3.1 Data Compression Techniques

From Twitter accounts perspective, our goal is to evaluate compression rates of account behaviours and to analyze compressed representations of account profile information. We had used different data compression approaches for our analyses on Twitter accounts.

3.1.1 Text Compression

Text Compression is a technique for changing the representation of given set of characters. Although there are different approaches for text compression, semantic information of the text is not important in our research since we do not concentrate on the content of the tweets, rather our spotlight is on the informative characteristics of tweets and such encoded texts do not require to have a meaning so we focused a simple text compression technique to only focus on the character sequences over the text, specifically a Python package called Compress (Hu, 2020) which is a compression API and we used zlib (Jean-loup Gailly, 2022) framework for compression algorithm. Zlib conducts compression operation in bit-wise so that it converts original string to a compressed string in bits and decompresses from bits to string. It searches patterns and repetitions into the original string to compress.

If a character or sequence of characters repeats itself consecutively, the algorithm counts the occurrence number of each pattern and rather keeping the repetitions, keeps the occurrence number bit-wise. It allows the algorithm to compress the given string. We can define a compression ratio metric to analyze the complexity of each account as follows:

- Original Text Size (t_0): Length of the input text
- Compressed Text Size(t_C): Length of the compressed text
- Offset (O): Length offset of zlib compression algorithm used. We found out it as 20 and subtracted it from the compression calculation to be consistent for all accounts.
- Compression ratio (CR): Complexity metric for account behaviours.

$$CR = (t_0 - t_C)/t_0 - O \tag{3.1}$$

We used Zlib text compressions in our research to find behavioral complexities of Twitter accounts. Each account's activities and behaviors encoded into sequence strings and applied text compression to get compression ratios and find account complexities. We interpreted the compression ratio as; higher the compression ratio, lower the account complexity in our framework.

3.1.2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique by applying linear transformation to input data (Wold, Esbensen & Geladi, 1987). The aim of PCA is changing the basis of data by calculating new uncorrelated variables into a given lower dimension so that optimizes explained variance of data and information loss. PCA can tell for which reduced dimension keeps how much of the information from the input data. This procedure can be applicable for our complexity analysis by setting an explained variance threshold for lower dimensions and comparing different bot datasets complexity. However, PCA only works for linear inputs and non-linearity of the data might cause a problem. Other dimensionality reduction techniques that account nonlinearity might be more confident for Bot

datasets. We applied PCA to get lower dimensional representations of account profile information and used it as a benchmark for account profile complexities and compared the results with other methods.

3.1.3 t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding(t-SNE) is a nonlinear dimensionality reduction technique which calculates a similarity measure between data instances for both original dimension and reduced dimensions than tries to optimizes the similarities and locates similar data instances closer (Van der Maaten & Hinton, 2008). t-SNE is used in our framework for visualization purposes to get 2-dimensional embeddings of higher dimensional data.

3.1.4 Autoencoders

Autoencoders are artificial neural networks used for learning dense representation of input data for unsupervised learning tasks (Baldi, 2012). Common usage areas for Autoencoders are data compression via dimensionality reduction, facial recognition, anomaly detection and image denoising. It consists of two parts which are encoder and decoder. Encoder network maps the input data into a lower dimensional space which often called as latent space. The layers of encoder network try to keep the important information of the original input and embed them into latent space to efficiently learn the important or hidden features of the data. Decoder network is the opposite of encoder which takes the latent embeddings and reconstructs them to the original dimension. During the encoding and decoding procedure, some information from the original input is lost and the whole network tries to minimize that loss while learning the important features of data. Figure 3.1 shows the architecture of a vanilla Autoencoder. Similarly with PCA, we used Autoencoders as a benchmark for profile complexities and compared Autoencoder compressions with other dimensionality reduction techniques for profile information in our analysis.

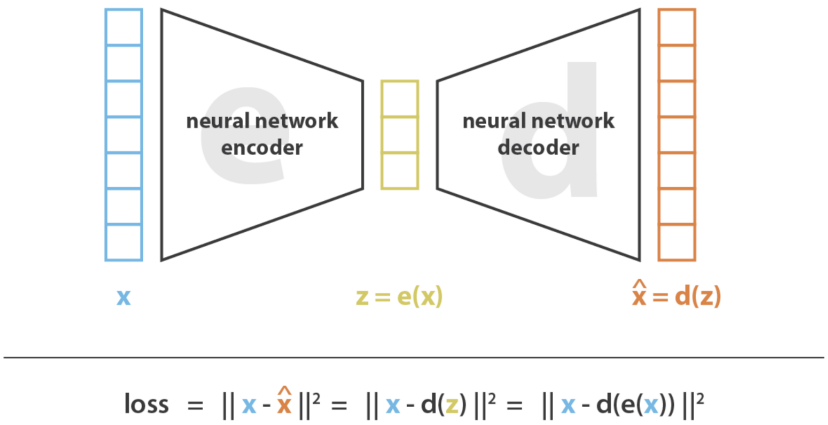

Figure 3.1 **Architecture of Vanilla Autoencoder** (Rocca, 2021)

3.1.5 Variational Autoencoders

Variational autoencoders are generative version of autoencoders and have similar architecture (McCaffrey, 2018). They also have encoder and decoder networks for encoding the input into lower dimensional latent space and reconstructing back to the original dimension. However, VAEs can generate unseen data and it makes them more flexible and powerful than standard autoencoders. The details about VAEs are discussed in the related work section earlier. Figure 3.2 shows a standard VAE architecture. Our main complexity analysis for account profile information is done by VAEs. We extracted profile features of each account from user metadata of Twitter API and fed them into VAEs to get latent space representations. Lower dimensional latent representations are compared for human and bot accounts in different dimensions to find out if there is a difference between human and bot accounts in their hidden spaces.

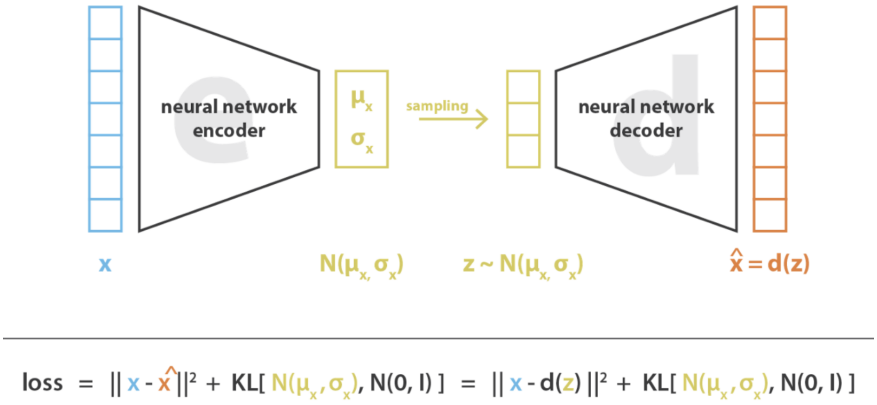

Figure 3.2 **Architecture of a Variational Autoencoder with loss function** (Rocca, 2021)

3.2 Markov Chains

Markov Chains are mathematical stochastic system that describes making transitions from one state to another using probabilistic approaches (Maltby, Khim, Lin, Williams, Hernandez, Jackson & Pakornrat, 2022).It has important properties as making the transition to the next state only depends on the previous state which means that it is independent of history of states. Moreover, future states are always fixed and transition probabilities are defined by certain rules that only depends on the system. There different kinds of Markov chain systems such as discrete time, continuous time, finite state etc. In our framework, we can define a state system which the states are the tweet types and transition probabilities can be calculated as for each tweet, counting the previous tweet occurrences divided by the total number of occurrence of that type. Such system is proper for discrete time Markov Chains. The states are: S = {A,T,Q,R}. For instance for a transition matrix as such in Figure 3.3 leads to a state system for the tweet types like Figure 3.4. Markov Chains are used to support behavioral complexity differences between human and bot accounts. We used account encoded behavioral sequence strings as an input and get predictions for each accounts' next activity and find system accuracy to check whether there is a difference between human and bot account predictability.

	A	T	Q	R
A	0.285714	0.000000	0.285714	0.428571
T	0.066667	0.100000	0.433333	0.400000
Q	0.051724	0.120690	0.275862	0.551724
R	0.011905	0.238095	0.321429	0.428571

Figure 3.3 **Example of Transition Matrix**

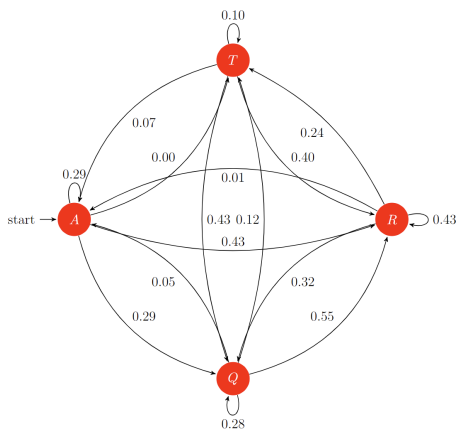

Figure 3.4 **Example of State System**

4. DATASET AND PREPROCESSING

We used publicly available datasets from bot repository[1] that are collected and labelled during different events and scientific purposes which are total of 14 distinct bot datasets. Table 4.1 contains number of human and bot accounts for each dataset along with annotation method for labelling the dataset.

Most of the datasets have human annotation method which means they are manually labelled by human efforts. For example ***astroturf*** dataset contains hyper-active political bots that are detected thorough systematically deleting content and follow trains which is basically group of accounts posting content addressing to follow each other (Sayyadiharikandeh et al., 2020). ***botwiki*** dataset have self-identified bot accounts from bot archive [2] (Yang et al., 2020). Filtering by verified account feature from the API stream, ***verified*** dataset created and consists of only human accounts (Yang et al., 2020). ***gilani-17*** dataset contains bot and human accounts that are clustered into four different categories depending on the number of followers which are *celebrity status*, *very popular*, *mid-level recognition* and *lower popularity.* They took samples from the categorized accounts and manually labelled the data based on key information (Gilani, Farahbakhsh, Tyson, Wang & Crowcroft, 2017). Another human annotated dataset is ***midterm-2018*** which accounts are filtered during 2018 U.S. elections (Yang, Hui & Menczer, 2019; Yang et al., 2020). Twitter user *@josh_emerson* shared political bot accounts in dataset ***josh-political*** (Yang, Varol, Davis, Ferrara, Flammini & Menczer, 2019). ***kevin-feedback*** have accounts from feedbacks given to *Botometer* (Yang et al., 2019). In ***rtbust*** dataset, manually labelled from Italian retweets for a 12 days span in 2018 (Mazza, Cresci, Avvenuti, Quattrociocchi & Tesconi, 2019). For ***varol-icwsm*** dataset, accounts are manually labelled depending on the *Botometer* score (Varol et al., 2017). There are other annotation methods also are used rather than human annotations. For example, as being the oldest dataset present in this framework, ***caverlee*** dataset used honeypot accounts to lure bots and human accounts that are manually verified

[1] `botometer.org/bot-repository`

[2] `botwiki.org`

by authors (Lee et al., 2011). ***gregory-purchased*** dataset contains fake followers that are purchased from companies (Yang et al., 2019). Andy Patel [3] shared scam site sharing bot accounts in ***pronbots*** dataset (Yang et al., 2019). ***stock*** dataset have bot accounts that are isolated to find same timelined tweets for five months for selected cashtags (Cresci, Lillo, Regoli, Tardelli & Tesconi, 2018,1). Lastly, ***cresci-17*** dataset have more than one bot type along with human labels which are social spambots, tradition spambots and fake followers. Social spambots classified by accounts repeating normal accounts to promote a hashtag or a content coordinatively. Tradition spambots are bot accounts that tweet always same content. Fake followers the ones that paid other accounts to follow (Cresci, Pietro, Petrocchi, Spognardi & Tesconi, 2017a,1).

Dataset Name	Annot. Method	#Bots	#Humans	Ref.
`astroturf`	Human Ann.	505	0	(Sayyadiharikandeh et al., 2020)
`botwiki`	Human Ann.	697	0	(Yang et al., 2020)
`caverlee`	Honeypot + verified	15,483	14,833	(Lee et al., 2011)
`cresci-17`	Various Methods	7,049	2,764	(Cresci et al., 2017)
`gilani-17`	Human Ann.	1,090	1,393	(Gilani et al., 2017)
`gregory-purchased`	Fake Followers	1,087	0	(Yang et al., 2019)
`josh-political`	Human Ann.	62	0	(Yang et al., 2019)
`kevin-feedback`	Human Ann.	380	138	(Yang et al., 2019)
`midterm-2018`	Human Ann.	0	7459	(Yang et al., 2020)
`pronbots`	Spam Bots	17,882	0	(Yang et al., 2019)
`rtbust`	Human Ann.	352	340	(Mazza et al., 2019)
`stock`	Sign of Coordination	7,101	6,174	(Cresci et al., 2018b;2019)
`varol-icwsm`	Human Ann.	733	1,495	(Varol et al. 2017)
`verified`	Human Ann.	0	1,986	(Yang et al., 2020)
`Total`		**52421**	**36582**	

Table 4.1 **Dataset information.** Number of human and bot accounts and annotation method for every dataset are given (Sayyadiharikandeh et al., 2020)

[3] `github.com/r0zetta/pronbot2`

4.1 Profile Feature Extraction

Feature engineering and selection is one of essential parts of bot detection systems (Varol, Davis, Menczer & Flammini, 2018). The datasets utilized in this research contain a user object for each account which has many informative profile features, along with account's most recent 200 tweets and its properties. The account features in the datasets had been used to develop *Botometer* classifier which is a popular bot detection system, over 1,000 features used for each account. (Davis et al., 2016; Varol et al., 2017). As the complexity analysis needed to be applicable for each Twitter account, regardless of being automated or natural, the features to be used are chosen in a way that solves the scalability and generalization problem of bot detection systems as mentioned in related work section. We extracted user metadata information, calculated metadata features' rates with respect to account age in seconds to capture how fast the account's activity occurs (Yang et al., 2020), calculated hashtag numbers used in the user profile and selected binary features regarding whether the user uses the default profile settings or have a customized profile. Table 4.2 shows the type and description of features that we used in our framework.

Metadata Feature Name	Type	Explanation
favourites_count	count	Total number of favourites
statuses_count	count	Total number of tweets of the account
friends_count	count	Number of followings of the account
followers_count	count	Number of followers of the account
listed_count	count	Number of public lists that the user is a member of
default_profile	binary	Whether the user has changed the theme or background or not
default_profile_image	binary	Whether the user has changed the default profile image
Calculated Feature Name	**Type**	**Explanation**
screen_name_length	count	Length of the screen name
num_digits_in_screen_name	count	Number of digits in the screen name
name_length	count	Length of the user name
num_digits_in_name	count	Number of digits in the user name
description_length	count	Length of the profile description
num_#_in_description	count	Number of hashtags in the profile description
num_#_in_name	count	Number of hashtags in the user name
age	numeric	Time in seconds between account creation and its last tweet
tweet_freq	numeric	statuses_count / age
followers_growth_rate	numeric	followers_count / age
friends_growth_rate	numeric	friends_count / age
favourites_growth_rate	numeric	favourites_count / age
listed_growth_rate	numeric	listed_count / age
followers_friends_ratio	numeric	followers_count / friends_count

Table 4.2 **Features are used for profile complexity analysis.** Metadata features comes from API stream and calculated features use them to create new ones (Yang et al., 2020)

After collection and calculation of features, we applied log-transformation to each of them to normalize and standardize them to prepare for our analyses since numeric and calculated features are distributed with different ranges and scales. This process did not change binary and count features much but still we applied for all features to have more confidence on the features. An example distribution for ***caverlee*** dataset for before and after log-transformation can be found in Figure 4.1 and Figure 4.2

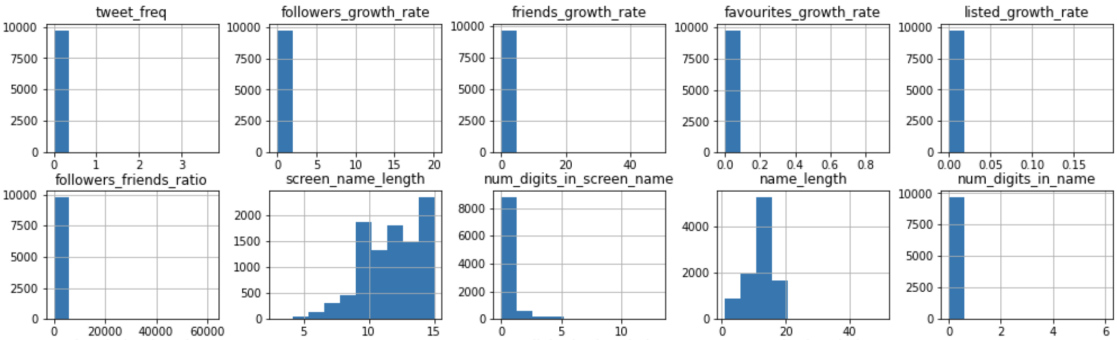

Figure 4.1 **Before normalization and standardization of profile features distribution**

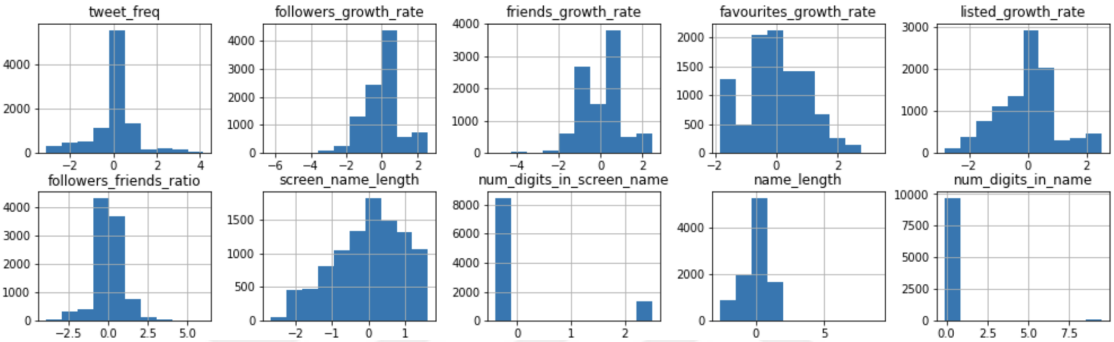

Figure 4.2 **After normalization and standardization of profile features distribution**

4.2 Encoding Behavioral Features

Other than the profile information, we encoded each account's activity into a sequence string by a method called *Digital DNA* extraction to analyze behavioral complexities of Twitter users. The original method is to characterize each tweet posted by an account chronologically, with a capital letter, from the alphabet $B = \{A, C, T\}$ where A stands for each tweet, C is for each reply and T for each retweet to create a behavioral sequence string like $s = ...TTATCT$ (Cresci et al., 2019). Figure 4.3 shows the application of Digital DNA method on an account timeline.

We extended the *Digital DNA* method by adding extra information into the sequence strings for each tweet and introduced a level system to analyze account behavior complexities. The levels are:

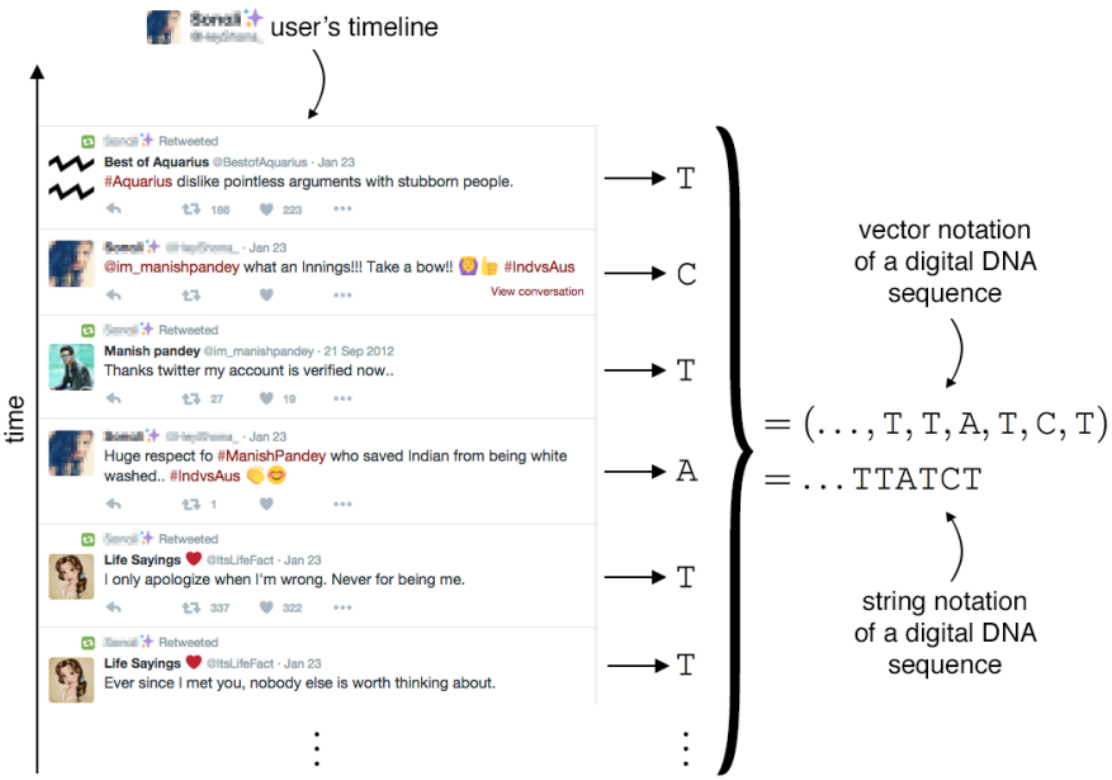

Figure 4.3 **Illustration of original *Digital DNA* method on a Twitter account timeline** (Cresci et al., 2019)

Time Segments	Time in Seconds	Logarithmic Scale
`second`	`1`	0
`minute`	`60`	1
`hour`	`60*60`	3
`day`	`60*60*24`	4
`week`	`60*60*24*7`	5
`month`	`60*60*24*7*4`	6
`year`	`60*60*24*7*4*12`	7

Table 4.3 **Inter-tweet time segments**. We took common logarithm of corresponding times in seconds to scale times to integers to use them in our behavioral level system

- `level_0`: Single character for each tweet, depending on the tweet type. The alphabet for `level_0` is $B = \{A, T, Q, R\}$, A stands for reply, T for normal tweet, Q for quoted tweet and R is for retweet.
- `level_1`: Two characters for each tweet, addition to the level_0 with a digit between 0-7, depending on the time passed between each tweet with its following tweet on a logarithmic scale. Times are calculated in seconds but transformed into a 1-digit by taking its common logarithm. Table 4.3 describes the

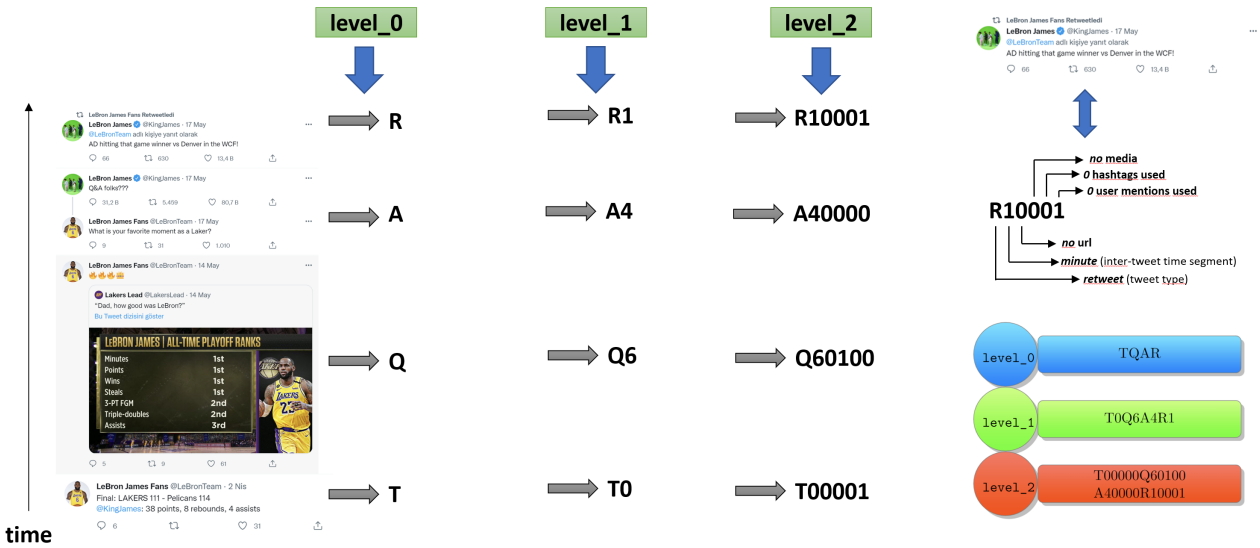

Figure 4.4 **Example illustration of our level system from four tweets of an account timeline.**

digits and corresponded time intervals between consecutive tweets. Since last tweets has no tweet to follow, we calculated the most common time interval for that account and added it to last tweets.

- `level_2`: In addition to `level_0` and `level_1`, `level_2` contains entity information. Entities that reflect the characteristics of tweets, are selected to add in this level, are as follows:v
  - `url`: Binary information, checks the tweet contains a URL or not.
  - `media`: Binary information, checks the tweet contains a media or not.
  - `hashtags`: Count information, calculates how many hashtags used in the tweet.
  - `user mentions`: Count information, calculates how many users mentioned in the tweet.

  If a tweet has more than 9 `hashtags` or `mentions count`, we interpreted them as 9 to be consistent with the one character for each information idea. Figure 4.4 and Figure 4.5 shows the illustration of behavioral encoding of our level system from account timelines.

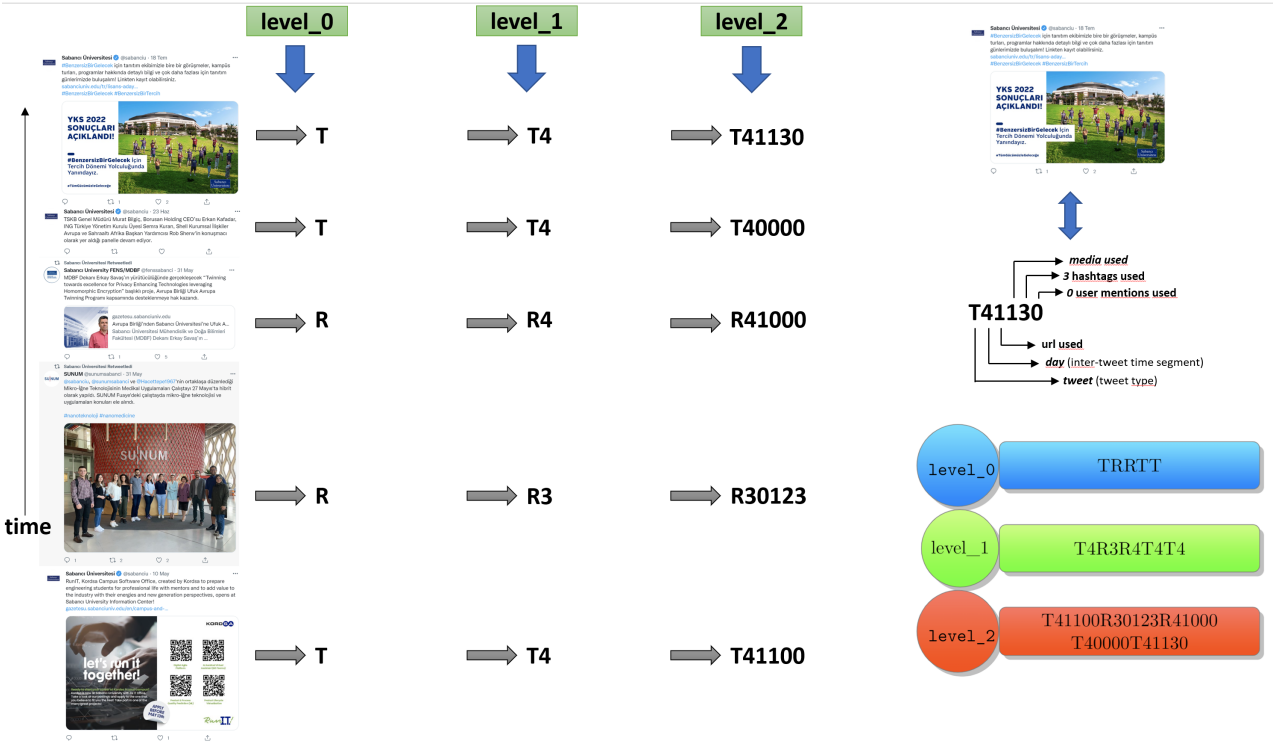

Figure 4.5 **Second example illustration of our level system from five tweets of an account timeline.**

5. RESULTS

In this part, we will present our findings by the methodology we explained in the previous part. We can divide the results into two parts and start with behavioral complexities, give behavior predictions and end with profile complexities.

5.1 Behavioral Complexities

As we encoded user activities as string sequences with our level system and defined a compression ratio by applying a text compression algorithm into the behavioral sequences, we can analyze complexities of each account and can come up with complexity comparisons between each dataset. Figure 5.1, Figure 5.2 and Figure 5.3 shows the compression densities for each dataset and for each label for all three levels. By only looking at the density plots side-by-side, it is visible that the bot accounts are more compressible comparing to the humans for almost each dataset and compressibility changes for different levels are seen.

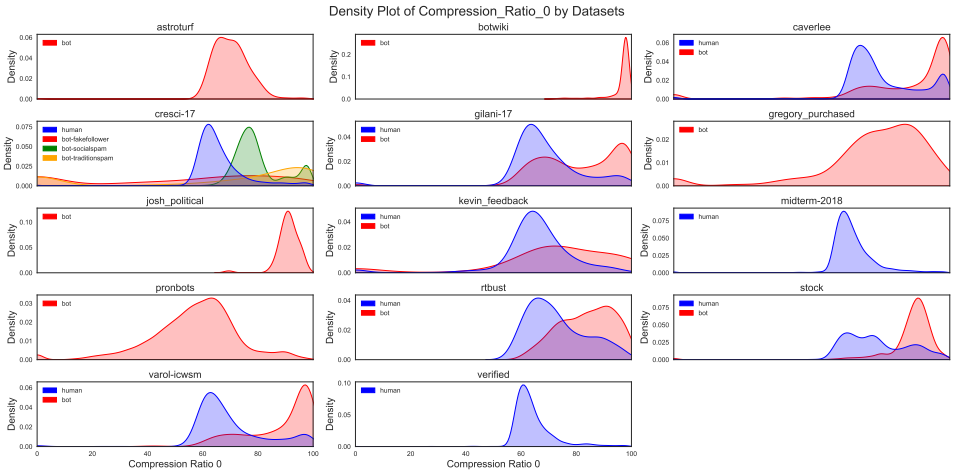

Figure 5.1 **Density Plots of `level_0` label-by-label for datasets**

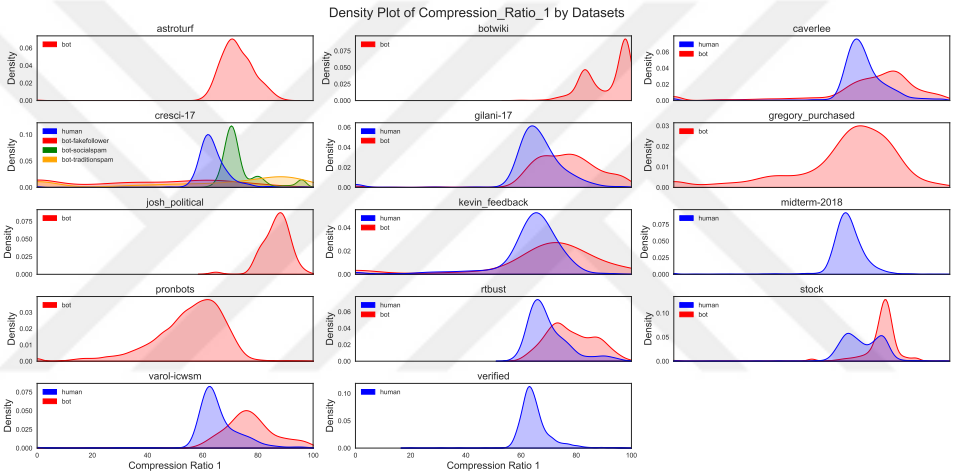

Figure 5.2 **Density Plots of `level_1` label-by-label for datasets**

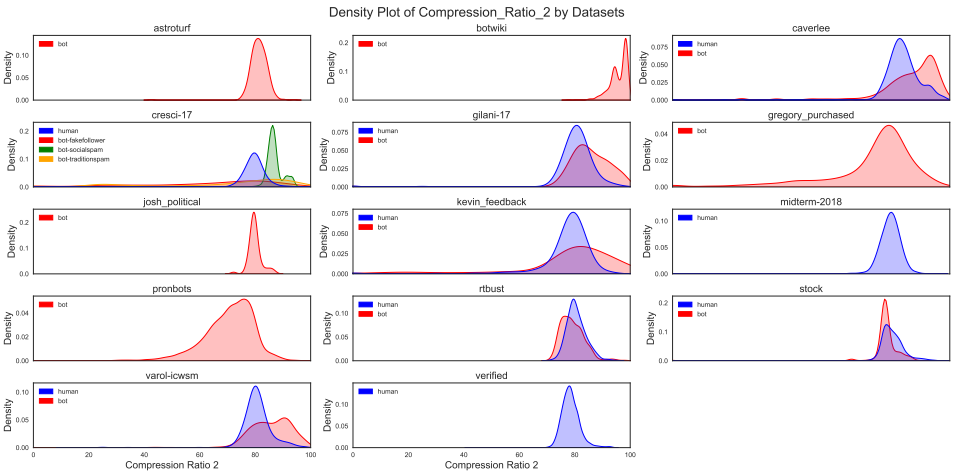

Figure 5.3 **Density Plots of `level_2` label-by-label for datasets**

Furthermore, we can inspect the dataset complexities at denser perspective by looking at Figure 5.4. This figure summarizes our framework over account activities. It contains mean compression ratios at x-axis and standard deviations at y-axis so that we can compare the complexities of human and bot accounts, each dataset and different levels of behavioral information separately in the same figure. We can clearly see that human accounts are more consistent within itself than the bot accounts for both dataset-to-dataset and level-to-level. Mean and standard deviation intervals are lower than bots and lowest mean compression ratio is around 60% while highest is around 80%. Also, for standard deviations, the intervals are around 5 to 15. Another aspect is that comparing different levels, level_2 have highest compression ratios as we include entity information into level_2 and for plain tweets level_2 have higher repetitions comparing to levels 0 and 1. Moreover, for most of the datasets, level_0 has higher compression ratios than level_1 as expected since inter-tweet times forces the sequences having lower compression ratios if the account did not send tweets for same and fixed time intervals. On the other hand, bot accounts are mover divergent within itself. We can see some bot datasets can reach near 100% compression mean while some only have around 50% and standard deviations also have a similar divergence. Likewise for different levels, we can not observe a consistency over different levels like humans since it different levels do not have similar compression range in the figure for bot accounts. By that results, we can conclude that the behavioral complexity of bot accounts are not consistent through dataset and it depend on the dataset and its annotation method. However humans are more likely to be dataset independent as expected.

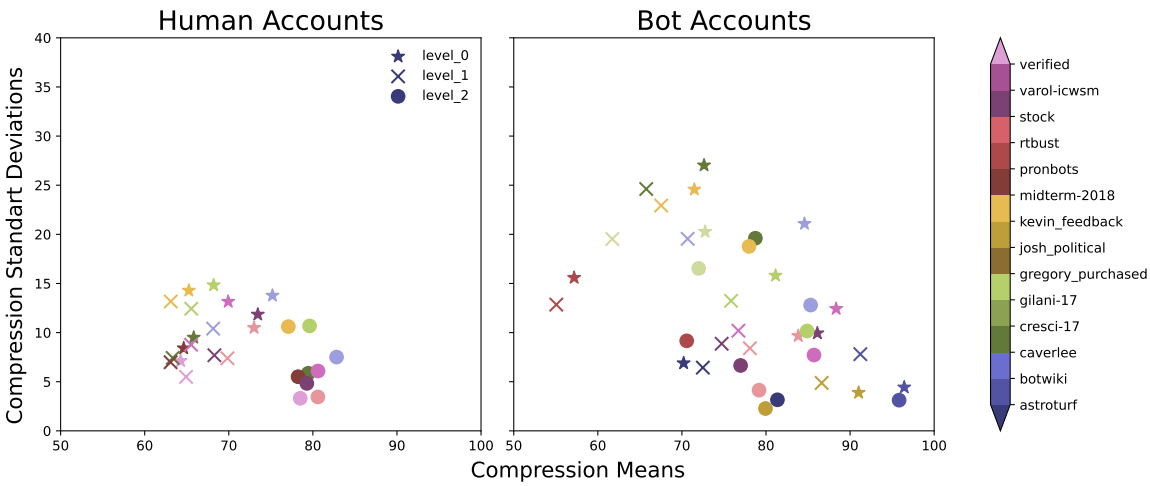

Figure 5.4 **Behavioral complexity summary.** For all datasets, compression means and standard deviations are given for humans and bots

5.1.1 Dataset Inspections

Apart from general perspective, we can also inspect the complexities of each dataset for which having both bot and human labels to compare. Inspecting ***cresci-17*** dataset is important as it contains different bot types as labels along with humans. Figure 5.5 shows the sequence lengths and compressed sequence lengths which are the original input length for tweets and length in bits for compressed representations, respectively. It contains the lengths for each user with all three levels in ***cresci-17*** dataset. By looking at the scatter diagram of each user, it is visible that compressed sequence lengths and sequence lengths are not colliding at the diagonal dashed $x = y$ line and the area between dots and the line shows us how much of the accounts behaviors compressed. Also from the fşgure, the compression lengths of humans are higher than other bot types, specifically than socialspam and traditionspam bots, and socialspam bots lengths are higher than traditionspam bots as it suggested in the density plot at the left corner of the figures.

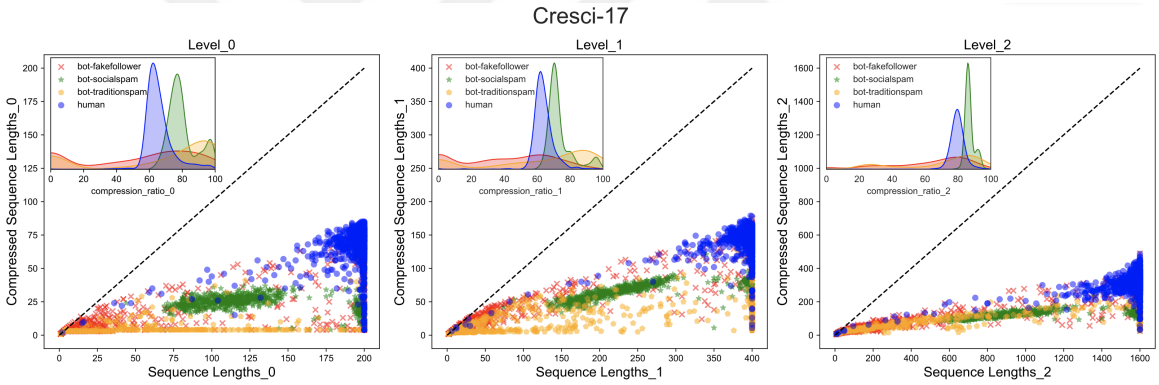

Figure 5.5 **Original and compressed sequence lengths of `cresci-17`.** Density plots also given for humans and bots at the left top figure

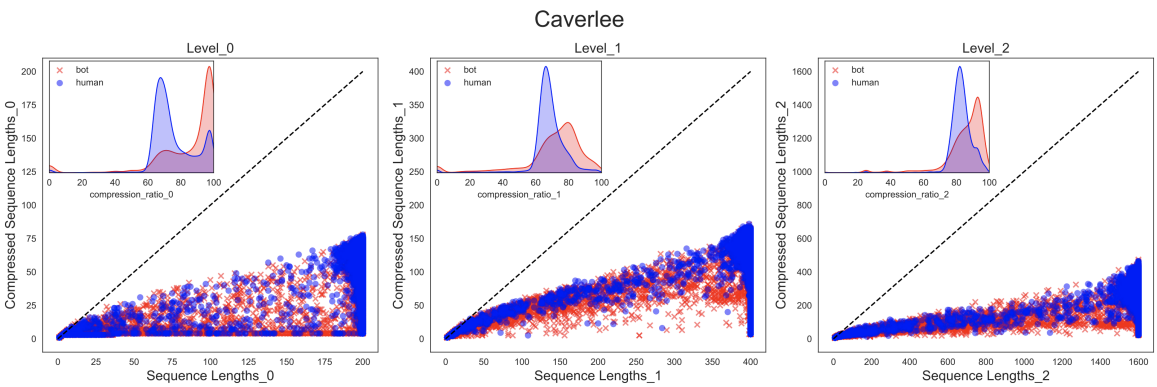

Figure 5.6 **Original sequence and compressed sequence lengths of *caverlee* dataset.** Density plots also given for humans and bots at the left top figure

We can also inspect single bot type and human labelled datasets. For instance Figure 5.6 shows sequence and compressed lengths of **caverlee** dataset. From both figures, we can conclude that `level_0` and `level_1` gives important hints about the differentiability of humans and bots in terms of account complexity since unique labels are visible in the graph in different positions. However, for `level_2`, which have the entity information along with tweet types and inter-tweet times, compression rates increases for both human and bots and complexity difference decreases as it is suggested from individual plots as well as summary figure. The cause of it may depend on the entities selected or the entity positions and it is discussed more in the conclusion and discussion section.

5.1.2 Temporal Behavioral Changes

Most of the datasets that we used are collected through long time spans. It gives us a chance to analyze temporal behavioral changes of accounts. We take little chunks of behavioral sequences and slide the chunk window with a given parameter. We selected the chunk size as 20 and slide size as 10 but parameter values are subject to change for better analyses. We selected two accounts, one human and one bot. Their behavioral changes are observable when we apply compression the little behavioral chunks over that accounts whole behavioral sequence and plot compression ratios with respect to mean date of chunks. Each chunk contains 20 tweet so we took the mean date of 20 tweets dates to plot with compression ratios. Figure 5.7 and Figure 5.8 shows example of temporal changes of accounts. Observation of such changes show that account behaviors change with respect to time for both humans and bots. Bot changes are important to observe since bot behaviour changes might cause bot detection systems to fail to capture them. Similarly, human behavior changes might reveal occurence of an important event during sharp increase or decrease in the compression ratios. We conducted more comprehensive analysis about this in Case Study section.

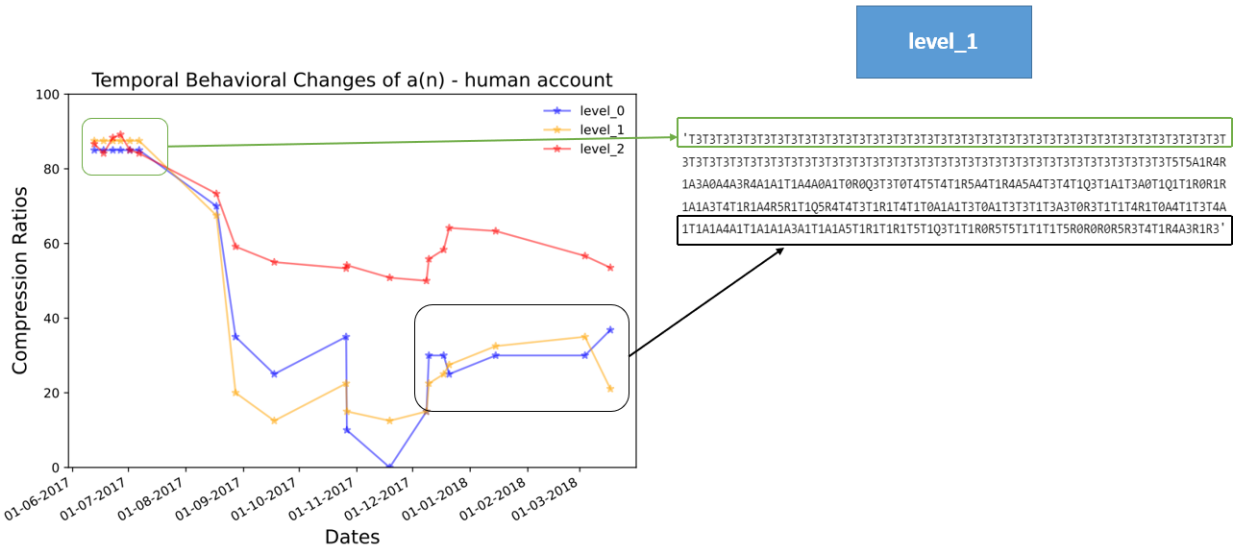

Figure 5.7 **Temporal behavioral changes of a human account from *cresci-17* dataset.** `level_1` sequences given and high and low compressed chunks are highlighted

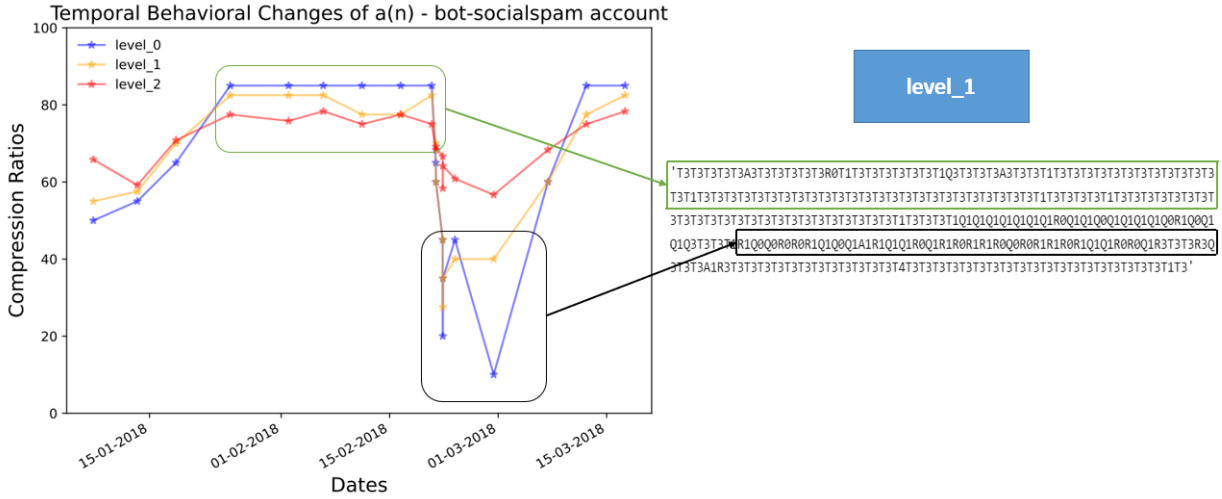

Figure 5.8 **Temporal behavioral changes of a socialspam bot account from *cresci-17* dataset.** `level_1` sequences given and high and low compressed chunks are highlighted

5.1.3 Sequence Predictions

Discrete-Time Markov Chains (DTMC) are proper tool for predicting the next variable of long sequences. We can use DTMC to predict the next tweet type that an account will post. In our level system, all 3 levels can be used to predict next

sequence information. For instance predicting only next tweet type, we can use level_0, to predict next tweet type along with inter-tweet time, level_1 can be used and predict entity information, level_2 can be used. However, for simplicity, we only used level_0 and predicted the next tweet type. In that setting we have 4 unique states and have 4x4 transition matrix and if we used level_1, we should have 28 different states since 4 different tweet types and 7 different time segments and 28x28 transition matrix. It is way bigger for level_2 because we add 4 new characters for each tweet, so we only focused on predicting the next tweet type. After setting the states, we should determine how to calculate transition probabilities. Markov property suggests that making a transition only depends on the previous state so, we can calculate transition probabilities by counting the next tweet type for each tweet and dividing it to total number of occurrences of the tweet type. We can extend the Markov property by looking previous 2 tweets for calculating the transition probabilities. We defined two parameters for sequence prediction and which are:

- `STATE_LEN`: number of previous tweets to consider for transitions
- `PRED_LEN`: number of next tweets to be predicted

Figure 5.9 shows the accuracy of predicting next tweet with using Markov properties for each dataset with `STATE_LEN` = 2 and `PRED_LEN` = 1 . The figure has labels for both human and bot accounts as well as it contains the biased prediction which only counts the tweet types and make a prediction based on the majority tweet type. For datasets which having both human and bot labels, prediction accuracy is always higher for bot accounts than humans. It shows that bot accounts are more predictable, less complex than humans as expected. Prediction results are consistent with the behavioral sequence compression results, for instance in `botwiki` dataset, bot accounts are self identifiable so they have very less complexity, we know it from previous figures, have the highest prediction accuracy. Another drawback is that except from a few cases, Markovian and biased accuracy are very close to each other for most of the datasets. It maybe suggest that, the next tweet that will be post by an account, may be independent from the previous tweets, it may depend on the tweeting habits of accounts. The results obtained by using different `STATE_LEN` and `PRED_LEN` can be found in the appendix section.

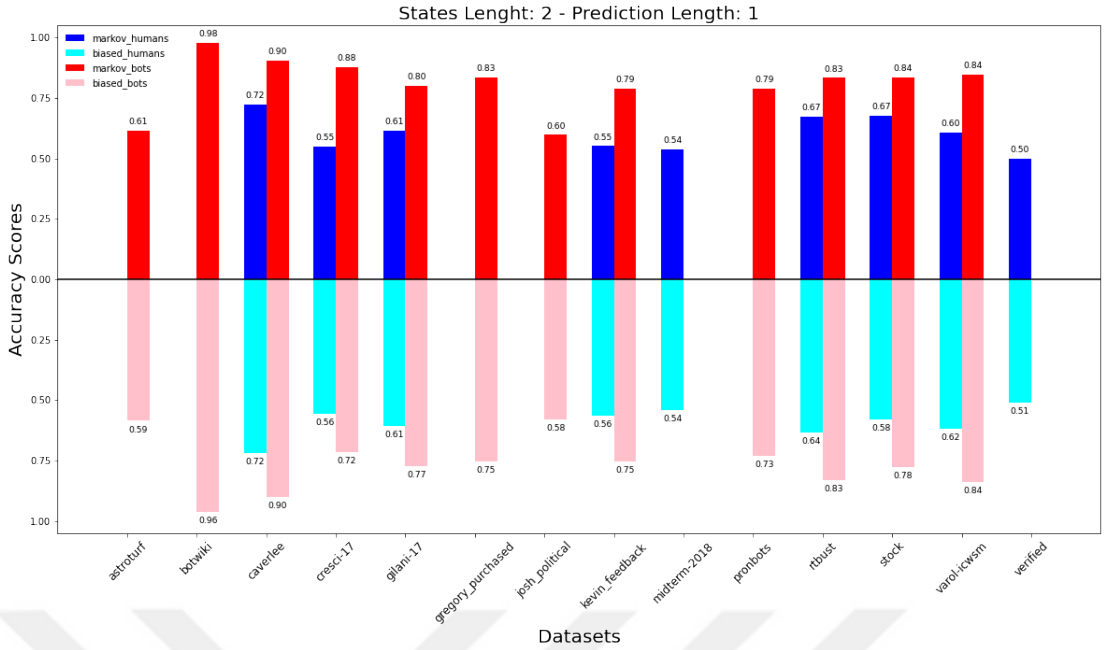

Figure 5.9 **Next tweet prediction accuracy of next tweet per dataset**

5.2 Profile Complexities

Apart from account behaviours, we also analyzed bot and human account's profile information to see that whether bots and humans are differentiable. It will allow us to compare different dataset complexities. Similar to the previous approach, we will use a compression method but with profile features of accounts. In Table 4.2, we listed profile features and we put them into VAE to reduce the dimensionality of our input data. As we discussed in the methodology section, VAEs keep important features into latent space with lower dimensions. Therefore, we defined a range of dimensions to compress, and compare different dimensions. We defined lower dimensions range as `range(2,21)` for which 2 is the lowest possible meaningful compressible range and 21 is the $numberoffeatures - 1$, in other words minimum compressible dimension. As VAEs apply lossy compression to the input data, we applied a grid search to the lower dimensions range to find the best compressible dimension with minimum information loss.

5.2.1 Implementation Details

We used *tensorflow* API (Abadi, Agarwal, Barham, Brevdo, Chen, Citro, Corrado, Davis, Dean, Devin, Ghemawat, Goodfellow, Harp, Irving, Isard, Jia, Jozefowicz, Kaiser, Kudlur, Levenberg, Mané, Monga, Moore, Murray, Olah, Schuster, Shlens, Steiner, Sutskever, Talwar, Tucker, Vanhoucke, Vasudevan, Viégas, Vinyals, Warden, Wattenberg, Wicke, Yu & Zheng, 2015) to train VAE network. The fully connected encoder and decoder networks have two hidden dense layers with sizes 64 and 32. Encoder network connected with the input data with shape of 22, number of profile features, to two hidden dense layers dimensions from 64 to 32 and output dense layer with shape as latent space dimension from the grid search. After the sampling process, the decoder network takes sampled input with latent space dimension, connected it to the hidden dense layers from 32 dimensional to 64 dimension, and as output, it connects to the original dimensional dense layer which is 22. For all layers, *relu* activation function is used. We trained all the datasets with batch size as 64 and for number of epochs, early stopping method is used and network stopped training when there is no improvement on the network loss. Similarly, network reduced learning rate when reaches to plateau by monitoring the loss.

5.2.2 Latent Space Analysis

VAE encodes given input data into a compressed lower dimension by forcing the input distribution to a normal distribution and taking samples from it. By this way, only the important features are kept and hidden insights of the input may reveal in the latent space. We analyzed different dimensional latent spaces of datasets, to see that if human and bot accounts are differentiable in the latent space. TSNE embeddings are used to produce 2D projections of higher dimensional latent spaces and original input. Figure 5.10 shows original dimension embeddings, latent space dimensions 3 and 15. Different classes in the dataset became more visible as the latent space dimension goes higher from 3 to 15. Furthermore, in the original TSNE embedding, the clusters are too messy and same labels are seperated, however in latent space dimension 15, it seems more concerted and clusters are visibly selectable. When we observe the latent embeddings of other datasets, Figure 5.11 Figure 5.12 shows original dimension embeddings, latent space dimensions 3 and 15 embeddings for ***stock*** and ***caverlee*** datasets. Original dimension embeddings, which is input data without applying VAE compressions, not showing any regularity between bot

and human classes but VAE embeddings seems more regular. Also, higher latent dimension 15 seems more regular than latent space 3. However, it is visibly understandable that VAEs performed better on ***cresci-17*** dataset than other two datasets. We believe that the cause for different latent space structures between datasets may be the annotation method for datasets. More detailed discussion can be found in the conclusion and discussion section.

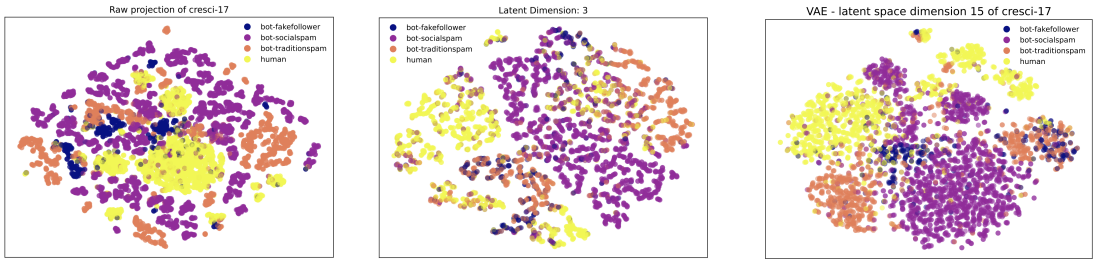

Figure 5.10 ***cresci-17*** **dataset with original embedding, latent space 3 and 15 embeddings**

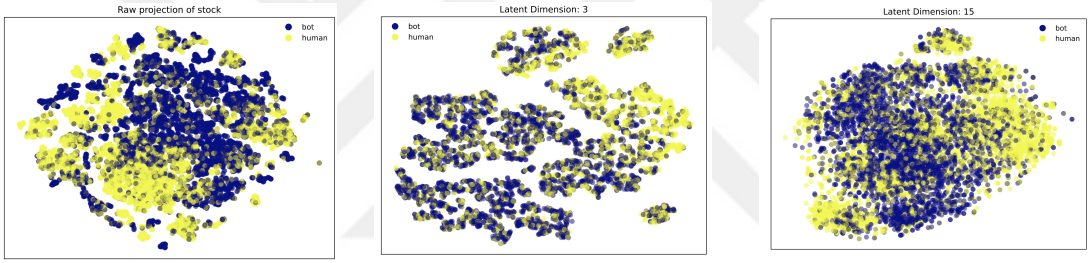

Figure 5.11 ***stock*** **dataset with original embedding, latent space 3 and 15 embeddings**

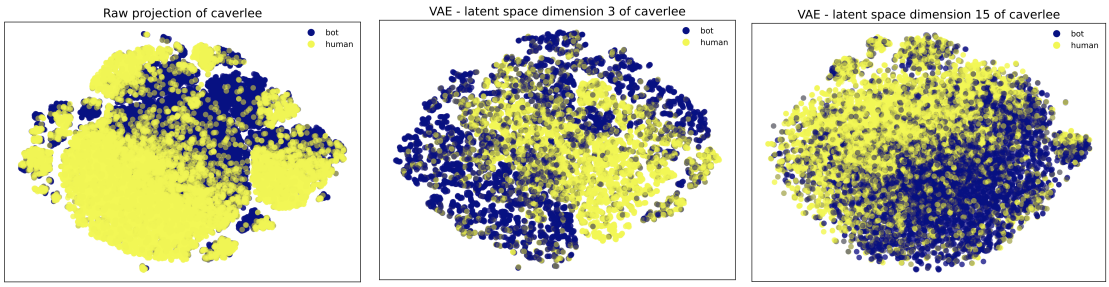

Figure 5.12 ***caverlee*** **dataset with original embedding, latent space 3 and 15 embeddings**

5.2.3 Comparison of Different Techniques

We can compare different dimensionality reduction techniques for profile features to see how well VAEs compress data into lower dimensional spaces. Figure 5.13 and Figure 5.14 shows the comparison of 2 and 15 dimensional representations of ***cresci-17*** dataset for PCA, Autoencoder and Variational Autoencoder. PCA and VAE have similar visible clusters in 2 dimensional representations of ***cresci-17*** dataset but AE fails to seperate labels. However, when we increase the compressed dimension, PCA acts similarly as AE and fails to seperate. Most regular and seperable cluster in dimension 15 is observed in VAE encodings. Therefore, VAEs are powerful tool to compress the input data into lower dimensions with keeping the most important features.

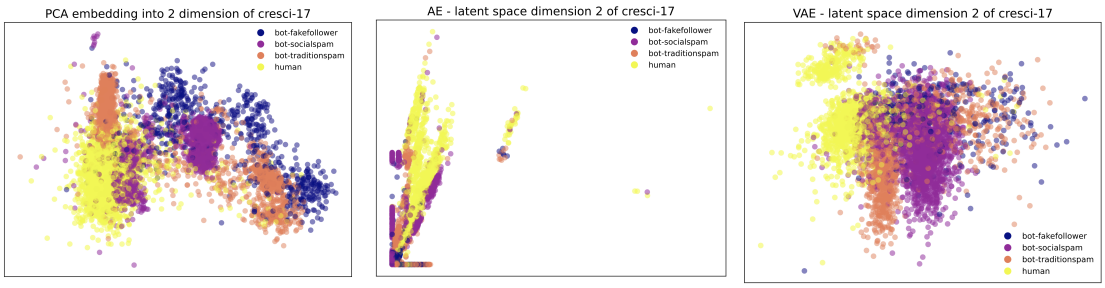

Figure 5.13 **PCA, Autoencoder and Variational Autoencoder representations of 2 dimensional space for** ***cresci-17*** **dataset**

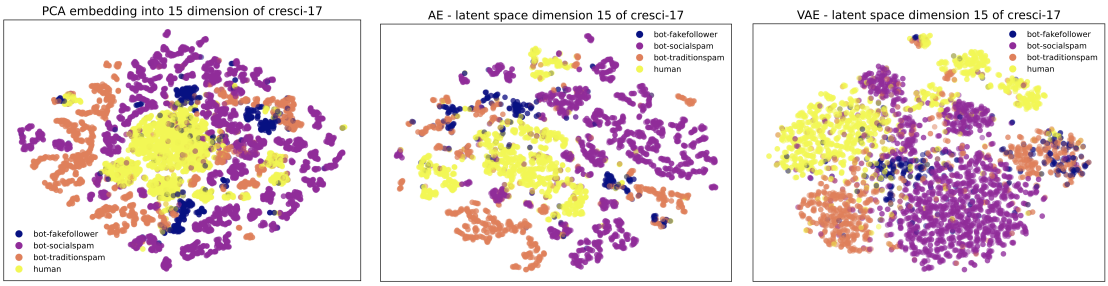

Figure 5.14 **PCA, Autoencoder and Variational Autoencoder representations of 15 dimensional space for *cresci-17* dataset**

5.2.4 Complexity Comparison

To validate the method that we used, we applied Random Forest Classifier (Ho, 1995; Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot & Duchesnay, 2011) with default parameters to both the original input data and lower dimensional compressed data for all lower dimensions to compare. We used repeated stratified K-fold cross validation technique with 10 fold and 10 repeats to validate the accuracy results. As VAE is a lossy compression technique, we should expect that we loose more information as the compressed dimension gets lower and the classification accuracy should be lower for compressed data in comparison with the original data. Figure 5.15 and Figure 5.16 shows the classification accuracy and accuracy change with respect to next latent space dimension for ***cresci-17*** and ***stock*** datasets. `RAW` corresponds to original input while other ticks on x-axis, shows the latent dimension of compressed data. As it can be seen from the figures, the classification accuracy goes higher from latent dimension 2 to some higher dimension gradually and become closer to the original input accuracy, but improvement nearly stops at some dimension. Therefore, we defined a heuristic complexity measure as determine a lower dimension for which the accuracy change of next consecutive 3 dimensions are under 1%. We designate such dimension as an optimal compressible dimension for the dataset. As the optimal dimension is lower, means the dataset has less complexity. We can use this measure to compare different dataset complexities.

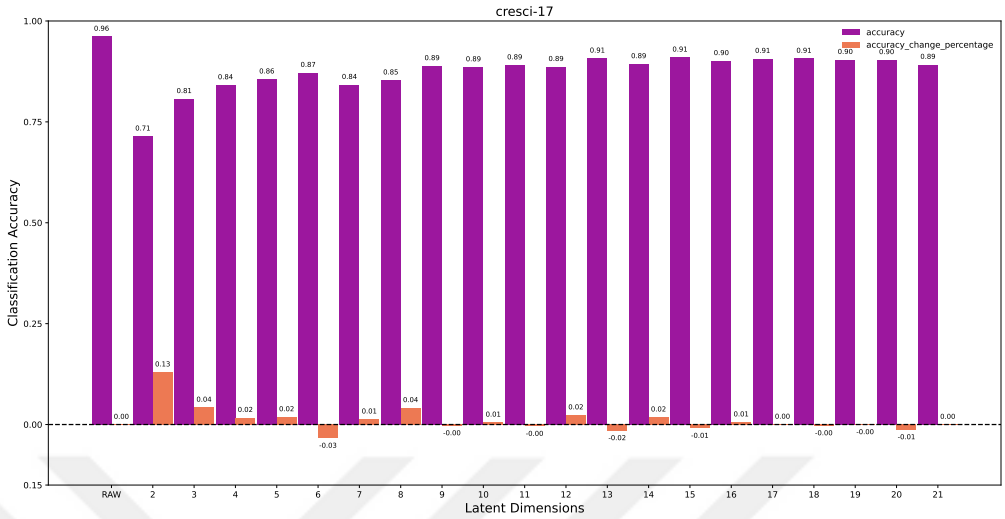

Figure 5.15 **Prediction Accuracy of *cresci-17* dataset on different latent dimensions**

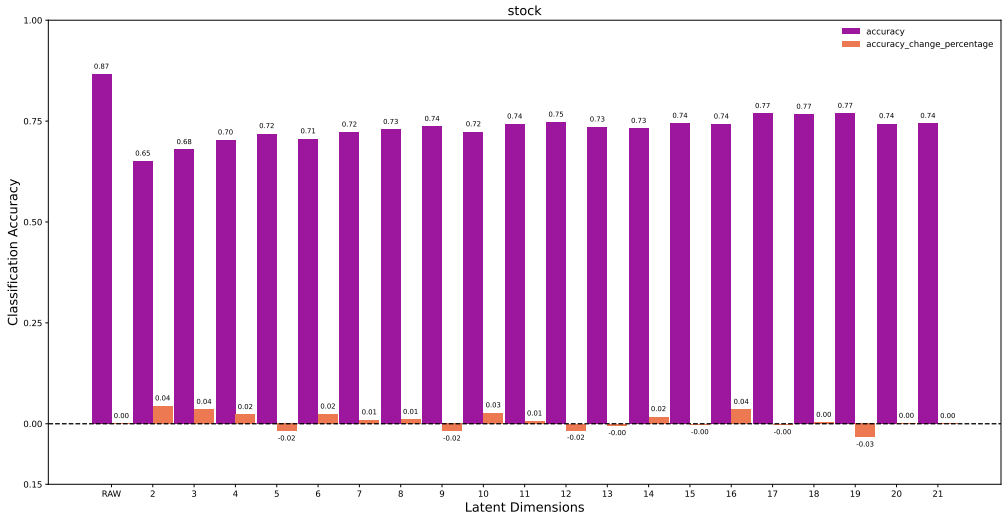

Figure 5.16 **Prediction Accuracy of *stock* dataset on different latent dimensions**

5.2.5 Optimal Compressions

We validated reduced dimension complexities by applying Random Forest Classifier. Regarding to the number of accounts contained and having both human and bot labels, we selected five proper datasets to find best compressible dimension which are ***caverlee***, ***cresci-17***, ***gilani-17***, ***stock*** and ***varol-icwsm***. To find the best dimension for the selected datasets, we developed a heuristic approach which keeps track of improvements on the classification accuracy of each dimension. We defined a parameter `epsilon` = 0.01 and when the classification accuracy increased, we calculated the difference between them and compared it with `epsilon`. If classification accuracy stops improving, meaning that accuracy change is lower than `epsilon` with the previous highest accuracy, we selected the dimension as the best compressible dimension in terms of minimum information loss with maxiumum compression. Figure 5.17 shows our heuristic approach results for finding best optimal dimension for selected five datasets.

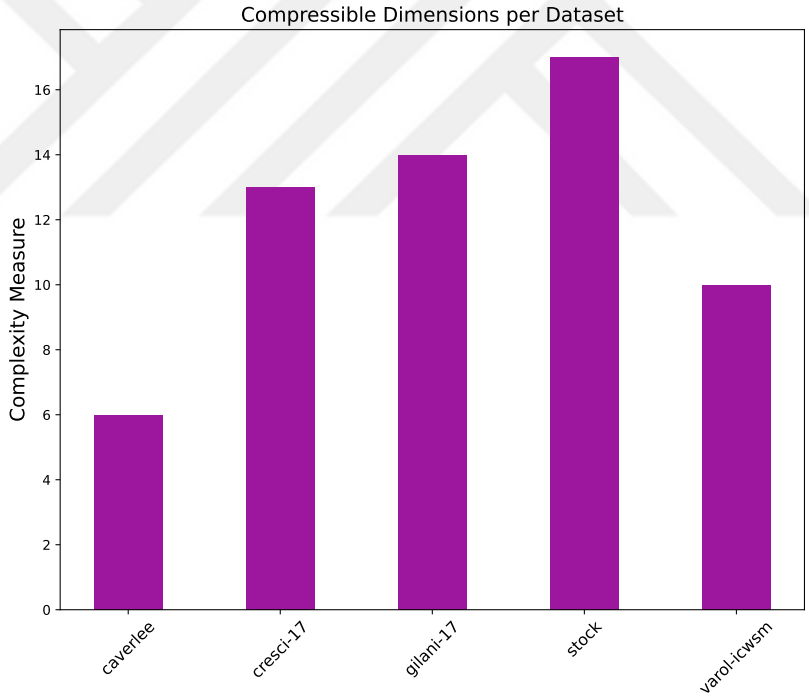

Figure 5.17 **Best compressible dimensions for different datasets**

The results show the dimensions that stop improvement on the accuracy. For instance, ***cresci-17*** dataset showed improvement until dimension 13 and improvement stops, so we found best lossy dimension. For ***caverlee*** dataset stopped improvement at dimension 6. So it shows ***cresci-17*** dataset complexity is higher than ***caverlee*** dataset.

6. CASE STUDY

In this section, we conducted a case study to show the application of our complexity framework on a different event and dataset than bots

6.1 2017 GERMAN FEDERAL ELECTIONS

This case study is about applying our behavioral complexity framework to a Twitter dataset that are collected before and during the Federal Elections of Germany that happened at 24 September 2017. The dataset comprises of Twitter interactions about 364 German politicians by more than 120.000 active Twitter users and more than 1.200.000 Tweets starting from May 29 till 25 September which is one day after the elections (Kratzke, 2017). Since we need some amount of tweets to create behavioral sequences to apply compression and not want to deal with outliers, we filtered the accounts that are having more than 20 tweets. Than we checked the tweet count distribution of the accounts and it can be found on Figure 6.1. By looking at the distribution, we chose to continue with users having number of tweets higher than 30 and lower than 130 based on the tweet count distributions.
After filtering, we left with 5,270 users having 30-130 tweets during a 4 month time span with total of 308,960 tweets. Noting that this dataset does not have any labels about the collected accounts, we use it see whether an important societal event like a federal election, changes the Twitter account's tweeting habits and behaviours or not. It is done by examining the behavioral complexities of accounts by comparing before and during the election month which is September 2017. We divided the dataset into before September tweets and after September tweets. We have total of 213,103 tweets before September and 95,857 after September. Since the dataset have no labels and our aim is not comparing human and bots, rather comparing pre and during event complexities, we only used behavioral sequence compression for this case study. Figure 6.2 shows the compression means and standard deviations of

before and after September tweets. It also contains the euclidean distances between same levels on before and after September. We can directly see that, compression means are decreasing and standard deviations are increasing in September which is the election month.

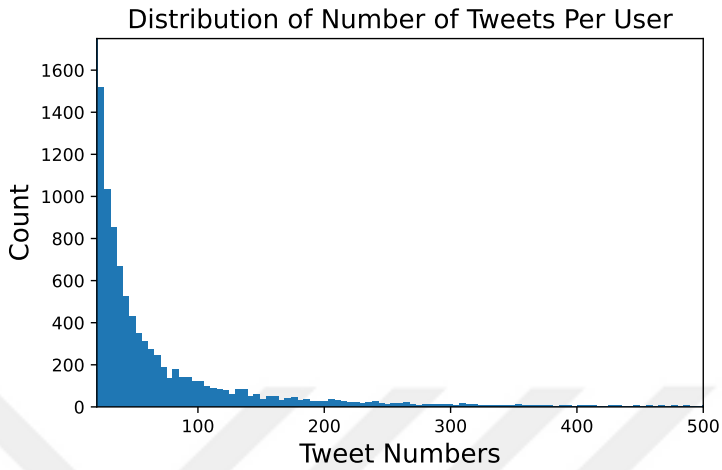

Figure 6.1 **Distribution of tweet counts**

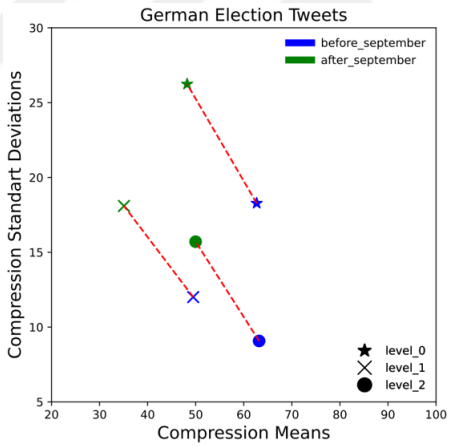

Figure 6.2 **Behavioral complexities of accounts tweeted about 2017 German Federal Elections**

Although level-by-level complexity differences are visible on the figure and tweet complexities are higher in the election month, the compressibility trend is similar for both time intervals. `level_0` has highest compression means and at the same time has highest standard deviation while `level_2` has smallest standard deviation and `level_1` has smallest compression means in both cases. Therefore, as complexity increases in the election month, tweeting habits of accounts are still consistent within each other. In both time intervals, entity information in `level_2`, are not used much and repeats itself to have highest compression means among other levels and inter-tweet times still hard to compress and increases the complexity of tweets.

Furthermore, `level_0` has the highest distance among the distances of same levels of different time intervals which actually explains what cause the difference between election month and preceding months. Since we saw the consistency between tweet characteristics in terms of inter-tweet times and entities, the cause for higher complexity in election month is the complexity on different tweet types posted. The natural cause of it may be the fact that Twitter users can post mixture of their subjective opinions and objective news in the election month as election results and other reflections of the election reveal in the election time comparing with months before the election.

This case study shows us that a major societal event like an election may affect tweet complexities. Such effect may cause a bias for the bot detection during such events since our study showed that humans have higher complexities over bots but in the case study, same users have different complexities in the presence of election.

7. CONCLUSION AND DISCUSSION

In this research, we tried to define different complexity measures for Twitter users. Our analyses have two major branches and which are account behavioral complexities and account profile information complexities. We analyzed Twitter accounts and defined complexity measures by compressing their activities and profile metadata information. Our hypothesis was to observe significant differences between bot and human accounts in terms of account and dataset complexity when we compare their compressed representations. We utilized two main methods to obtain the results. We extended so-called *Digital DNA* method and add more information from the characteristics and content of each tweet and introduced a level system for calculating the behavioral complexities of accounts. We also adopted Variational autoencoders to use it on Twitter account datasets, to analyze latent representations of bot and human accounts. For behavioral part of our analysis, the results we achieved were supportive to our prior beliefs, and we had saw the differences between human and bot account behaviors. Overall, for different datasets, human compressions distributed into a denser space while bot compressions have a wider range of space in terms of compression means and standard deviations. We expected to see humans are more complex than bots and the results proved our prior belief, human compressions are lower than bots in average. However, as there are different kinds of bot accounts for different datasets, the behaviors of bots are inconsistent than humans. We can observe bot behaviour differences from two aspects of our analysis; firstly, standard deviations are higher in bots and secondly, embedded behaviours in our level system proved it. Compression level ratios of different levels positioned very close for different datasets for human accounts but it is still inconsistent for bot accounts. Nevertheless, we managed to achieve highest compression means with bot accounts. At that point, we tried to predict the next behaviour for accounts by using Markov Chains by defining behavioral sequences as a state system. Markov predictions showed us that bot accounts are more predictable for all datasets than humans and it supports our belief of bot accounts are less complex than humans.

Digital DNA method is a fine and powerful tool for analyzing account activities. Its

power proven via our analysis, human and bot behavior differences were outlined by applying compression over sequences. However, our extended level system is still open for improvements. Zlib compression searches patterns over a string and in our level system, the position of information in different levels or even position of entities within `level_2` may cause differences of compression amounts. Furthermore, many other levels can be introduced as every tweet contains much more information that we demonstrated in this research. Semantic information may be one of the most important improvement to our system as we did not consider semantics of tweets. Even further, we introduced levels as cumulative way it also may cause biasses to the compressions working on single but different information per level also may improve our level system.
After the analyses done over bot and human account activity complexities on bot datasets, to validate our methodology, we conducted a case study on a dataset collected during 2017 German Federal Elections. It contains accounts that are actively tweeted about German politicians and elections starting from four months before the election until election date. Therefore, we applied our behavioral complexity framework to see whether account activities change through different time spans during a major societal event like election or not, and divided the dataset as tweets before the election month and on the election month. The results showed that the overall account complexities increased during the election month when comparing to the previous months. The behavioral levels that we defined are consistently changed in both time spans and complexity of all increased in the election month. As a result, our framework showed that, not only for bot datasets, the behavioral complexity framework can be applicable for all accounts and it is useful to analyze behavioral changes of accounts through time and existence of major societal events. Such changes may cause biases for bot detection systems.

Another aspect of our complexity analysis was profile information compressions by VAEs. One of our research questions was if human and bot accounts are differentiable in their lower dimensional representations or not. Our results showed that, VAEs yielded visually most regular latent spaces comparing with AEs and PCA. VAEs showed that, the regularity over latent space increases as the latent space dimension increases since information loss gets lower in higher dimensions. However, VAEs perform differently from dataset-to-dataset. ***cresci-17*** dataset has a regular and clustered latent space in dimension 15 but ***stock*** and ***caverlee*** dataset latent spaces are less regular and seperable when comparing with ***cresci-17*** dataset. The reasons for it should be investigated by conducting more analyses and experiments in the future but there might be a correlation between correctness of annotated labels with the visual clustering and regularization quality of latent

spaces. ***cresci-17*** dataset used various and more sophisticated annotation methods comparing with the other datasets and it might be the key reason for VAEs performed better over ***cresci-17*** dataset. Although, ***caverlee*** dataset used honeypot luring for bot annotation, manual verification used for human accounts and VAE performed poorly for distinguishing human and bot accounts.

Our other goal for profile complexity analysis is to quantify complexities of different datasets by applying a grid search procedure for latent space dimensions from 2 to number of $input features - 1$ to find a best compressible dimension. As objective function of VAEs are information loss, during the network training, we tried to find minimum loss with maximum compression. To validate our results, Random Forest Classifier used to evaluate different latent space dimensions of each dataset. The best compressible dimension is defined heuristically by a dimension which the accuracy stops improving afterwards. The improvement change tracked by an `epsilon` parameter which we used its value as 0.01. The approach is highly dependent and sensible to the value of `epsilon` parameter, however, we believed 0.01 is a fair threshold by considering the accuracy change values on Figure 5.15 and Figure 5.16 and we want the accuracy to improve so we selected it as 0.01. The results showed that ***stock*** dataset have the highest complexity while ***caverlee*** dataset has the least one.
Such results are open to discussion as it is an heuristic approach. From Dataset and Preprocessing section, we explained the annotation methods for each dataset. ***stock*** dataset has a different annotation method as isolating the bot accounts by the selected cashtags and that method yielded highest dataset complexity. Human annotated ***gilani-17*** and ***varol-icwsm*** datasets placed in the middle in terms of complexity among other datasets but it is not enough to make an inference about human annotation methods since ***cresci-17*** dataset used various bot detection methods but still has close complexity with human annotated datasets. Lastly, ***caverlee*** dataset has lowest complexity between other datasets and it is understandable as it is one of the very early bot datasets and long time passed since its annotations and caverlee bots can be considered as simplest bots because of its age. Another discussion point may be the number of samples contained in the datasets. ***caverlee***, ***stock*** and ***cresci-17*** datasets have considerably higher number of samples comparing to ***gilani-17*** and ***varol-icwsm*** datasets. It may affects the classification accuracy over different dimensions. Likewise, class imbalances may be another challenge over this approach. Although we used stratified sampling for all datasets, bot and human numbers is imbalanced in some of them and this is may be an issue over complexity analysis.

8. FUTURE WORK

Although our work answered most of our research questions and validated our hypotheses, more improvements needs to be added to broaden the extent of the research and to have more results to validate and support our findings in the future.

As we extended *Digital DNA* methodology by encoding crucial features from each tweet, we selected tweet type, inter-tweet times and entity information with a prior belief that the selected features characterize a tweet best. However, more features may added to the methodology since Twitter API stream comes with has much more information for tweets. More experiments needs to be conducted for selection of tweet features and our level system needs to be organized accordingly. Moreover, we added new levels in a cumulative manner and did not consider the position of each information. Since text compression algorithms are sensitive to repetitions and patterns, the positions of information in the encoded sequence strings are important and may give different results for different positions. We need to be more careful and rigorous to have reliable results.

Another important point in the future is that we should analyze more real events to evaluate our method. We had conducted a case study about German Federal elections on 2017 and compare the results we obtained for behavioral complexity differences before and during the election month. Nevertheless, a federal election is a major societal event and in that country, Twitter users were very active about the election when the election date approaches. Twitter's worldwide scope may allow us to conduct more case studies like ours in the future. Major societal events happens all the countries, sometimes globally like a recent COVID-19 pandemic. We believe that if we investigate more, we can find more dataset about a societal event and we can widen the scope of our complexity analyses by investigating the complexity changes of users during such events.

The case study we examined and temporal change analyses of our research brought new ideas and more questions about the results we obtained. We clearly conclude that bot or human account behaviors usually changes as time passes. Societal events

being one of key factors for such changes and there all other factors to influence them. However, we know that bot detection systems are sensitive for such dramatic changes for an account and they are designed to catch bot accounts. Behavioral changes on bot accounts thorough time is understandable in an aspect of they are adopting new systems and evolving. However, similar changes on human accounts may cause biasses over bot detection systems and our case study showed that an event like elections causes similar behavioral changes on human accounts. We need to investigate that if temporal changes on human account behaviors cause biasses over bot detection systems and if so, we should use our findings to improve the robustness of bot detection systems to make more reliable detections even during major societal event. That way, manipulation campaigns of bots would be easier to expose and prevent.

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Al-Masri, A. (2019). How does back-propagation in artificial neural networks work?

Ali, K. (2021). What are the types of feature learning algorithms?

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In Guyon, I., Dror, G., Lemaire, V., Taylor, G., & Silver, D. (Eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, (pp. 37–49)., Bellevue, Washington, USA. PMLR.

Brownlee, J. (2020). What is deep learning?

Cresci, S., Lillo, F., Regoli, D., Tardelli, S., & Tesconi, M. (2018). $fake: Evidence of spam and bot activity in stock microblogs on twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, *12*(1).

Cresci, S., Lillo, F., Regoli, D., Tardelli, S., & Tesconi, M. (2019). Cashtag piggybacking. *ACM Transactions on the Web*, *13*(2), 1–27.

Cresci, S., Petrocchi, M., Spognardi, A., & Tognazzi, S. (2019). Better safe than sorry. *Proceedings of the 10th ACM Conference on Web Science*.

Cresci, S., Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2016). Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intelligent Systems*, *31*, 58–64.

Cresci, S., Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2017a). The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race.

Cresci, S., Pietro, R., Petrocchi, M., Spognardi, A., & Tesconi, M. (2017b). Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, *PP*.

Crocetti, P. & Sliwa, C. (2017). What is data compression? - definition from whatis.com.

Davis, C. A., Varol, O., Ferrara, E., Flammini, A., & Menczer, F. (2016). Botornot. *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*.

Echeverrï!a, J., De Cristofaro, E., Kourtellis, N., Leontiadis, I., Stringhini, G., & Zhou, S. (2018). Lobo: Evaluation of generalization deficiencies in twitter bot classifiers. (pp. 137–146).

Ferrara, E., Varol, O., Davis, C., Menczer, F., & Flammini, A. (2016). The rise of social bots. *Communications of the ACM*, *59*(7), 96–104.

Ferrara, E., Wang, W.-Q., Varol, O., Flammini, A., & Galstyan, A. (2016). Predicting online extremism, content adopters, and interaction reciprocity. *Lecture*

Notes in Computer Science, 22–39.

Fitriya, L., Purboyo, T., & Prasasti, A. (2017). A review of data compression techniques. *International Journal of Applied Engineering Research*, *12*, 8956–8963.

Gilani, Z., Farahbakhsh, R., Tyson, G., Wang, L., & Crowcroft, J. (2017). Of bots and humans (on twitter). *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*.

Girin, L., Leglaive, S., Bie, X., Diard, J., Hueber, T., & Alameda-Pineda, X. (2021). Dynamical variational autoencoders: A comprehensive review.

Hemmendinger, D. (2013). Data compression.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, (pp. 278–282). IEEE.

Howard, P. N. & Kollanyi, B. (2016). Bots, strongerin, and brexit: Computational propaganda during the uk-eu referendum. *SSRN Electronic Journal*.

Hu, S. (2020). Compress.

Jean-loup Gailly, M. A. (2022). Zlib - compression compatible with gzip.

Kingma, D. & Welling, M. (2014). Auto-encoding variational bayes.

Kingma, D. P. & Welling, M. (2019). An introduction to variational autoencoders.

Kratzke, N. (2017). The #BTW17 Twitter Dataset - Recorded Tweets of the Federal Election Campaigns of 2017 for the 19th German Bundestag. Funded via general support for research by Lübeck University of Applied Sciences.

Kutzkov, K. (2022). 9 steps of debugging deep learning model training.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, K., Eoff, B., & Caverlee, J. (2011). Seven months with the devils: A long-term study of content polluters on twitter.

Maltby, H., Khim, J., Lin, C., Williams, C., Hernandez, A., Jackson, J., & Pakornrat, W. (2022). Markov chains.

Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., & Tesconi, M. (2019). Rtbust. *Proceedings of the 10th ACM Conference on Web Science*.

McCaffrey, J. D. (2018). The difference between an autoencoder and a variational autoencoder.

Paul, S. (2020). "reparameterization" trick in variational autoencoders.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rocca, J. (2021). Understanding variational autoencoders (vaes).

Sayyadiharikandeh, M., Varol, O., Yang, K.-C., Flammini, A., & Menczer, F. (2020). Detection of novel social bots by ensembles of specialized classifiers. (pp. 2725–2732).

Shao, C., Ciampaglia, G. L., Varol, O., Yang, K.-C., Flammini, A., & Menczer, F. (2018). The spread of low-credibility content by social bots. *Nature Communications*, *9*(1).

Van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Varol, O., Davis, C. A., Menczer, F., & Flammini, A. (2018). Feature engineering for social bot detection. *Feature Engineering for Machine Learning and Data Analytics*, 311–334.

Varol, O., Ferrara, E., Davis, C., Menczer, F., & Flammini, A. (2017). Online human-bot interactions: Detection, estimation, and characterization.

Wang, Y., Han, R., Lehman, T., Lv, Q., & Mishra, S. (2021). Analyzing behavioral changes of twitter users after exposure to misinformation. *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*.

Watson, C., Cooper, N., Palacio, D. N., Moran, K., & Poshyvanyk, D. (2022). A systematic literature review on the use of deep learning in software engineering research. *ACM Transactions on Software Engineering and Methodology*, *31*(2), 1–58.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1), 37–52. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

Yang, K., Varol, O., Davis, C. A., Ferrara, E., Flammini, A., & Menczer, F. (2019). Arming the public with artificial intelligence to counter social bots. *Human Behavior and Emerging Technologies*, *1*(1), 48–61.

Yang, K.-C., Ferrara, E., & Menczer, F. (2022). Botometer 101: Social bot practicum for computational social scientists.

Yang, K.-C., Hui, P.-M., & Menczer, F. (2019). Bot electioneering volume: Visualizingnbsp;socialnbsp;botnbsp;activitynbsp;duringnbsp;elections. *Companion Proceedings of The 2019 World Wide Web Conference*.

Yang, K.-C., Varol, O., Hui, P.-M., & Menczer, F. (2020). Scalable and generalizable social bot detection through data selection. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(01), 1096–1103.

Zafar, I., Tzanidou, G., Burton, R., Patel, N., & Araujo, L. Hands-on convolutional neural networks with tensorflow.

Öngün, C. (2020). Variational autoencoder (vae) nedir? autoencoder'dan ne farkı vardır?