AUTOMATED DETECTION OF SOCKPUPPET ACCOUNTS IN WIKIPEDIA

by

Mostofa Najmus Sakib

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2022

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Mostofa Najmus Sakib

Thesis Title: Automated Detection of Sockpuppet Accounts in Wikipedia

Date of Final Oral Examination: 23 June 2022

The following individuals read and discussed the thesis submitted by student Mostofa Najmus Sakib, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Francesca Spezzano, Ph.D.	Chair, Supervisory Committee
Edoardo Serra, Ph.D.	Member, Supervisory Committee
Nasir Eisty, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Francesca Spezzano, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

Dedicated to my beloved parents and all the researchers who devoted their lives to the path of science.

ACKNOWLEDGMENTS

First and foremost, I would like to convey my sincere gratitude to my advisor Dr. Francesca Spezzano, for her constant support towards this challenging yet memorable journey of learning. She has ceaselessly supported and motivated me to reach the goal of completing this thesis. Her mentorship has helped me evolve as an independent researcher. I will always cherish the memory of this journey and the freedom to research. The amount of learning would be helpful for me towards a better future.

I would also like to express my appreciation to Dr. Edoardo Serra and Dr. Nasir Eisty for generously serving on my thesis committee. Alongside my advisor, the committee members have always guided me with valuable advice and direction. All the other faculty members of the Computer Science Department at Boise State University have always been helpful as well and never turned me away from their guidance and support.

Finally, my gratitude and thanks to my family and all the Computer Science Department graduate students at Boise State University.

ABSTRACT

Wikipedia is a free Internet-based encyclopedia that is built and maintained via the open-source collaboration of a community of volunteers. Wikipedia's purpose is to benefit readers by acting as a widely accessible and free encyclopedia, a comprehensive written synopsis that contains information on all discovered branches of knowledge. The website has millions of pages that are maintained by thousands of volunteer editors. Unfortunately, given its open-editing format, Wikipedia is highly vulnerable to malicious activity, including vandalism, spam, undisclosed paid editing, etc.

Malicious users often use sockpuppet accounts to circumvent a block or a ban imposed by Wikipedia administrators on the person's original account. A sockpuppet is an "online identity used for the purpose of deception." Usually, several sockpuppet accounts are controlled by a unique individual (or entity) called a puppetmaster. Currently, suspected sockpuppet accounts are manually verified by Wikipedia administrators, which makes the process slow and inefficient.

The primary objective of this research is to develop an automated ML and neural-network-based system to recognize the patterns of sockpuppet accounts as early as possible and recommend suspension. We address the problem as a binary classification task and propose a set of new features to capture suspicious behavior that considers user activity and analyzes the contributed content. To comply with this work, we have focused on account-based and content-based features. Our solution was bifurcated into developing a strategy to automatically detect and categorize suspicious edits made by the same author

from multiple accounts. We hypothesize that "***you can hide behind the screen, but your personality can't hide.***" In addition to the above-mentioned method, we have also encountered the sequential nature of the work. Therefore, we have extended our analysis with a Long Short Term Memory (LSTM) model to track down the sequential pattern of users' writing styles.

Throughout the research, we strive to automate the sockpuppet account detection system and develop tools to help the Wikipedia administration maintain the quality of articles. We tested our system on a dataset we built containing 17K accounts validated as sockpuppets. Experimental results show that our approach achieves an F1 score of 0.82 and outperforms other systems proposed in the literature. We plan to deliver our research to the Wikipedia authorities to integrate it into their existing system.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AA	Authorship Attribution
AI	Artificial Intelligence
API	Application Programming Interface
LSTM	Long Short-Term Memory
ORES	Objective Revision Evaluation Service
RF	Random Forest
RNN	Recurrent Neural Network
UI	User Interface

CHAPTER ONE: INTRODUCTION

Wikipedia is a free Internet-based encyclopedia that started in 2001 [9]. It operates under an open-source management style and is maintained by the nonprofit Wikimedia Foundation [9]. They use collaborative software known as "wiki" that eases the creation, development, and distribution of articles. The goal of Wikipedia is to benefit readers by acting as a widely accessible encyclopedia that is free of cost and a comprehensive written synopsis that contains information on all discovered branches of knowledge [20]. Furthermore, generic audiences with minimal electronic device access benefit from Wikipedia articles because it presents a neutrally written summary of the available mainstream knowledge maintaining accuracy and fairness with a straightforward, "just-the-facts style" [20].

Collaborative projects like Wikipedia have been prevalent in recent times. The world's largest crowd-sourced encyclopedia has emerged due to its decentralized nature [29]. Given its open-editing format, Wikipedia is highly vulnerable to malicious activity, including vandalism, spam, undisclosed paid editing, etc. [22, 23, 24]. A free online forum like Wikipedia provides an excellent platform for users to communicate and share knowledge. On the other hand, it also facilitates online culprits to trick, scam, and increase the peril of universal users. According to Wikipedia's policies, each user is supposed to create only one user account to maintain clarity and increase community trust. However, Wikipedia does not have a strict provision for a one-user one-account system [30]. As a result, users are free to create multiple accounts according to their choice. This

freedom of creating user accounts with minimal information has led malicious users to create multiple identities and use them for various purposes, ranging from the promotion of products, pushing one's point of view, getting paid for articles, evasion of sanctions, false majority opinion claims, avoiding scrutiny, etc. [21]. If any user creates a secondary account for the above-mentioned malicious purposes, it is referred to as a sockpuppet. In technical terms, a sockpuppet is an "online identity used for creating deception" [21]. Usually, several sockpuppet accounts are controlled by a unique individual (or entity) called a puppetmaster.

In Wikipedia, any user proven to contribute false information to generate an extra payment, vandalize existing articles, or manipulate generic perspectives through falsifying information is identified as guilty. Such proof can result in an immediate ban which is imposed upon them for some hours to a day, depending on the severity of the crime. Malicious users often use sockpuppet accounts to circumvent a block or a ban imposed by Wikipedia administrators on the person's original account for unfaithful purposes [29].

Typically, different sockpuppet accounts or IP addresses are operated to continue such articulated works by taking advantage of Wikipedia's relaxed account creation policy. If any claim is pointed towards a user related to sockpuppetry, a sockpuppetry investigation case is filed. Unlike the simple account creation steps, the claim requires sufficient proof to result in a permanent ban. In addition, such claims need to be backed by concrete evidence related to manipulation, vandalism, advertising information, similar writing patterns, etc. [29].

Although, in most cases, multiple accounts are created for personal gain, there are a few situations where it is required to maintain more than one account. For example, there

might be a scenario where a content creator writes an article related to sensitive topics such as politics or religion. Editors might need to use pseudonyms for contribution to such cases as revealing their real identity can create hatred and result in life-threatening consequences. In addition, users are also allowed to make extra accounts for privacy issues. For example, if the primary account is compromised, maintaining security while connecting through an unsecured network, keeping privacy while editing highly controversial topics, a clean start under a new username, participating in educational purposes, testing the appearance of another account while creating content, etc. [21].

Currently, suspected sockpuppet accounts are manually verified by Wikipedia administrators, which makes the process slow and inefficient [29]. The existing works of sockpuppetry detection from faithful singular or multiple accounts have focused on the stylistic, syntactic, and social network-focused features predominantly through crosschecking the similarity of different account holders. Inherited semantic meanings of edits are rarely taken into consideration by prior researchers. Alongside account-based stylistic and syntactic features, we will emphasize in this study the content or, in other words, the semantic meaning of edits to investigate the patterns associated with the sockpuppet accounts held by the same user. Our research extends the prior works by bringing the semantics, i.e., users' writing patterns, tone, and additional elements of an edit, to connect to multiple account holders.

1.1 Thesis Statement

This thesis aims to detect the presence of sockpuppet accounts on Wikipedia. Our works apply machine learning and deep learning algorithms to outcast such accounts. Throughout this work, we have focused on finding answers to the following research questions.

RQ1: What are the patterns of sockpuppet accounts created by puppet masters?

RQ2: Does semantic analysis from edits capture the writing pattern and contribution pages more sophisticatedly and identify the sockpuppet accounts better than syntactic, stylistic, and graph network-based works and bring out a deep level of contextual meaning?

RQ3: Is it possible to detect sockpuppet accounts early and recommend suspension?

CHAPTER TWO: RELATED WORK

Sockpuppet accounts are generally utilized to enhance the internet traffic of undesired niche content, paid posts, controversial topics, and irrelevant documents by manipulating votes and views of the content [32]. In addition, those extra accounts are also used for specific malicious behavior such as fraudulent attempts, spamming, identify fraud, and malware distribution. In general, multiple fake identities are created by a user to manipulate users' perspectives, whereas the other forms of work include a sockpuppet group. A sockpuppet group can be a troupe of accounts created by one or multiple users to deviate the audience's attention to the targeted posts and generate an illusion of support [32].

The research history of sockpuppetry attempts on Wikipedia is not age-old. Until recent times such a concept was not established. With the emergence of social media and online platforms, multiple identity generation and fraudulent attempts on online platforms have become more prominent. Wikipedia has made its admin-based evaluation of sockpuppetry claims publicly available. Traditionally researchers have taken advantage of those publicly available data to move forward with the sockpuppetry investigation.

In the literature, several works have analyzed and detected sockpuppet accounts in online social networks and discussion forums [25, 26, 27,28]. The initial approach to the sockpuppet detection problem revolved around the authorship attribution (AA) detection. All those types of AA detection generally followed a text classification framework where the authors were the number of classes. Historically, such works include simple and easy-

to-implement machine learning algorithms for classification [1,2,3,4,5]. Specifically to Wikipedia, Solorio et al. [29,30] have addressed the problem of detecting whether or not the same user maintains two accounts using text authorship identification features. They have extensively focused on the comments and edits on talk pages and considered features such as punctuation marks, use of emoticons, capitalization, and part-of-speech to characterize the user writing style. Many of those earlier researches [29,30] drew our attention to the fact that low-level features like character n-grams can successfully identify unique writing styles. Their analysis reemphasized that semantic features such as bag-of-words, stylistic features such as punctuation marks, use of emoticons, capitalization information, and syntactic information like part-of-speech level, all these types are particularly useful for sockpuppetry detection [29]. A different kind of work followed the ideology of similarity-based approaches. Author-specific features aided the process in such cases as similarity-based scores are usually calculated from them [6,7,8].

Yamak et al. [31] have focused on classifying sockpuppets vs. genuine accounts by using non-verbal behavior and considering editing patterns. They considered Wikipedia-specific features, i.e., the number of edits, frequency of revert after each contribution in the same article, the time between registration and edits, etc. In continuation of the work, the same authors also addressed the grouping of detected sockpuppet accounts created by the same individual [32]. The authors developed relational graphs and combined them with community detection algorithms and account-focused attributes to catch sockpuppet groups. Tsikerdekis and Zeadally [33] performed a Wikipedia-focused analysis to detect identity deception through possessing non-verbal user activity. Their experiment reflected on 7,500 sockpuppet accounts with at least one revision and calculated non-verbal

behavior, including the number of total revisions on different Wikipedia pages (article, article discussion, user page, user discussion page), and the average number of bytes added or removed.

Zheng [34] executed a sockpuppet analysis by considering sockpuppets in the same forum and cross-platform. They compared keyword-based similarity profiles for posts A1 and A2 in two different forums and evaluated the probability of being a sockpuppet pair. They assumed puppet masters tend to follow similar writing patterns even if they use multiple accounts.

Like Wikipedia, multiple account generation is prevalent in miscellaneous online social media. For instance, Maitry et al. [35] analyzed sockpuppet accounts on Twitter, and Swati Adhikari [36] performed a similar sockpuppet detection on Reddit data. In addition, Maitry et al. [35] emphasized real-time tweets and profile-focused features to identify accounts under the same user in a quick time, whereas Swati Adhikari [36] included Reddit users, their posts, subreddits, and their karma scores. However, both works are platform-dependent and cannot be generalized on other cross-platforms.

A multiple online community-based analysis was conducted by Kumar et al. [28]. The authors analyzed sockpuppetry behaviors across nine different communities. Their in-depth analysis revealed that the sockpuppets differ from ordinary users regarding their pattern of social media activity and corresponding social network structure. For example, they pointed that sockpuppets follow unique linguistic traits (more singular first-person) and have more chances of posting on the same discussion in a short timeframe. In addition, they claimed sockpuppet pairs follow similar writing styles and patterns compared to regular contributors.

Joshi et al. [24] investigated the use of sockpuppet accounts to perform undisclosed paid edits on Wikipedia. They found that sockpuppet accounts associated with undisclosed paid editors only work on a limited number of Wikipedia titles they are interested in promoting, whereas genuine users edit more pages related to their field of expertise. This shows that sockpuppets accounts' behavior in Wikipedia differs from sockpuppetry in online discussion communities, where sockpuppets' main goal is to interact with each other to deceive other users [28].

CHAPTER THREE: METHODOLOGY

This section describes how we built a dataset containing sockpuppet and benign user accounts. We have collected and analyzed sockpuppet investigation data through an API (Application Programming Interface) that retrieves relevant information from Wikipedia. The following chapter describes our methodology and guides readers to apply the same methods to other problems. We started with defining the dataset curation process and later included the feature description and extraction process.

3.1 Dataset

For collecting the Wikipedia data, we have used the MediaWiki Action API [10]. The MediaWiki Action API is a web service that allows access to some wiki features like authentication, page operations, and search. In addition, it can provide meta-information about the wiki and the logged-in user.

To start with the Wikipedia data collection through API, we have looked for all the subcategories that fall under the major category "**Suspected Wikipedia sockpuppets**." All those subcategories under the major category were retrieved until 28th May 2022. These subcategories are sockpuppetry accounts identified by Wikipedia. All those subcategories usually follow the standard naming convention of Wikipedia and start with "**Wikipedia sockpuppets of**" followed by the account name. For instance, "Wikipedia sockpuppets of -dantbh" is a subcategory of sockpuppetry cases. Once all the Wikipedia subcategories were extracted, we focused on the user accounts under each sockpuppet subcategory. Usually, each sockpuppetry subcategory (for example, Wikipedia sockpuppets of -dantbh)

had multiple user accounts under the same account name. Our selected example subcategory (Wikipedia sockpuppets of -dantbh) had 20 different user accounts for the same account. Once the user accounts were retrieved for each user under each subcategory, we looked for each user's contributions or edits. Our focus of the analysis was the contribution of each user. This contribution includes various kinds of information for each edit of the users. Based on the default parameter settings for the users, the generic format and the retrieved data look like figure 3.1 for a user.

```
{
    "batchcomplete": "",
    "continue": {
        "uccontinue": "20190130180447|880978627",
        "continue": "-||"
    },
    "query": {
        "usercontribs": [
            {
                "userid": 24,
                "user": "Jimbo Wales",
                "pageid": 9870625,
                "revid": 881893498,
                "parentid": 881892978,
                "ns": 3,
                "title": "User talk:Jimbo Wales",
                "timestamp": "2019-02-05T14:05:11Z",
                "comment": "/* Fancy I edit Wikipedia T-Shirt */",
                "size": 29753
            },
            {
                "userid": 24,
                "user": "Jimbo Wales",
                "pageid": 9870625,
                "revid": 881282261,
                "parentid": 881270759,
                "ns": 3,
                "title": "User talk:Jimbo Wales",
                "timestamp": "2019-02-01T15:29:31Z",
                "comment": "/* Macedonian President Gorge Ivanov is
now in the House arrest */",
                "size": 60166
            },
```


Figure 3.1 Basic dataset structure from Wikipedia API

For each edit, we retrieved the following information: the username (user), the userid, the page id, the parent id, the revision id, page namespace (Wikipedia groups articles into multiple categories or namespaces, namely article, article discussion, user page, user discussion page, project, etc.), the page title, the edit timestamp, the text of the

user contribution, and the size of the user contribution. A list of Wikipedia namespaces is shown in figure 3.2.

Namespaces			
Subject namespaces		Talk namespaces	
0	(Main/Article)	Talk	1
2	User	User talk	3
4	Wikipedia	Wikipedia talk	5
6	File	File talk	7
8	MediaWiki	MediaWiki talk	9
10	Template	Template talk	11
12	Help	Help talk	13
14	Category	Category talk	15
100	Portal	Portal talk	101
118	Draft	Draft talk	119
710	TimedText	TimedText talk	711
828	Module	Module talk	829

Figure 3.2 Wikipedia categories by namespaces

In our dataset, userid and user are the unique id and name for a user account. We identified all the accounts related to sockpuppetry as positive datasets for sockpuppet detection purposes. We initially collected a total number of 20,978 sockpuppet categories mentioned under the Wikipedia sockpuppets category. However, after intensive cleaning and removing empty and nan comments, we remained with 17,180 valid sockpuppet accounts.

3.2 Negative Data

To contrast the positive or identified sockpuppet account, we also needed some account information that is either identified as a genuine user or never had any claims against their accounts. We will be calling such examples negative samples. To get the negative dataset, we depended on the works of Kumar et al. [22]. They recorded 16,496

positive accounts, and we have used their recorded accounts as examples of negative users. Their reported dataset contains usernames which are identified as benign users. With a similar approach to the data retrieval process from Wikimedia API, the contribution of the benign users was downloaded as the set of negative users. To be consistent with the sockpuppet or positive dataset, we went through the same cleaning process for the benign users and remained with 16,043 final cases. So our combined dataset was almost balanced.

For each of the considered accounts (both sockpuppets and benign users), we retrieved their first 20 edits. We considered 20 edits for each user as our goal is to build an automated detection system that can identify sockpuppet accounts as early as possible.

3.3 Account-based Features for Identifying Sockpuppet Users

In this section, we will describe and list down all the features we have used for sockpuppet detection.

As mentioned in the data extraction process, we have a bunch of account attributes available from the contribution section of the user accounts. Based on that information, we have fixed several features derived from the users' account names. From previous literature, it is evident that username is an important feature to detect spammers, undisclosed paid editing, sockpuppetry, and other malicious behavior [23, 24, 37]. Hence we considered the following features extracted from the username:

The number of digits in a username: In order to create several accounts, sockpuppet users sometimes focus on creating similar account names with additional digits as the differentiator. That is why we have considered the number of digits in the username as an impactful indicator of sockpuppetry.

The ratio of digits to total alphabet characters in a username: Like the digits, characters are also a critical component of any username. Multiple account users often create an additional account just by tweaking some characters. In this feature, we have focused on the ratio of digits to total alphabet characters in the username to capture similar usernames with minor changes.

The number of leading digits in a username: To differentiate between the usernames, puppet masters sometimes create accounts with leading digits that can distinguish between account names. To catch that sort of behavior, we have also focused on the number of leading digits that's been used as a username. However, using leading digits is distinctive behavior compared to using numbers anywhere else in the user name. So, the total number of digits and username with the leading digit would be capable of capturing two different naming convention patterns.

The unique character ratio in username: This feature focuses on the unique character ratio in the username. To derive this feature, we calculated the unique characters of the username and divided it by the total length of the user name.

In addition to the username-focused features, we have included user characteristics to discover the hidden pattern of sockpuppet users. The following features are extracted to identify a user's generic writing styles and norms.

Average contribution length: An essential piece of information retrieved from each user's contribution was their comments on each successive edit. Since benign users try to collaborate and contribute more, the length of the comment should be higher than their counterparts. That is why we considered the comment length a critical feature.

Average title length: We considered the average length of the titles of the pages a user-contributed to.

Average time difference between two consecutive edits: The behavior over time is an essential feature for detecting any fraudulent activity [38]. Therefore, we considered the average time difference between two consecutive contributions as another feature.

All the features mentioned earlier were calculated for each contribution of the user accounts. However, our focus is on detecting sockpuppet users, not their contributions. To serve that purpose, we have averaged the values of all the previously described features for each user. So the username-based features would be exactly the same for each user. However, each contribution's comment or title length and the time difference are different. So for these three features, we have calculated their average value.

3.4 Content-based Features for Sockpuppet Detection in Wikipedia

The second category of feature we examined is content-based, for which we have evaluated edit content. Each edit is considered a single document in this case and carried out through the later-described process to elicit content-based features for our analysis.

We have followed two basic approaches to analyze the content of user contributions for sockpuppetry detection. One includes using the BERT transformer model [39], and another was integrating topic modeling to add topics of an edit as features for our analysis. The major motivation behind applying the transformer model and the topic modeling is to capture the semantics and meaning of the content. Traditionally sockpuppetry detection and similar NLP tasks have been primarily focused on capturing the syntactic inheritance and stylistic of the content [29, 30]. Little emphasis has been put on semantics-focused features. Our major contribution through this research is to bring in

the semantics meaning to understand the deep inheritance of the content or edit. The syntax is the set of rules needed to ensure a sentence is grammatically correct. Semantics, on the other hand, is how one's writing pattern, grammatical structure, tone, and other elements of a sentence coalesce to communicate its meaning.

We hypothesize that considering the semantics of the user edits would capture the deep-level pattern of the content from the edits done by the same puppet master. For example, if a puppet master focuses on a specific type of content or person, that account holder will edit or publish similar content from multiple accounts. Since the behavioral pattern of the puppet master can be captured more efficiently through semantics, we decided to include the BERT embeddings and topic modeling in our study. For example, suppose a puppet master or group account holder tries to edit the pages related to Barack Obama. In that case, there is a high probability they would do that similar edit from multiple accounts. Capturing the semantic meaning would be the ideal step to shed light on such a problem. That is why in continuation to the stylistic or syntactic-focused analysis by previous researchers, we will carry out semantic-based research for further improvement.

3.4.1 BERT Embeddings

In this approach, we have put our concentration on the state-of-the-art transformer models. The transformer model is now widely used for several natural language processing tasks, i.e., machine translation [15], named entity recognition [16], biological sequence analysis [17,18,19], etc. We would also like to use a similar technology to see if transformer models can better perform to understand the sequential editing patterns compared to the existing approaches described in the related work section. The BERT model is our choice for this task.

BERT stands for Bidirectional Encoder Representations from Transformers. It is a unique deep learning model that works upon the attention process. Every output element in the model is connected to all the input elements and keeps the information flow by adjusting the weights. This unique process of connecting refers to as the attention mechanism and makes the whole system robust and powerful.

BERT generally uses the attention mechanism to understand the contextual relationship between words. Two separate steps (encoding the text and decoding for prediction tasks) go harmoniously and extract the deep inheritance relationship between words in a text. It specifically helps to resolve ambiguity in texts by revealing the context. Unlike the directional model, which reads words sequentially (either left to right or right to left), the BERT encoder takes the entire sentence as one input. This simple strategy helps to understand the whole context of a text instead of focusing word by word. This specific capacity was included by the introduction of transformers and referred to as bi-directionality.

We used the BERT model to compute the embedding of each user contribution. Specifically, we used the BertTokenizer for tokenization and converting to tensors and the BERT "base" model trained on lower-cased English (12 Transformer layers, 12 self-attention heads, hidden size of 768) from the Huggingface library [39]. Our choice of feature-based approach here comprised extracting the activations (or contextual embeddings or token representations or features) from one or more of the 12 layers without fine-tuning any parameters of BERT. The model contributes 768 contextual embeddings from each layer, and the output from the last layer was used as input to regular machine

learning and LSTM, followed by the classification of benign users from sockpuppet accounts.

3.4.2 Topic Modeling

Topic modeling is a way of discovering high-level topics through statistical modeling with respect to document collection. Our hypothesis is that identifying the contents' topic can contribute significantly to detecting multiple identities. Users with good faith usually contribute to various sorts of content. However, sockpuppet users tend to post similar content even if they were removed earlier. To comply with this premise, we have taken advantage of the Latent Dirichlet Allocation (LDA) topic modeling technique provided by the Gensim library (we used the WordNetLemmatizer and the bigram model) [12]. LDA is a simple yet powerful topic generation process from a given corpus.

To utilize the techniques mentioned above, we have retrieved the summary of content or contribution of the users again through the MediaWiki Action API. Before this work, we analyzed a single comment or edit made by each user. However, we required more information to understand and calculate topics through LDA. MediaWiki API has another parameter named "extracts" which returns any page's plain-text or limited HTML. Through the similar data collection process described in section 3.1, we retrieved the contents for each user. Finally, we used those content for extracting topics using LDA.

The content that we received through the API consisted of HTML tags, extra punctuations, and spaces. Before feeding to the LDA model, this data required extensive cleaning. First, we have followed the basic text cleaning process, removing punctuation, extra spaces, and additional special characters. Later through tokenization and lemmatization, we prepared the raw texts for the next steps. Once we had the tokens for

each observation point, we developed a bigram model followed by a corpus on the entire data set, combining the sockpuppet and benign data. Specifically, we trained an LDA model with 20 topics on all the users' comments and then assigned to each comment a vector with the corresponding topic distribution.

3.5 Classification Models

In order to test the features we are proposing for the automated detection task, we considered different classifiers, namely Logistic Regression, Gaussian Naive Bayes, Decision Tree, Multilayer Perceptron (MLP) Classifier, Random Forest, ExtraTree Classifier, and a Long short-term memory (LSTM). LSTMs are a complex area of deep learning whose network is a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is achieved because the recurring module of the model has a combination of layers interacting with each other. The above-aforementioned methodology would help us to understand the dependability of the temporal sequence of users' edit patterns. We have made the following considerations in deciding on architecture selection:

i. The problem of detecting an editor's comment is a classification task based on the edit history as such kind of data is generated while editors edit over a time period.

ii. In order to predict the sequence of a user's edit behavior at any time step, it is essential to learn from its behavior or action from earlier time steps. This gives our solution holistic feedback from prior time steps to the current step.

iii. Additionally, LSTM can relay a constant flow of feedback without vanishing or exploding for a long sequence.

Although LSTM is a precise form of Recurrent Neural Network (RNN), unlike RNN, LSTM incorporates input, forget and output gates [13] that effectively resolves the problem of vanishing or exploding gradient. In our approach, we used an LSTM model architecture with a many-to-one setup or hidden layer output from only the last layer, as shown in figure 3.3.

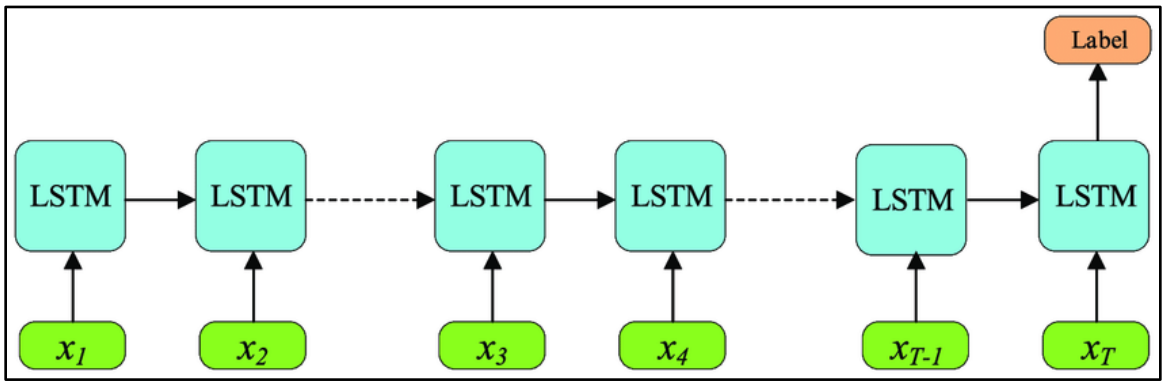

Figure 3.3 Many-to-one LSTM architecture

We have used class-specific weighting to deal with class imbalance. In addition, this process allows the model to consider the entire sequence of a contributor before classifying an edit. With such model architecture, a standard cross-entropy loss function takes the form shown in equation 3.1.

$$loss = \sum_{u \in U} CE\ (W^T . h_L) \qquad (3.1)$$

Here,

u € user in the set of users U

L = length of edit sequence of user u

For classical machine learning models, we considered all features described in the methodology section plus the average vector of the user contributions' BERT embeddings and the average vector of the user contributions topics to capture the user semantics. One of our fundamental contributions through this research is to detect sockpuppet accounts

quickly. We have experimented with editing one to twenty sequentially, resulting in twenty different scenarios. For example, in the first scenario, we only took the first edit of each user's contributions and evaluated all the features required for the classical models. For the second scenario, we took two consecutive edits and similarly calculated all the features again, and averaged for each user. We continued the same pattern for the rest of the edits, increasing the number of edits by one each time. By the end, we had results for k (1 to 20) edits at the user level as we averaged the features at the user level.

For the LSTM model, we considered in input the sequence of features for each edit. For each edit, we considered the contribution length, the title length, the time difference between the current and previous edits, the BERT embedding of the contribution, and the vector of topics of the contribution. Finally, we concatenated the username-based features to the representation of the last cell of the LSTM and passed them to the classification layer. The contribution of the articles was not homogeneous for each user for neither benign nor sockpuppet users. We used padding in case there were less than 20 contributions by the editors to make each user input to the LSTM a fixed size of the number of features X 20.

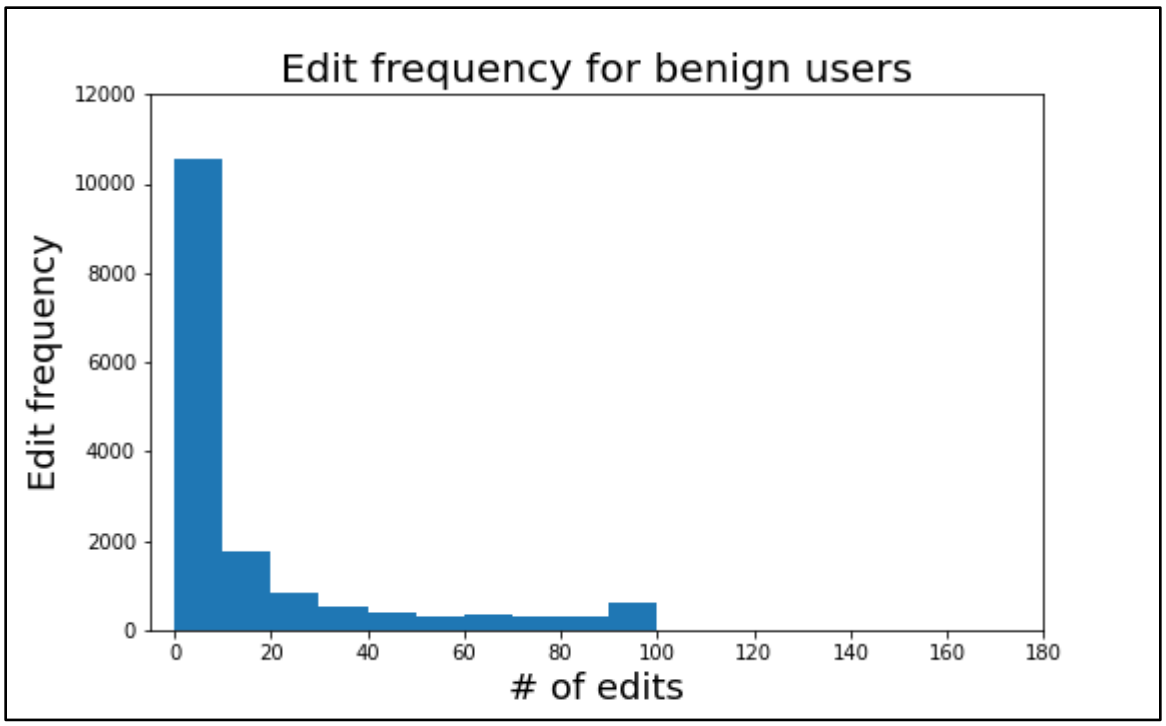

Figure 3.4 Edit frequency for benign users

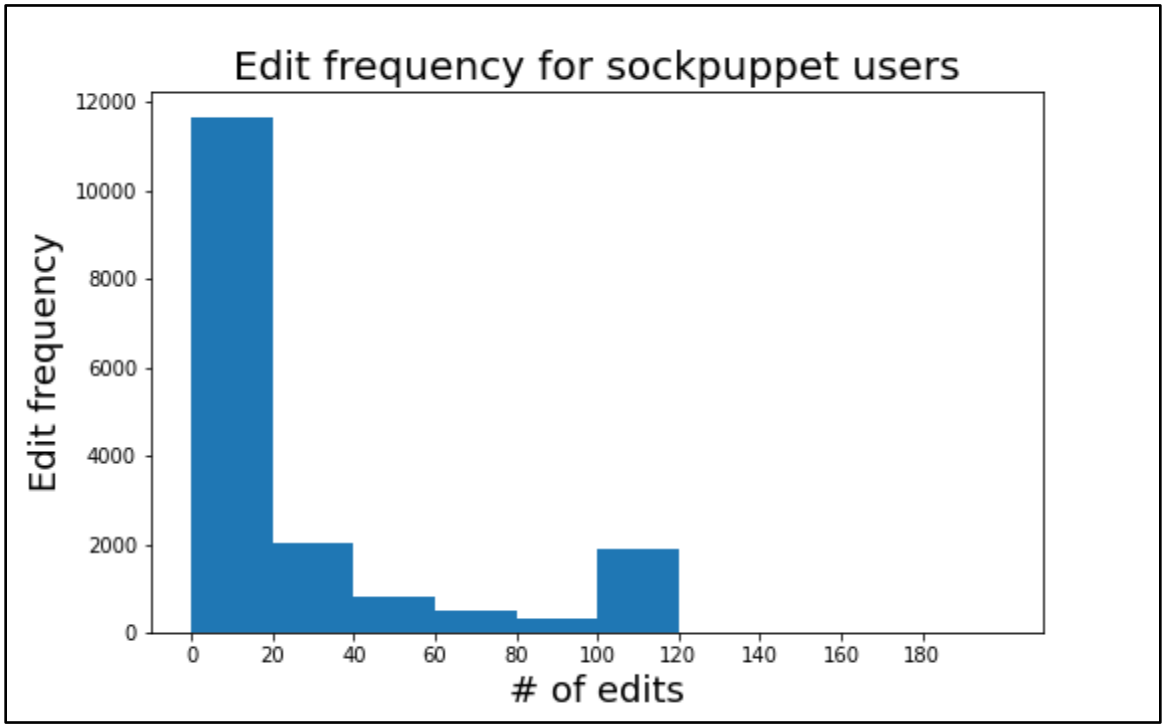

Figure 3.5 Edit frequency for sockpuppet users

Figure 3.4 and 3.5 shows the distribution of comments by each user for the benign and sockpuppet categories. An interesting pattern can be seen from the number of contributions by editors in both cases. After 100 usually, the benign users didn't have any contribution, but the sockpuppet users kept contributing. Since there are few comments after 20 edits, we considered 20 edits for each category to avoid padding many zero values in LSTM and detecting sock puppetry quickly.

3.6 Evaluation of Proposed Methods

This section reports on our evaluation protocol.

3.6.1 Metrics

To evaluate our model's performance, we have used the F1 score. F1 score is the weighted average of Precision and Recall. Precision refers to the total number of correctly classified positive data compared to the total number of positive data. The recall is the ratio of correctly predicted positive observations to all observations in actual class - yes. Therefore, the F1 score takes both false positives and negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more helpful than accuracy, especially if we are dealing with an uneven class distribution.

3.6.2 Comparison with Related Work

To compare our work with the prior results, we have also included the works done by Solorio et al. [29] and Yamak et al. [31] in our research. Both tried to detect sockpuppet accounts using different feature sets and approaches but had similar objectives to ours. The work of Solorio et al. [29] was one of the preliminary works done on sockpuppet detection, whereas the last one is more recent. We have compared the previously mentioned metrics

through our methodology and their approach and tried to devise a more accurate way of detecting sockpuppetry as early as possible.

Solorio et al. [29] approached the problem from the authorship attribution perspective. Every single comment made by the user is considered one document, and they were classified to check the sockpuppetry claims. They worked following two steps. In the first step, they collected the comment level prediction for each account. Then through a majority voting schema, they put the account in the suspected or benign category. Their mentioned feature sets are specified below.

Total number of characters: The authors calculated this feature to model the contributor's behavior of writing, specifically long texts or short comments.

Total number of sentences: This feature computes the total number of sentences in the comments. The authors assumed this would be a valuable feature to identify contributors' choice of organizing text in sentences. To count sentence numbers, we have taken advantage of the sent_tokenizer package from NLTK.

Total number of tokens: The total number of tokens excluding the white spaces are counted here. We have used the word_tokenizer package from NLTK to compute this feature.

Words without vowels: The rate of words without vowels might indicate a signal for some contributors. Examples of words without vowels are: try, cry, fly, etc.

Total alphabet count: This feature is the summation of all the alphabetic characters in the text.

Total punctuation count: The user's choice of punctuation usually varies in unique ways. For example, semicolons and hyphens are commonly used by some

contributors, and the rest ignores them. Some punctuation also varies in the way it's been used worldwide. For example, the use of commas is distinctive, an important feature in detecting writing patterns.

Two or Three punctuation count: In modern days, many formal and informal writing contains the use of multiple punctuation marks to put importance or simply express emotions. Such cases can be identified by checking the use of multiple punctuation marks used by contributors. Therefore, the authors believe various ways of expressing emotions would be an ideal indicator of sockpuppet users.

Total contraction count: Contractions are generally used to shorten and combine words, i.e., don't, it's, and I'm. Separately used or contracted form, both cases are correct in English grammar. However, how a contributor writes or contributes is a choice of personal preference, and the calculation of contraction is an ideal way to extract the writing pattern or behavior.

Parenthesis count: This feature is a generic way to determine authorship attribution and would play an important role in distinguishing contributors.

All caps letter word count: The authors counted the number of tokens where all the words were upper case letters. Traditionally contributors use all caps letters either as abbreviations or to emphasize some words. Some examples are "USA" or "the word was pronounced INCORRECTLY."

Emoticons count: In today's arena, expression and writing style are widely dominated by emoticons, especially in writings on web pages. Emoticons are a pictorial representation of feelings, especially facial expressions and internal emotions. The authors

evaluated the usage pattern of emoticon selection by counting the total number of emoticons in the content.

Happy emoticons count: People are usually biased while selecting emoticons or expressing feelings. Many users only express positive or happy feelings. Happy emoticons dominate such writings. The authors separately counted the happy emoticons such as :) and :-) to evaluate the contributors.

Sentence count without capital letter at the beginning: Some contributors prefer to start writing with a small letter or number. Examples of such cases can be "1862 was the year" or "big and bold all apply to our suspect." The authors believe this feature would also capture the unique writing pattern.

Quotation count: Similar to parenthesis count, authorship contribution is also essential to detect authorship contribution. In a real-life scenario, users are distinctive with their choice of quotation. So quotation count would help to discriminate writers from others.

Parts of speech (POS) tags frequency: The authors considered 36 parts of speech tags from the Penn TREE-bank POS tag set and removed the punctuation marks as those were already considered through other features.

Frequency of letters: English alphabet contains 26 letters, and the frequency of those letters in each comment was computed as separate features. The count was normalized by the total number of non-white characters in each comment.

Function words frequency: Choice of functional words is an excellent way to tag writers to their corresponding writings. For example, the authors considered a list of

function words from [11]. This choice created 150 features from a list of 150 function words.

All the above-mentioned features are typically used in authorship attribution, and the authors integrated some more features through manual inspection of their Wikipedia dataset.

Small "i" frequency: Small "i" in place of "I" was commonly used by some Wikipedia contributors. It was interesting that contributors were prone to this mistake.

Full stop without white frequency: Many writers forget to add white space after the full stop, and this was counted as a feature to distinguish sockpuppet accounts.

Questions frequency: A few authors use question marks more often than others. So, this is an idiosyncratic feature as the authors claim some writers abuse the use of question marks for sentences that do not require question marks or use multiple question marks where one question mark would suffice.

Sentence with small letter frequency: The authors observed a homogeneous writing pattern of not starting a sentence with capital letters, and they considered this a feature to examine unique writing habits.

Alpha, digit, uppercase, white space, and tab frequency: The authors mentioned that this group of characters usually varies between Wikipedia contributors. So this would capture the formatting preferences of texts such as "zero" and "one" instead of "0" and "1" and uppercase letters for every word.

"A" and "an" error frequency: Wikipedia users often make mistakes while typing "a" and "an'. Many content creators are habituated to such mistakes, and considering those can help us to detect sockpuppet cases.

“he” and “she” frequency: Choice of “he” and “she” is preferential to each contributor. The authors mentioned that any contributor’s use of “he” or “she” for an indefinite subject is consistent across edits or comments in different articles or talk pages.

We averaged all the above-listed features among the same user contributions when putting them in input to classical machine learning classifiers. At the same time, we considered the feature sequence in input to LSTM.

Yamak et al. [31] experimented with a few types of features in their work. Those are listed below.

The number of users’ contributions by namespaces: The user’s contribution is basically categorized into six types. These are article, article discussion, user page, user discussion page, project namespace, and other (all the other namespaces goes into this category). The authors assumed that the categories mentioned above are the most important in terms of detecting the writing behavior and interest of Wikipedia users.

The average of bytes added and removed from each revision: With the desire to identify the writing patterns of user’s behavior, the authors calculated the average of the numbers of bytes of the information that was added in the article for all the contributions (revision) of each account. They also calculated the average number of bytes of the information removed in the articles for each account’s contributions. Their hypothesis was the manipulation of Wikipedia contributors can be checked through the addition/removal behavior.

The average contribution in the same article: The idea behind the inclusion of this feature was to compute the average number of time an author contributes to an article.

The authors assumed manipulators usually try to manipulate the same article multiple times.

The interval between the user's registration and his first contribution: For this feature, the authors calculated the difference between the registration and the time of the first contribution in the EnWiki by each account. They assumed sockpuppet users create many accounts at the beginning and later leave them unused. However, these backup accounts are resued when an active account is blocked.

The frequency of revert after each contribution in the same article: The underlying hypothesis for this feature is that most of the manipulation of a sockpuppet user will be reverted by another user, as multiple contributors generally manage each page. Whenever they find a malicious contribution, they usually revert them directly.

The last feature considers whether an edit has been reverted by another user, making the detection not completely automated as human input is required. As we propose an automatic detection approach that does not rely on human input, we did not include the reverted-based feature in our implementation of the Yamak et al. [31] approach for a fairer comparison. We also excluded the interval between the user's registration and his first contribution as we do not have this information in our dataset.

3.6.3 Comparison with ORES

Objective Revision Evaluation Service (ORES) is a machine learning-based prediction system as a web service that provides services for Wikimedia projects like Wikipedia and Wikidata. Such a system is designed to help human editors perform sophisticated tasks while considering Wikipedia as an information source [14]. In addition, ORES can detect vandalism and remove edits that were not done in good faith. ORES is

developed by the Wikimedia Scoring Platform [14]. They are experts in developing easy-to-access AI (Artificial Intelligence) based models which are transparent and ethical. This open-access tool aids in human decision-making.

ORES is designed as a back-end service and was intended to generate structured information by developers. To retrieve the ORES scores, a simple scores API (Application Programming Interface) and a reference UI (User Interface) is available [14]. Many researchers also access ORES via third-party tools developed by volunteers.

We have also used the available API to gather ORES scores for each edit for both the benign and sockpuppet users' contributions More specifically, given an edit, ORES provides a probability distribution (draft quality scores) of being in one of the following four classes: spam, vandalism, attack, or OK. The faster seriously problematic types of draft articles are removed, the better. We averaged the draft quality scores of all the edits of the same user when using classical machine learning algorithms while we considered the sequence of the draft quality scores for the edits of the same user in input to the LSTM.

CHAPTER FOUR: EXPERIMENTAL RESULTS

This section will look at the experimental results from our machine learning and neural network-based approach. For this, we have utilized features described in the methodology section. We have performed our analysis by considering all the features described in Section 3 to determine the sockpuppet accounts on Wikipedia. Details of the analysis steps and works are presented in this chapter.

4.1 Final Dataset Size

We have used the entire dataset of the positive and negative samples mentioned in the methodology section. After collecting and cleaning, we had nearly a balanced dataset. However, the contributions of those accounts' total number of edits were different. The final dataset sample counts are listed in Table 4.1.

Table 4.1 Number of final samples

	Positive data	Negative data
Number of users	17,180	16,043
Total number of edits	420,111	393,950

4.2 Experiment Process and Setup

As described in the methodology section, we have used several classification algorithms for our features to build a model ideal for separating genuine accounts from multiple account holders. The dataset that we used was pretty much balanced. So, we did not need to use any class imbalance techniques. However, to be on the safer side, we have

focused on stratified cross-validation. We have done a 5-fold cross-validation. To measure the performance, we considered the F1 score.

4.3 Results of Our Proposed Features

Results of different machine learning models with our proposed features are shown in Table 4.2. As we can see, among all the considered machine learning models, Random Forest achieves the best F1 score of 0.82. Furthermore, these models perform better than LSTM, which achieves a lower F1 score of 0.75.

Table 4.2 F1 score comparison of different machine learning models with our proposed features in input to predict sockpuppet accounts. The best scores are in bold.

Classifier	F1 score
Random Forest	**0.82**
Logistic Regression	0.75
Extra tree classifier	0.75
Gaussian Naive Bayes	0.60
Decision tree	0.75
MLP classifier	0.77
LSTM	0.75

4.4 Feature Analysis

To measure the feature importance, we performed feature ablation, i.e., for each group, g of considered features were moved and performed the classification with the remaining features. The higher the drop in the F1 score, the more important the group of features for the classification task. Results are shown in Figures 4.1 and 4.2. As we can see, the most important group of features is the one of LDA topics, as removing it decreases

the F1 score to 0.71 for 20 edits. The second most important feature group contains the average contribution length, the average title length, and the average time difference between two consecutive edits. Removing this group of features decreases the F1 score to 0.81. Username-based features and the BERT embedding of user comments are equally important, and removing one of them slightly decreases the F1 score. Removing both of them drops the F1 score to 0.81. Figure 4.2 ensures this pattern is consistent even if fewer edits are considered.

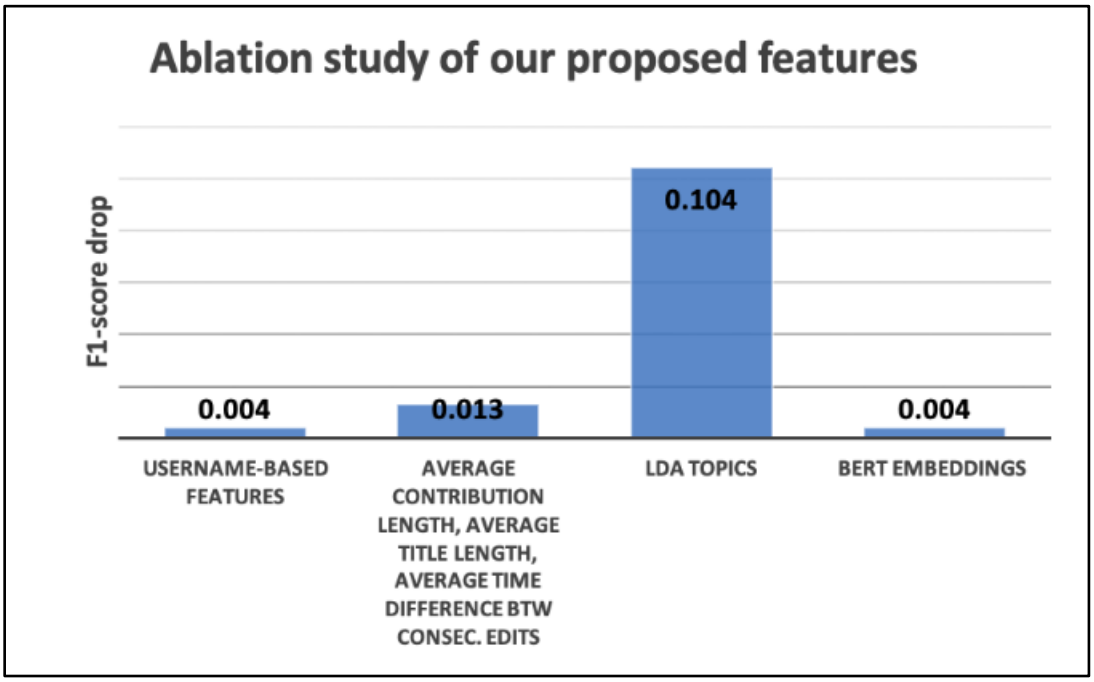

Figure 4.1 Ablation study of our proposed features: drop-in F1 score for each considered group of features

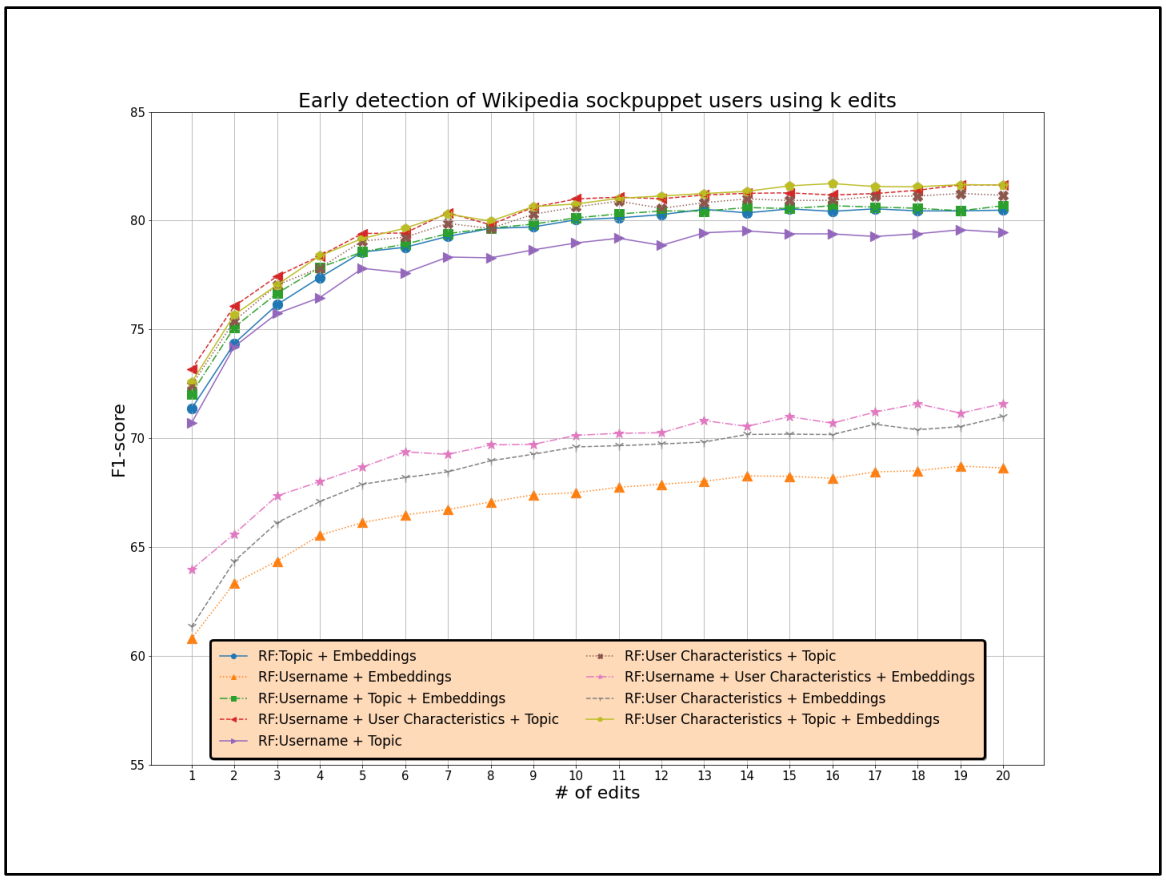

Figure 4.2 Ablation study of our proposed features for k edits

4.5 Comparison of Our Proposed Method with Related Work

The F1 scores of our proposed approach and the considered competitors are shown in Table 4.3, where we also compare the features in input to the best classical machine learning model (Random Forest in the case of all competitors) and LSTM. As we can see, our proposed approach achieves a higher F1 score of 0.82 as compared to ORES with Random forest (RF), Yamak et al. [31] with RF, and Solorio et al. [29] with LSTM, which achieve an F1 score of 0.54, 0.64, and 0.77, respectively.

Table 4.3 F1 score comparison of our proposed features vs. related work. We compare features in input to Random forest (which results in the best classical machine learning algorithm) and LSTM. The best scores are in bold.

	Random forest	LSTM
Our proposed features	**0.82**	0.75
ORES	0.54	0.53
Yamak	0.64	0.59
Solorio	0.75	0.77

4.6 Early Detection of Wikipedia Sockpuppet Accounts

We study the effect of the first-k edits made by the user on the prediction F1 score. Figure 4.3 shows the variation in the F1 score when k is varied from 1 to 20. We show our features compared to related work features in input to Random Forest and LSTM. Our proposed set of features is able to detect a sockpuppet account with an F1 score of 0.73 by just considering the user's first edit (vs. 0.68 achieved by Solorio et al. [29]) and an F1 score of 0.80 by considering the first six edits. Moreover, Random Forest is always better than LSTM, especially for early prediction. The only exception is given by Solorio et al. [29], where LSTM is slightly better starting from 12 edits.

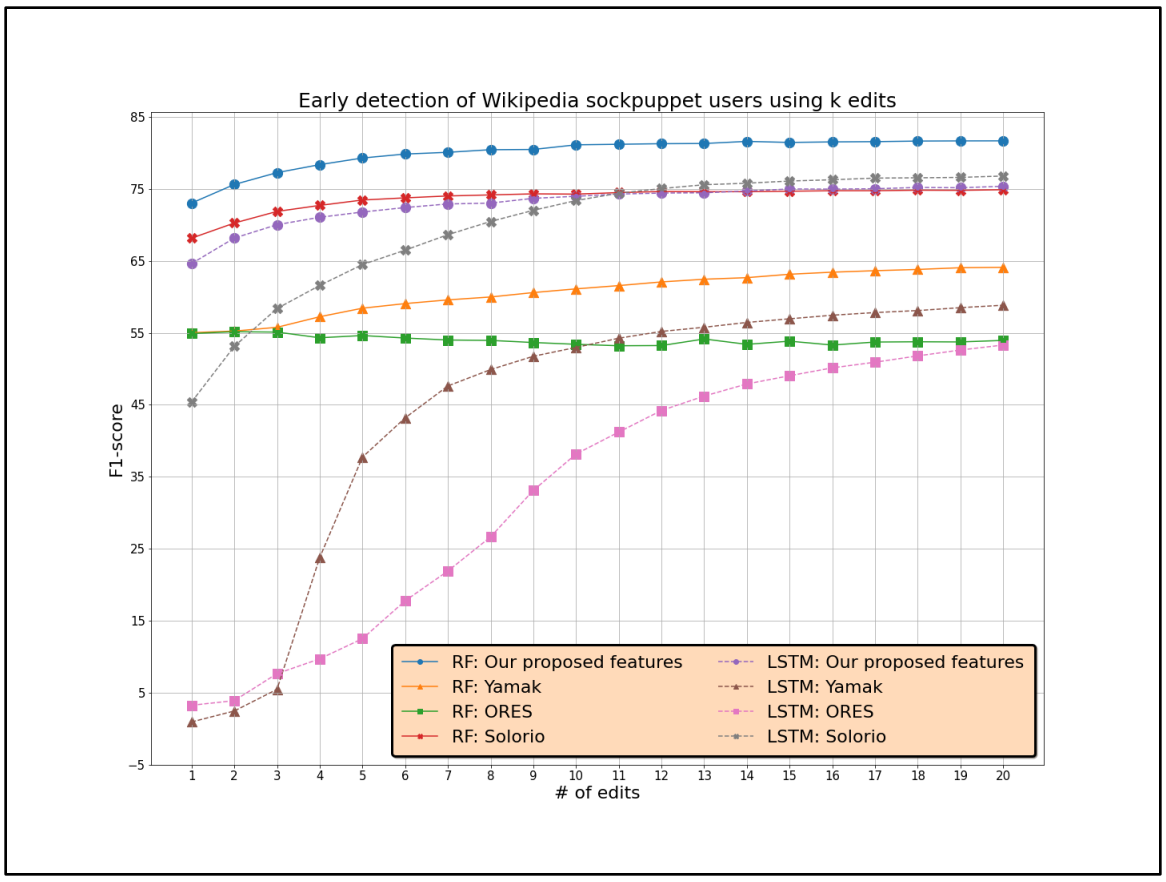

Figure 4.3 Early detection of Wikipedia sockpuppet accounts

4.7 Answer to Research Questions

In this section, we will try to summarize our findings to answer the research questions fixed at the beginning of the study.

RQ1: What are the patterns of sockpuppet accounts created by puppet masters?

Ans: By analyzing the features included in our research, we found that sockpuppet accounts make shorter contributions as compared to benign users (mean average contribution length of 27 vs. 31 characters), and edit pages with longer titles (the mean average title length is 18 for sockpuppets vs. 17 characters for benign users), and edit more frequently (the mean average time difference between two consecutive edits is 3.5 days vs.

17 days for benign users). So overall, puppetmasters' sockpuppet accounts have a distinctive contribution pattern compared to innocent users.

RQ2: Does semantic analysis from edits capture the writing pattern and contribution pages more sophisticatedly and identify the sockpuppet accounts better than syntactic, stylistic, and graph network-based works and bring out a deep level of contextual meaning?

Ans: Our selected semantic analysis from edits captured the writing patterns better than the syntactic, stylistic, and graph network-based works. Our RF-based model performed better than the established method and brought out a deep level of contextual meaning.

RQ3: Is it possible to detect sockpuppet accounts early and recommend suspension?

Ans: Our described approach could early detect sockpuppet accounts by considering the user's first 20 edits and achieved an F1 score of 0.73 by just considering the first edit (vs. a score of 0.68 achieved by the best competitor). So, it is possible to detect sockpuppet accounts right after they start contributing.

CHAPTER FIVE: CONCLUSIONS

5.1 What Have We Done So Far?

In this research, we presented our proposed approach to address the problem of automatically identifying sockpuppet accounts on Wikipedia. We handle the problem as a binary classification task and propose a set of new features to capture suspicious behavior that considers user activity and analyzes the contributed content. Specifically, content-based features have never been considered before and constitute the novelty of our work.

We tested our approach on a dataset we collected containing 17K accounts validated by Wikipedia as sockpuppets. Experimental results show that our proposed method can detect sockpuppet accounts with an F1 score of 0.82 (vs. a score of 0.77 achieved by the best competitor) by considering the user's first 20 edits and 0.73 by just considering the first edit (vs. a score of 0.68 achieved by the best competitor). We also showed that computing the topics of the user contributions is particularly important for detecting these types of malicious accounts. We could also distinguish the generic pattern of sockpuppet users as the mean average contribution length and the mean average time difference between two consecutive edits differed significantly from authentic user accounts. In general, we have seen the importance of semantic level features for sockpuppetry detection compared to other established prior separate approaches. Our analysis also includes extensive early detection of unfaithful accounts to eliminate their contribution in quick times.

5.2 Future Directions

As part of future work, we plan to test our features on predicting whether two accounts belong to the same sockpuppet investigation. Throughout the current work, we have focused on detecting if an account is a sockpuppet or not. To extend such phenomena, we would like to work in the future on evaluating if two accounts are tied under the same investigation.

We are also interested in cross-media platforms. For instance, we will check if the same sockpuppets group exists on both Facebook and Twitter. Such analysis would be fundamental to recognize if abusive users focus on only one platform or carry out similar behavior across any other platform. The study's motivation is to check whether advertisers, spammers, and promoters, irrespective of the social platform, work in a similar pattern or form a group to carry on such heinous activity. An ensemble model capable of combining data from multiple platforms and analyzing sockpuppetry would ensure the holistic improvement of the functionality of tracking numerous account holders.

REFERENCES

[1] K. Luyckx and W. Daelemans, “Authorship attribution and verification with many authors and limited data,” in Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008). Manchester, UK: Coling 2008 Organizing Committee, Aug. 2008, pp. 513–520. [Online]. Available: https://aclanthology.org/C08-1065

[2] K. Luyckx and W. Daelemans, “Personae: a corpus for author and personality prediction from text,” in Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08). Marrakech, Morocco: European Language Resources Association (ELRA), May 2008. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2008/pdf/759paper.pdf

[3] K. Luyckx and W. Daelemans, “The effect of author set size and data size in authorship attribution,” LLC, vol. 26, pp. 35–55, 04 2011.

[4] H. J. Escalante, T. Solorio, and M. Montes-y G´omez, “Local histograms of character n-grams for authorship attribution,” in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 288–298. [Online]. Available: https://aclanthology.org/P11-1030

[5] S. Raghavan, A. Kovashka, and R. Mooney, “Authorship attribution using probabilistic context-free grammars,” in Proceedings of the ACL 2010 Conference Short Papers. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 38–42. [Online]. Available: https://aclanthology.org/P10-2008

[6] V. Keselj, F. Peng, N. Cercone, and C. Thomas, “N-gram-based author profiles for authorship attribution,” 2003.

[7] E. Stamatatos, “Author identification using imbalanced and limited training texts,” in 18th International Workshop on Database and Expert Systems Applications (DEXA 2007), 2007, pp. 237–241.

[8] M. Koppel, J. Schler, and S. Argamon, “Authorship attribution in the wild,” Lang. Resour. Eval., vol. 45, no. 1, p. 83–94, mar 2011. [Online]. Available: https://doi.org/10.1007/s10579-009-9111-2

[9] “Wikipedia,” https://www.britannica.com/topic/Wikipedia, accessed: 2022-07-06.

[10] “Api:main page,” https://www.mediawiki.org/wiki/API:Mainpage, accessed : 2022-07-06.

[11] R. Zheng, J. Li, H. Chen, and Z. Huang, “A framework for authorship identification of online messages: Writing-style features and classification techniques,” Journal of the American Society for Information Science and Technology, vol. 57, no. 3, pp. 378–393, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20316

[12] “Latent Dirichlet Allocation,” https://radimrehurek.com/gensim/models/ldamodel.html, accessed: 2022-07-06.

[13] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, p. 1735–1780, nov 1997. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[14] “Ores,” https://www.mediawiki.org/wiki/ORES, accessed: 2022-07-06.

[15] J. Zhu, Y. Xia, L. Wu, D. He, T. Qin, W. Zhou, H. Li, and T.-Y. Liu, “Incorporating bert into neural machine translation,” arXiv preprint arXiv:2002.06823, 2020.

[16] U. Zaratiana, P. Holat, N. Tomeh, and T. Charnois, “Hierarchical Transformer Model for Scientific Named Entity Recognition,” arXiv e-prints arXiv:2203.14710, Mar. 2022.

[17] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, "Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences," Proceedings of the National Academy of Sciences, vol. 118, no. 15, p. e2016239118, 2021. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2016239118

[18] A. Nambiar, S. Liu, M. Hopkins, M. Heflin, S. Maslov, and A. Ritz, "Transforming the language of life: Transformer neural networks for protein prediction tasks," bioRxiv, 2020. Available: https://www.biorxiv.org/content/early/2020/06/16/2020.06.15.153643

[19] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, X. Chen, J. Canny, P. Abbeel, and Y. S. Song, "Evaluating protein transfer learning with tape," bioRxiv, 2019. [Online]. Available: https://www.biorxiv.org/content/early/2019/06/20/676825

[20] "Wikipedia:purpose," https://en.wikipedia.org/wiki/Wikipedia:Purpose, accessed: 2022-07-06.

[21]"Wikipedia:sockpuppetry,"https://en.wikipedia.org/wiki/Wikipedia:Sockpuppetry, accessed: 2022-07-06.

[22] S. Kumar, F. Spezzano, and V. S. Subrahmanian, "VEWS: A wikipedia vandal early warning system," in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015. ACM, 2015, pp. 607–616.

[23] T. Green and F. Spezzano, "Spam users identification in wikipedia via editing behavior," in Proceedings of the Eleventh International Conference on Web and Social Media, 2017. AAAI Press, 2017, pp. 532–535.

[24] N. Joshi, F. Spezzano, M. Green, and E. Hill, "Detecting undisclosed paid editing in wikipedia," in Proceedings of The Web Conference 2020, 2020, pp. 2899–2905.

[25] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, "An analysis of social network-based sybil defenses," ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 363–374, 2011.

[26] Z. Bu, Z. Xia, and J. Wang, “A sock puppet detection algorithm on virtual spaces,” Knowledge-Based Systems, vol. 37, pp. 366–377, 2013.

[27] D. Liu, Q. Wu, W. Han, and B. Zhou, “Sockpuppet gang detection on social media sites,” Frontiers of Computer Science, vol. 10, no. 1, pp. 124–135, 2016.

[28] S. Kumar, J. Cheng, J. Leskovec, and V. S. Subrahmanian, “An army of me: Sockpuppets in online discussion communities,” in Proceedings of the 26th International Conference on World Wide Web, WWW 2017, 2017, pp. 857–866.

[29] T. Solorio, R. Hasan, and M. Mizan, “A case study of sockpuppet detection in wikipedia,” in Proceedings of the Workshop on Language Analysis in Social Media at NAACL HTL, 2013, pp. 59–68.

[30] T. Solorio, R. Hasan, and M. Mizan, “Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities,” arXiv preprint arXiv:1310.6772, 2013.

[31] Z. Yamak, J. Saunier, and L. Vercouter, “Detection of multiple identity manipulation in collaborative projects,” Proceedings of the 25th International Conference Companion on World Wide Web, 2016.

[32] Z. Yamak, J. Saunier, and L. Vercouter, “Sockscatch: Automatic detection and grouping of sockpuppets in social media,” Knowledge-Based Systems, vol. 149, pp. 124–142, 2018.

[33] M. Tsikerdekis and S. Zeadally, “Multiple account identity deception detection in social media using nonverbal behavior,” IEEE Transactions on Information Forensics and Security, vol. 9, no. 8, pp. 1311–1321, 2014.

[34] X. Zheng, Y. M. Lai, K. Chow, L. C. Hui, and S. Yiu, “Detection of sockpuppets in online discussion forums,” Ph.D. dissertation, University of Hong Kong, 2011.

[35] S. K. Maity, A. Chakraborty, P. Goyal, and A. Mukherjee, “Detection of sockpuppets in social media,” in Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, 2017, pp. 243–246.

[36] S. Adhikari, “Detection of sockpuppet accounts on reddit,” 2020.

[37] R. Zafarani and H. Liu, “10 bits of surprise: Detecting malicious users with minimum information,” in Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, 2015, pp. 423–431.

[38] K. Lee, B. D. Eoff, and J. Caverlee, “Seven months with the devils: A long-term study of content polluters on twitter,” in Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011, L. A. Adamic, R. Baeza-Yates, and S. Counts, Eds. The AAAI Press, 2011.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.