DETECTING UNDISCLOSED PAID EDITING IN WIKIPEDIA

by

Nikesh Joshi

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2020

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Nikesh Joshi

Thesis Title: Detecting Undisclosed Paid Editing in Wikipedia

Date of Final Oral Examination: 24th June 2020

The following individuals read and discussed the thesis submitted by student Nikesh Joshi, and they evaluated the 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
Steven Cutchin, 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.

dedicated to my family

ACKNOWLEDGMENTS

I would like to express my profound gratitude to my advisor, Dr. Francesca Spezzano, for giving me an opportunity to work as a graduate assistant in her research field. I am grateful for her mentorship, valuable advice, and direction with this thesis.

I would also like to thank distinguished members of my thesis committee, Dr. Edoardo Serra and Dr. Steven Cutchin, for their time and guidance.

I would like to thank Smart SE (English Wikipedia administrator) for providing the dataset and contributions of his domain expertise.

Finally, I would like to thank Mr. Aaron Halfaker, Principal Research Scientist at the Wikimedia Foundation, for his help in retrieving ORES scores for Wikipedia articles.

ABSTRACT

Wikipedia is a free and open-collaboration based online encyclopedia. The website has millions of pages that are maintained by thousands of volunteer editors. It is part of Wikipedia's fundamental principles that pages are written with a neutral point of view and are maintained by volunteer editors for free with well-defined guidelines in order to avoid or disclose any conflict of interest. However, there have been several known incidents where editors intentionally violate such guidelines in order to get paid (or even extort money) for maintaining promotional spam articles without disclosing such information.

This thesis addresses for the first time the problem of identifying undisclosed paid articles in Wikipedia. We propose a machine learning-based framework that uses a set of features based on both the content of the articles as well as the patterns of edit history of users who create them. To test our approach, we collected and curated a new dataset from English Wikipedia with ground truth on undisclosed paid articles and a history of users who created those articles. Our experimental evaluation shows that we can identify undisclosed paid articles with an AUROC of 0.98 and an average precision of 0.91. Moreover, our approach outperforms ORES, a scoring system tool currently used by Wikipedia to automatically detect damaging content, in identifying undisclosed paid articles.

We further propose recurrent neural network-based frameworks, that are variants of Long Short-Term Memory (LSTM), using a set of features based on the patterns of edit history of users. Our experimental evaluation also shows that we can identify undisclosed paid editors with an AUROC of 0.93 and an average precision of 0.90 outperforming existing approaches while also outperforming other baseline approaches in early detecting undisclosed paid editors. Finally, we show that our proposed approaches can also be used to address other similar tasks achieving the maximum AUROC score of 0.96, average precision score of 0.97, and accuracy score of 0.90. Also, in this thesis, we show that our approaches are able to outperform other baseline approaches in early detecting both Undisclosed Paid Editors and Wikipedia vandal editors surpassing the performance scores with as little as just two edits.

This thesis is an extension of our work that was published in WWW '20: The Web Conference 2020 held in Taipei, Taiwan in April 2020 [13]. Wikipedia have shown significant interest in our published work and we are currently collaborating for possible deployment of our system directly into their platform.

TABLE OF CONTENTS

ABSTRACT vi

LIST OF TABLES x

LIST OF FIGURES xii

LIST OF ABBREVIATIONS xiv

1 Introduction 1

2 Related Work 5

3 Dataset 10

3.1 Article Network Analysis 12

4 Detecting Undisclosed Paid Articles 18

4.1 Features for Identifying Undisclosed Paid Articles 18

4.1.1 Article-based Features 18

4.1.2 User-based Features 20

4.2 Experimental Results 22

4.2.1 Feature analysis 23

4.2.2 Comparison with ORES 24

4.2.3 Robustness of Model 25

4.2.4 False Positives Analysis 27

5 Early Detection of Undisclosed Paid Editors 30

5.1 Methodology 30

5.1.1 LSTM: Overview 31

5.1.2 LSTM: Loss function 32

5.1.3 LSTM: Loss function for user edit sequence 34

5.1.4 LSTM1: Using hidden layer output from last layer 35

5.1.5 LSTM2: Using all hidden layer outputs 35

5.2 Features for Identifying Undisclosed Paid Editors 37

5.3 Experimental Results 39

5.3.1 Early Detection of Vandal Users 44

6 Conclusions 50

6.1 What have we done so far? 50

6.2 Future directions 51

REFERENCES 53

A Reproducing Experiments 57

A.1 Getting the code 57

A.2 Repository Structure 57

A.3 Getting the data 58

A.4 Pre-processing of data 58

A.5 Running the experiments 60

A.5.1 Detection of Undisclosed Paid Articles 60

A.5.2 Early Detection of Undisclosed Paid Editors 62

LIST OF TABLES

3.1 Size of Positive and Negative Data. Positive data refers to newly created paid articles or known undisclosed paid editors (UPEs). 11

4.1 Performance of our proposed features to detect undisclosed paid articles according to different classification algorithms (best scores highlighted in bold) and comparison and combinations with ORES features (which are article-based) according to AUROC and average precision metrics. 22

4.2 Class averages of each feature. 25

4.3 Analysis of model robustness in detecting undisclosed paid articles. . . . 26

5.1 Performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC and average precision. 41

5.2 Early detection performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC and average precision. 44

5.3 Performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC, average precision, and accuracy 46

5.4 Early detection performance of our user-based features to detect Wikipedia vandal editors and comparison with related work according to AUROC and average precision. 48

LIST OF FIGURES

3.2 Data collection process 12

3.1 Article network: two articles are connected by an edge if they have been edited by a common user. Colors indicate articles create by the same sockpuppet group of undisclosed paid editors (UPEs). Negative articles (in gray) are articles never edited by an UPE. 16

3.3 Standard deviation of unique edited articles per user per sockpuppet investigation. 17

4.1 Top-10 most important features for detecting undisclosed paid articles. 23

4.2 False positive contribution averaged over 10 experimental runs. 28

4.3 False positive contribution averaged over 10 experimental runs. 29

5.1 Visual representation of a single LSTM cell 32

5.2 LSTM gates 33

5.3 LSTM1 overview 36

5.4 Visualization of LSTM1 model architecture 37

5.5 LSTM2 overview 38

5.6 Visualization of LSTM2 model architecture 39

5.7 AUROC and average precision comparison using first 20 user-edits 41

5.8 Comparison of approaches using AUROC 43

5.9 Comparison of approaches using average precision. 43

5.10 Comparison of approaches using AUROC, average precision, and accuracy on UMDWikipedia [16] 46

5.11 Comparison of approaches using AUROC. 48

5.12 Comparison of approaches using average precision. 49

5.13 Comparison of approaches using Accuracy. 49

A.1 Detection of Undisclosed Paid Articles Process. 62

A.2 Early Detection of Undisclosed Paid Editors Process. 64

LIST OF ABBREVIATIONS

AUROC – Area Under the Receiver Operating Characteristics

COI – Conflict of Interest

LSTM – Long Short-Term Memory

ORES – Objective Revision Evaluation Service

RNN – Recurrent Neural Network

UPE – Undisclosed Paid Editors

CHAPTER 1

INTRODUCTION

Wikipedia is the free online encyclopedia based on the principle of open collaboration; for the people by the people. Anyone can add and edit almost any article or page. Voluntary contributors are, however, expected to follow a set of guidelines when editing Wikipedia. The purpose of Wikipedia is “to provide the public with articles that summarize accepted knowledge, written neutrally and sourced reliably” [1] and the encyclopedia should not be considered as a platform for advertising and self-promotion. Wikipedia’s guidelines strongly discourage any form of *conflict-of-interest (COI) editing* and require editors to disclose any COI contribution. *Paid editing* is a form of COI editing and refers to editing Wikipedia (in the majority of the cases for promotional purposes) in exchange for compensation. The guidelines set by Wikipedia are based on good faith, and malicious editors who earn a living through paid editing of Wikipedia choose to ignore the requirement to disclose they are paid. Moreover, these malicious editors often use *sockpuppet accounts* to circumvent a block or a ban imposed on the person’s original account. A sockpuppet is an “online identity used for purposes of deception.” [2] Usually, several sockpuppet accounts are controlled by a unique individual (or entity) called a *puppetmaster*.

The first discovered case of paid editing was the “Wiki-PR editing of Wikipedia”,

[1] `https://en.wikipedia.org/wiki/Wikipedia:Conflict_of_interest`

[2] `https://en.wikipedia.org/wiki/Sockpuppet_(Internet)`

in 2013. [3] Wiki-PR is a company, which still exists but is banned by Wikipedia, whose core business is to offer consulting services to create, edit and monitor "your" Wikipedia page. The 2013 investigation found out that more than 250 sockpuppet accounts were related to and controlled by the company. On August 31, 2015, the Wikipedia community uncovered an even bigger set of 381 sockpuppet accounts, as part of an investigation nicknamed "Orangemoody" [4], operating a secret paid editing ring where participants extorted money from businesses who had articles about themselves rejected. The Orangemoody accounts themselves may have been involved in the deletion of some articles.

When undisclosed paid articles or editors are identified, such pages are removed from Wikipedia, and accounts are blocked. However, the Wikipedia community still relies on administrators who manually track down editors and affected articles. The differences between good faith editing and spam can be hard for even experienced editors to see, and, with hundreds of articles to be examined each month, the review process can be tedious, inefficient, and possibly unreliable.

In this thesis, we focus, for the first time, on automatically detecting Wikipedia undisclosed paid contributions and editors, so that they can be quickly identified and flagged for removal. We make the following contributions. (1) We propose a machine learning-based framework to classify undisclosed paid articles that uses a set of features based on both article content, metadata, and network properties, as well as the patterns of the edit behavior of users who create them. (2) We propose neural network-based frameworks to classify and detect early undisclosed paid editors using a set of features based on the patterns of the edit behavior of users. (3) To

[3] https://en.wikipedia.org/wiki/Wiki-PR_editing_of_Wikipedia

[4] https://en.wikipedia.org/wiki/Orangemoody_editing_of_Wikipedia

test our framework, we built a curated English Wikipedia dataset containing 73.9K edits by undisclosed paid editors (including deleted edits) and 199.2K edits by genuine editors, with ground truth on undisclosed paid articles. (4) Through our experimental evaluation, we show that our proposed machine learning-based method can efficiently identify undisclosed paid articles with an AUROC of 0.98 and an average precision of 0.91. We also show that our approach outperforms ORES [5], the state-of-the-art machine learning service created and maintained by the Wikimedia Scoring Platform team to detect content damage on Wikipedia. (5) Through our experimental evaluation, we also show that our proposed neural network-based methods can efficiently identify undisclosed paid editors with an AUROC of 0.93 and an average precision of 0.90 and easily outperform approaches based on ORES, SAFE [33], and an approach by Yamak et al. [30]. We also show that our approaches outperform other baseline approaches in early detecting undisclosed paid editors, exceeding the performance scores with as little as just two edits. (6) Finally, we demonstrate that our proposed approaches can also be used to address other similar tasks of detecting vandal users in Wikipedia achieving the maximum AUROC score of 0.96, average precision score of 0.97, and accuracy score of 0.90 using the UMDWikipedia [16] dataset curated for the purpose of detecting Wikipedia vandal editors. We show that our approaches are able to outperform other baseline approaches in early detecting Wikipedia vandal editors.

The thesis is organized as follows. Chapter 2 discusses related work associated with our thesis work. Chapter 3 describes the dataset, provided by Wikipedia, used in our research work. Chapter 4 describes our approach and reports experimental results and baseline comparison for detection of undisclosed paid articles. Chapter 5

[5] https://www.mediawiki.org/wiki/ORES

describes our approaches and reports experimental results and baseline comparison for early detection of undisclosed paid editors. Chapter 5 also reports experimental results and baseline comparison when we use our approaches in similar tasks of detecting Wikipedia vandal editors. Finally, we draw our conclusions in Chapter 6. Appendix A provides instruction to repeat and rerun our experiments for the purpose of reproducibility.

CHAPTER 2

RELATED WORK

Past literature has studied different forms of content damage on Wikipedia, including vandalism, hoaxes, and spam. Wikipedia vandalism is "the act of editing the project in a malicious manner that is intentionally disruptive", e.g., through text that is humorous, nonsensical, or offensive. [1] Detecting vandalism was the first problem studied in the context of Wikipedia content deception. Research shows that linguistic, metadata, and user reputation features are all important to detect vandal edits in Wikipedia [20, 29, 2, 1]. Kumar et al. [16] proposed Vandal Early Warning System (VEWS) to address the problem of early detecting vandal users by leveraging editor's behavioral patterns. The literature showed that, on average, VEWS is able to detect vandal users 2.39 edits before ClueBot NG, a state-of-the-art vandal detection tool. VEWS, with its accuracy score of over 0.87 also outperforms less than 0.60 accuracy scores from STiki, another state-of-the-art vandal detection tool.

Kumar et al. [18] studied the characteristics and impact of Wikipedia hoaxes, articles that deceptively present false information as fact. They showed that Wikipedia hoaxes can be detected using features that consider the article structure and content, hyperlink network properties, and the hoaxes' creator reputation. Utilizing such features, Kumar et al. [18] achieved the AUROC score of 0.98 in detecting hoaxes

[1] `https://en.wikipedia.org/wiki/Vandalism_on_Wikipedia`

compared to the AUROC score of just 0.63 with human-based approaches of detecting hoaxes without any specialized tools.

Spam, which is another form of content damage on Wikipedia, refers to the unsolicited promotion of some entities such as external link spamming and advertisements masquerading as articles (as the promotional articles written by undisclosed paid editors). The majority of the work on spam detection on Wikipedia has focused on detecting link spamming via metadata, URL properties, landing site characteristics [28, 27], or spam users using behavioral-based features [8]. Green and Spezzano achieved an AUROC score of 0.842 in detecting Wikipedia spam users using behavior-based features. *To the best of our knowledge, there is no work addressing the problem of detecting promotional Wikipedia articles or undisclosed paid edits.*

Several bots and tools run on Wikipedia to detect vandalism or general damaging edits. ClueBot NG [2] and STiki [3] [29] are designed to detect vandalism. ClueBot NG is a bot that analyzes edit content, scores edits and reverts the worst-scoring edits. STiki is an intelligent routing tool that suggests potential vandalism to humans for definitive classification. It uses metadata and reverts to score edits and computes a reputation score for each user. Currently, Wikimedia ORES[5] is the state-of-the-art approach to classify the quality of Wikipedia articles. Specifically, given an article, ORES evaluates the content of the article according to one of the following classes: spam, vandalism, attack, or OK. Thus, we will compare our proposed approaches to detect undisclosed paid articles with ORES in our thesis work.

As explained in Chapter 1, undisclosed paid editors typically act as a group of sockpuppet accounts. Several literary works have analyzed and detected sockpuppet

[2] `https://en.wikipedia.org/wiki/User:ClueBot_NG`

[3] `https://en.wikipedia.org/wiki/Wikipedia:STiki`

accounts in online social networks and discussion forums [26, 4, 19, 15]. Specific to Wikipedia, Solorio et al. [21, 22] have addressed the problem of detecting whether or not two accounts are maintained by the same user using text authorship identification features. In detecting sockpuppets in Wikipedia, Solorio et al. [22] were able to achieve an accuracy score of 0.68 outperforming baseline systems performance with an accuracy score of 0.53 from a trivial classifier that predicts every case as a sockpuppet (majority) and the accuracy score of just 0.50 from a random baseline (coin toss). Other approaches have focused on classifying sockpuppet vs. genuine accounts by using non-verbal behavior and considering editing patterns [25, 30]. Tsikerdekis and Zeadally [25] were able to achieve an overall accuracy of 0.71 in identifying sockpuppet accounts using non-verbal user activity. Yamak et al. [30] showed that their approach based on the contribution behavior of the users achieves better performances than other works based on the analysis of the contribution text [21, 22] or using non-verbal behavior [25]. Yamak et al. [30] was able to achieve the best overall accuracy score of 0.998 using Random Forest algorithm as compared to the accuracy score of 0.713 using the Non-verbal expectancy Violations Detection approach from Tsikerdekis and Zeadally [25], 0.730 using Adaptive SVM Text Attribute Disagreement Algorithm approach from Solorio et al. [21], and 0.688 from the Natural Language Processing Similarity Searching approach from Solorio et al. [22].

Beyond a multitude of classic machine learning approaches, researchers have also used deep learning architectures in the detection of malicious users. Zheng et al. [34] used Long Short-Term Memory (LSTM), an artificial Recurrent Neural Network architecture, to address the problem of class imbalance in datasets when detecting fraud in online social networks. The literature utilizes LSTM to learn benign user representation in hidden space from user's edit sequences and uses complimentary

Generative Adversarial Network (GAN) model to train and generate complimentary user representation in hidden space. Through its results, Zheng et al. [34] showed that it performs better than nearest neighbor based approach such as the One-class nearest neighbors (OCNN) [23], Gaussian-based approach such as the One-class Gaussian process (OCGP) [14], and classical classifier-based approach such as the One-class SVM (OCSVM) [24]. This work by Zheng et al. [34] shows the applicability of LSTM in learning user's edit sequences in hidden space and predicting vandal users. Zheng et al. [33] addressed the problem of consistent detection of fraudsters in time, i.e., early detection, by incorporating a survival analysis approach with a recurrent neural network (RNN). The literature utilizes user-activity sequences through an RNN and its output at each timestamp to determine the survival probability that is used to make predictions at that timestamp. Through its results, SAFE [33] showed that it performs better than classical classifiers such as Support Vector Machine (SVM), a typical survival model such as the Cox proportional hazard (CPH) model by Cox D. R. [5], and Multi-source LSTM (M-LSTM) by Yuan et al. [31] which is a classification-based fraud early detection model that uses LSTM to capture time-dependent covariates.

While automated and efficient detection of undisclosed paid contributions is paramount in maintaining consumers' faith on Wikipedia as a trusted source of information, it can also result in unintended consequences of unpaid good-faith contributions flagged as undisclosed paid contributions. Halfaker et al. [9] studied the impact of algorithmic tools, utilized by Wikipedia to automatically reject contributions, on retention of new contributors. The study found that automated reversion of contributions by desirable new users reverts amplify the negative effect of rejection on survival, thus driving away genuine contributors threatening the functional existence of open collaboration systems in general. Likewise, false positives when detecting undisclosed paid contribution to Wikipedia by flagging genuine good-faith contributions as undisclosed paid contribution can have similar negative effects in retaining genuine contributors. Therefore, apart from the focus on efficiency and effectiveness, the minimization of false-positive detection is also one of the major challenges.

CHAPTER 3

DATASET

This chapter describes the dataset we used to perform this study. We collaborated with an English Wikipedia administrator [1] active in reviewing articles that may have a conflict of interest (especially paid editing) to collect and curate a dataset of newly created positive articles, created by known undisclosed paid editors, and newly created negative articles, created by genuine users who are not paid editors. We collected the data through the publicly available Wikipedia API. We were able to access currently deleted edits from known undisclosed paid editors, thanks to our administrator's account. Deleted edits are not visible to general users through the Wikipedia API.

To gather the set of positive articles, we started by considering a manually curated set of 1,006 known undisclosed paid editor (UPE) accounts from English Wikipedia, which includes accounts from 23 different sockpuppet investigations [3]. Another set of 98 additional known UPE accounts were manually added by our Wikipedia administrator, resulting in a total of 1,104 UPE accounts. Among the set of new articles created by these UPE accounts, our administrator manually classified 748 of these articles (authored by 330 different editors) as paid articles (positive data). [2]

To collect the set of negative articles, we started by retrieving accounts of users who created a new article (or moved pages created in their user page or draft page

[1] Smart SE, `https://en.wikipedia.org/wiki/User%3ASmartse`

[2] Not all the articles created by a UPE are paid articles as these editors may create "genuine" articles to build a reputation.

Table 3.1: Size of Positive and Negative Data. Positive data refers to newly created paid articles or known undisclosed paid editors (UPEs).

	Positive Data	**Negative Data**
Newly Created Articles	748	6,984
Editors	1,104 (UPEs)	1,557
Total Num. of Edits	73,931	199,172

to the article namespace as some UPEs do) in March 2019 (time of data collection) and who, similarly to UPEs, had made relatively few edits (less than 200 edits) in their account lifetime. 1,557 of these users resulted in being genuine, i.e., they are not known paid editors (or even Wikipedia blocked users [3]), or potentially paid editors as manually verified by our Wikipedia administrator. Then, we considered as the set of negative articles, all the newly created articles by these genuine users. This resulted in 6,984 articles.

For each article in the positive and negative sets, we built a dataset containing the username of the user who created the page, the creation timestamp, the content of the article corresponding to the last edit by the article creator, and computed its size (in bytes). Further, in order to be able to compute features about the article creator account, we collected the time when the account was created and the whole edit history of all the genuine and UPE accounts collected as explained above. For each of these edits (or revisions), we collected timestamp, edited page title, revision ID, revision size, and size difference with respect to the previous version of the edited page. As explained above, deleted edits by UPEs are included in this data, providing us with a complete edit history for the UPE accounts. In total, we collected 73,931 edits by our 1,104 UPEs and 199,172 edits by our 1,557 genuine editors. Table 3.1 summarizes the size of the collected data. Figure 3.2 summarizes the data collection

[3] List of all Wikipedia blocked users: `https://en.wikipedia.org/wiki/Special:BlockList`

process.

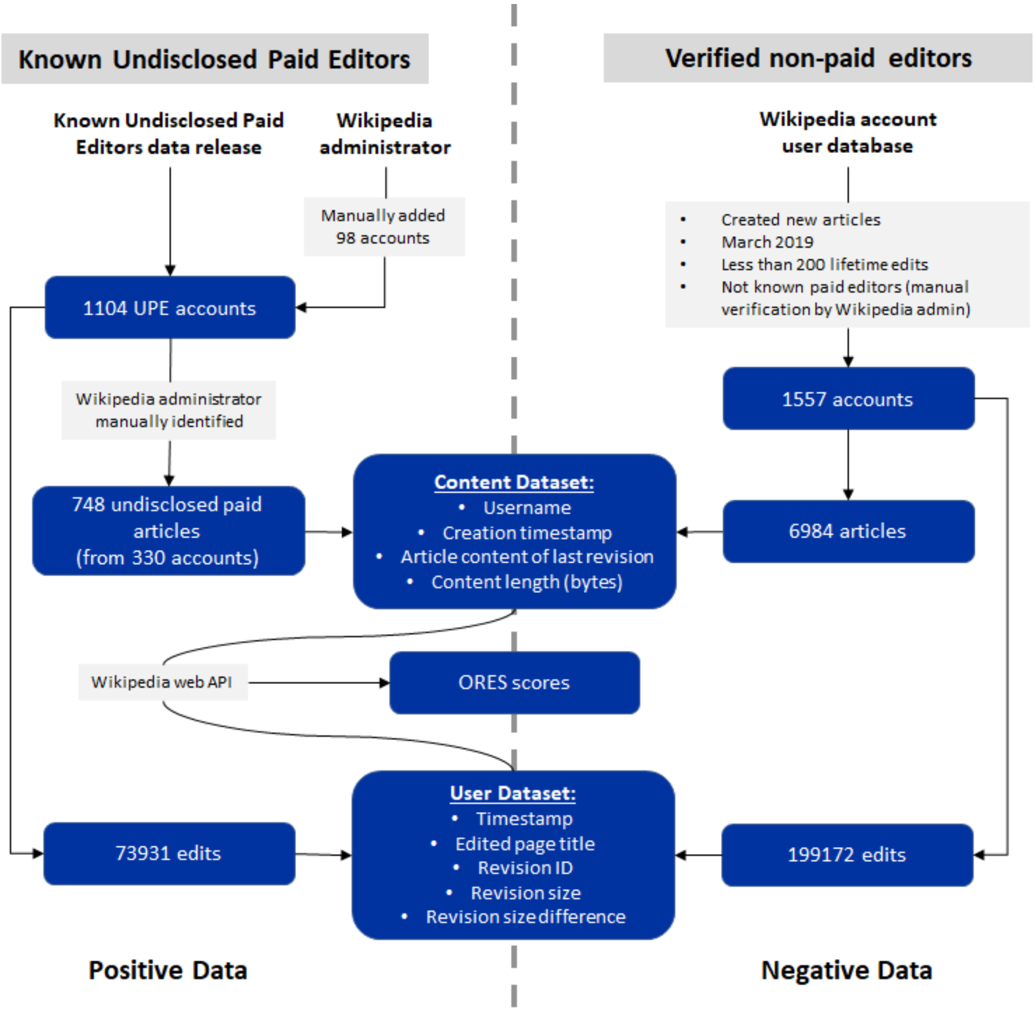
Known Undisclosed Paid Editors
Verified non-paid editors
Known Undisclosed Paid Editors data release
Wikipedia administrator
Manually added 98 accounts
1104 UPE accounts
Wikipedia administrator manually identified
748 undisclosed paid articles (from 330 accounts)
Wikipedia account user database
Created new articles
March 2019
Less than 200 lifetime edits
Not known paid editors (manual verification by Wikipedia admin)
1557 accounts
6984 articles
Content Dataset:
Username
Creation timestamp
Article content of last revision
Content length (bytes)
Wikipedia web API
ORES scores
User Dataset:
Timestamp
Edited page title
Revision ID
Revision size
Revision size difference
73931 edits
199172 edits
Positive Data
Negative Data

Figure 3.2: Data collection process

3.1 Article Network Analysis

We built an article-article network where Wikipedia articles are nodes, and there is an edge between two articles if the same user has edited them. We considered the edit history of all the users in our dataset for creating this network. Figure 3.1 shows the resulting network (93,406 nodes and 44,264,072 edges) where colored nodes represent articles where at least one undisclosed paid editor contributed (referred as positive articles in the rest of this section), and gray ones indicate articles edited only by benign users (referred as negative articles in the rest of this section). Two positive nodes have the same color if the same sockpuppet group has created them. We used the list of sockpuppet investigations in the context of undisclosed paid editing provided by Ballioni et al. [3].

Statistical analysis of datasets gave us some insights into the network structures and shapes of node concentrations based on sockpuppet investigations. We evaluated the total number of unique articles edited by unique users in each socksuppet investigation and used standard deviation to measure the amount of variation of unique article edits in each one of the investigations. Figure 3.3 shows standard deviations of unique article edits per user in each sockpuppet investigations. We found out that the investigations with higher standard deviation of unique articles edits counts per user tend to form connected network shapes with larger nodes concentration. For example, the LogAntiLog investigation has the highest standard deviation among all the investigations and its ”drop” shape has the largest nodes concentration. For the investigations with standard deviations in intermediate range, connected networks of investigations tend to form ”bracket” shapes with sparse node concentrations with small concentration of nodes. For example, Singer_Jethu_Sisodiya/Rudra.shukla and Ventus55/Anatha_Gulati investigation pairs with similar standard deviation values form near identical shapes. As we move from highest to lowest standard deviations of unique article edits per user, connected networks lose concrete form of nodes concentration.

By studying the network, we found that positive articles are less central in the network than negative ones. On average, positive articles have a PageRank of 1.15e-05 (vs. 1.17e-05 for negative ones) and an average local clustering coefficient (LCC) of 0.966 (vs. 0.974 in the case of negative articles). In both cases, the means are different with a p-value ¡ 0.001 according to an independent t-test. This means that there is less user collaboration among positive articles. UPEs only work on a limited number of Wikipedia titles that they are interested in promoting, whereas genuine users edit more pages related to their field of expertise. That results in negative pages being more tightly knit in the network. This result also shows that sockpuppets accounts' behavior in Wikipedia is different from sockpuppetry in online discussion communities where sockpuppets' main goal is to interact with each other to deceive other users, and they have higher PageRank and LCC than benign users [15].

By looking at the articles edited by the same sockpuppet group in Figure 3.1, we observe that, for some investigations, the corresponding pages are more clustered than others. To further understand the meaning of different cluster shapes, for each investigation, we computed the number of unique articles edited by each sockpuppet account and used the standard deviation (SD) to measure the amount of variation of unique edited articles per account in each one of the investigations. Figure 3.3 shows the SD values for each investigation. We found out that the investigations with higher standard deviation tend to form denser clusters with a "drop" shape. This is the case, for instance, of the LogAntiLog investigation that has the highest standard deviation of 697 among all the investigations with fewer sockpuppet accounts contributing to most of the articles. For investigations with standard deviations in the intermediate range, corresponding clusters tend to form "bracket" shapes as UPEs' contributions are more distributed among affected articles. For example, Singer_Jethu_Sisodiya (SD=405)/Rudra.shukla (SD=397) and Ventus55 (SD=49)/Anatha_Gulati (SD=70) investigation pairs have similar standard deviation values and form near-identical shapes. As we move to investigations with a lower standard deviation of unique edited articles per account, clusters start to lose shape (e.g., Orangemoody investigation with SD=12). This analysis suggests that different undisclosed paid editor groups may adopt different editing strategies, which makes the problem of detecting undisclosed paid articles more challenging.

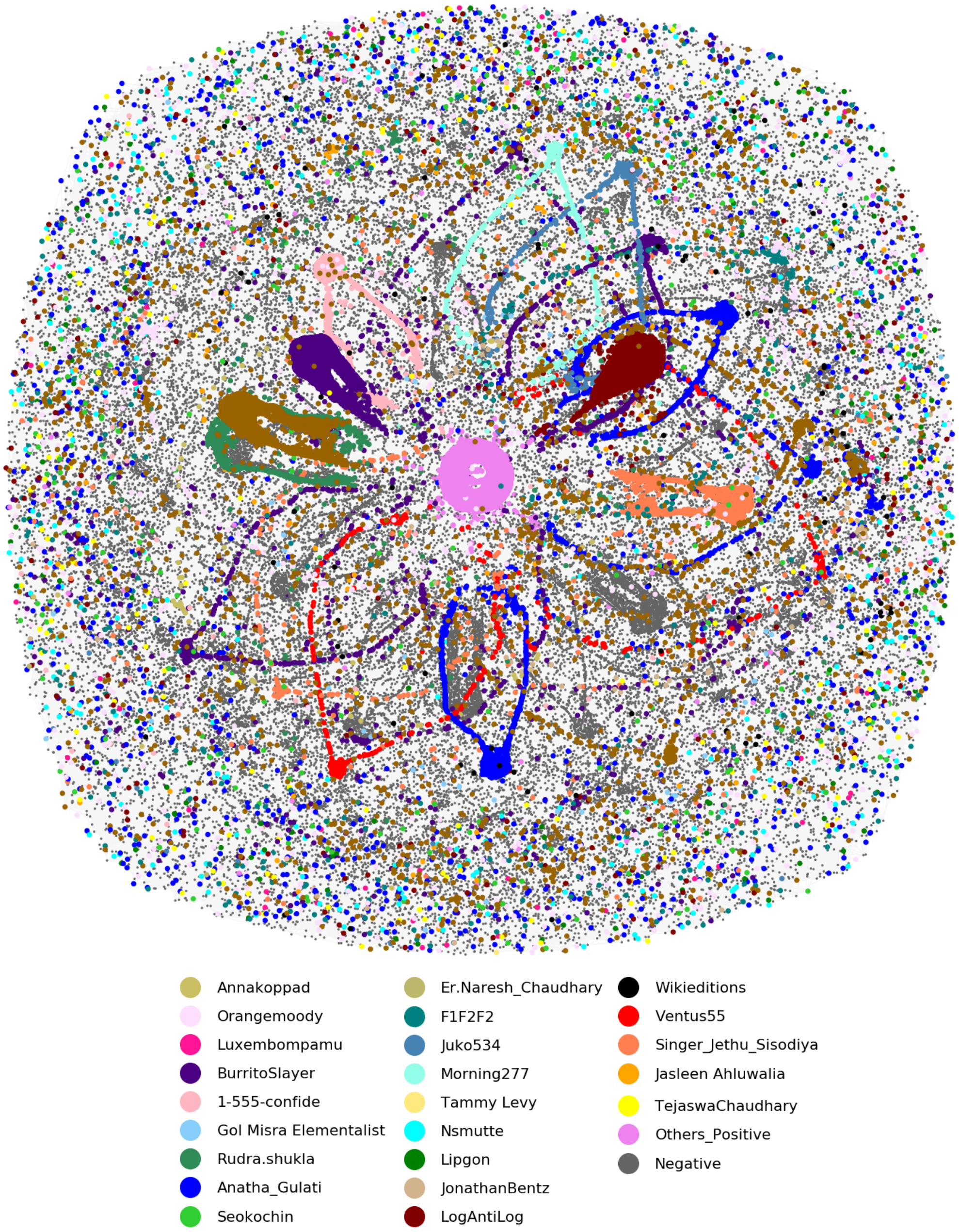

Figure 3.1: Article network: two articles are connected by an edge if they have been edited by a common user. Colors indicate articles create by the same sockpuppet group of undisclosed paid editors (UPEs). Negative articles (in gray) are articles never edited by an UPE.

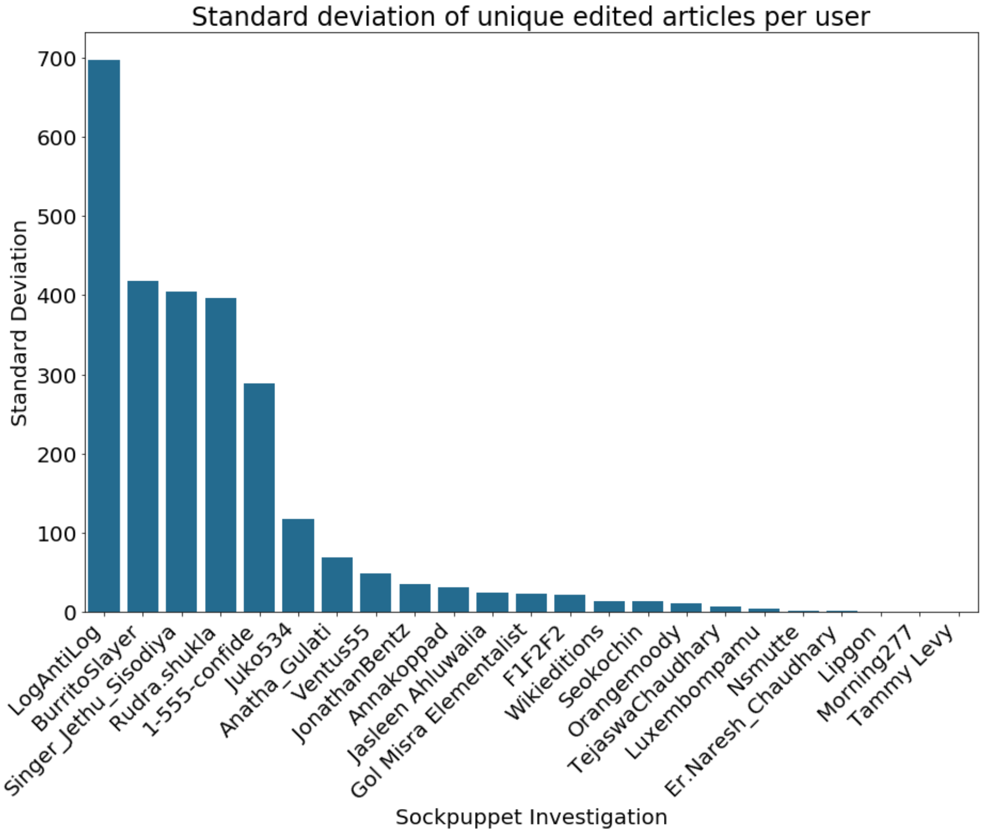

Figure 3.3: Standard deviation of unique edited articles per user per sockpuppet investigation.

CHAPTER 4

DETECTING UNDISCLOSED PAID ARTICLES

In this chapter, we address the problem of identifying undisclosed paid articles in English Wikipedia as a binary classification task. This chapter is organized as follows. Section 4.1 introduces our features to be used with our framework. Section 4.2 discuses experimental setup, results, and baseline comparisons.

4.1 Features for Identifying Undisclosed Paid Articles

In this section, we discuss two sets of features used with our framework based on the properties of articles as well as the edit history of users who created such articles. The features we chose are based on the patterns or behaviors that are more likely to be associated with malicious behavior and paid editing to distinguish them from a benign one. The list of features considered in our approach is described in the following two subsections.

4.1.1 Article-based Features

This first set of features we discuss includes features related to the article, such as metadata, content, and network-based features: *Age of user account at article creation (user_age)* - Since sockpuppet accounts are more likely to be created at the time of the creation of an article, the age of user account at the time of article creation can be considered as one of the features in detecting undisclosed paid articles.

Infobox - This feature checks if the article contains the infobox. The infobox is “a fixed-format table usually added to the top right-hand corner of articles to consistently present a summary of some unifying aspect that the articles share and some time to improve navigation to other interrelated articles.” [1] Undisclosed paid editors tend to add the infobox to the pages they create to increase the exposure of the entity they are promoting as the presence of an infobox is an easy way for humans to grasp a summary of article content.

Number of references - This feature indicates the total number of references (including URL links) present in a given Wikipedia article. Regular Wikipedia articles (especially newly created ones) have a lot of missing references, and researchers have been addressing the problem of suggesting proper references [12]. On the other hand, the purpose of creating undisclosed paid articles is promotional, hence several explicit references to the promoted item are added at the time of page creation. Therefore, a higher number of references in a given article can be a useful indicator of undisclosed paid editing.

Number of photos - This feature refers to the number of photos present in a given Wikipedia article. Uploading images on Wikipedia is relatively complicated as it requires copyright verification. The majority of images added to Wikipedia articles are removed within hours or days of being uploaded because of inappropriate, insufficient, or inaccurate copyright information. Then, to avoid that a promotional article looks suspicious because of its associated images, undisclosed paid articles tend to have fewer images than regular articles.

[1] https://en.wikipedia.org/wiki/Help:Infobox

Number of categories - This feature represents the number of categories associated with a given article. Articles that belong to many categories deal with more complex topics and are less likely to be undisclosed paid articles.

Content length - This feature indicates the total length, in bytes, of the content of the given Wikipedia page. As regular pages are more curated and edited collaboratively by many editors, they tend to have more content and longer in size than undisclosed paid ones.

Network-based features - We also consider the article *PageRank* and *Local Clustering Coefficient (LCC)* as additional features for the article (cf. Section 3.1).

4.1.2 User-based Features

The second group of features we discuss refer to characteristics, such as choice of username and editing behavior, of the user account that created the article. All the features but the username-based ones are computed by considering the history of contributions made by the editor.

Characteristics of usernames can be linked to malicious users that could create undisclosed paid articles [32]. For instance, Green and Spezzano [8] showed that username-based features are important to detect Wikipedia spammers. Thus, we consider the *number of leading digits*, the *number of digits*, the *ratio of digits to characters*, and the *ratio of unique characters* in a username as features indicating a suspicious account.

Average size of added text (avg_size_added) - Given an editor, this feature computes the average size of text added to an article by the editor. Undisclosed paid editors are more likely to create new article content offline and then add it to Wikipedia at once, while benign users edit Wikipedia directly with smaller additions over time.

Average time difference (avg_time_diff) - This feature indicates the average time between two consecutive edits made by the same user. As explained in the above feature, undisclosed paid editors do not regularly edit Wikipedia. They work mainly offline and then add the content whenever they are ready. Thus, we expect the average time difference to be higher for these malicious editors than benign editors.

Ten-byte ratio - This feature computes the percentage of edits made by a user that are less than 10 bytes. Undisclosed paid editors try to become *autoconfirmed* users; thus, they typically make around ten minor edits before creating a promotional article. A registered user account becomes automatically autoconfirmed if the account is more than four days old and has made at least ten changes. Autoconfirmed users are considered benign users that are therefore allowed to move pages to a different title and make changes to pages that have been semi-protected by administrators. The main reason for having autoconfirmed status on Wikipedia is to prevent vandalism and other types of disruptive editing. [2]

Percentage of edits on User or Talk pages (user_talk_edits) - This feature computes the percentage of edits a user has done on a User or Talk page. Undisclosed paid editors may want to edit User or Talk pages for several reasons: they want to have a User page to look like genuine editors; they may draft some content on the article Talk page before moving it to the main article page. Further, the content of an article is discussed by Wikipedia editors on the article Talk page or the contributor's User page. As the contribution of undisclosed paid editors may be disputed by administrators and genuine editors, we expect these malicious editors to engage more in editing these types of pages than genuine editors.

[2] See `https://en.wikipedia.org/wiki/Wikipedia:User_access_levels`

4.2 Experimental Results

We tested our features for the classification task by using three different classification algorithms, namely Logistic Regression, Support Vector Machine (SVM), and Random Forest. We used class weighting to deal with class imbalance in all the classifiers. Class weighting is a way to learn from an unbalanced dataset where the classification imposes, during training, a penalty proportionally inverse to the class distribution on the model for making classification mistakes. To evaluate the performances, we considered the Area Under the Receiver Operating Characteristics curve (AUROC) and the Average Precision metrics, which are well-suited to measure classification results in case of unbalanced data, and performed stratified 5-fold cross-validation.

We were able to achieve the best performance with Random Forest with an AUROC of 0.856 and an average precision of 0.507 using features based on article content, as shown in Table 4.1.

Table 4.1: Performance of our proposed features to detect undisclosed paid articles according to different classification algorithms (best scores highlighted in bold) and comparison and combinations with ORES features (which are article-based) according to AUROC and average precision metrics.

	Article-based Features		User-based Features		Article + User Features	
	AUROC	Average Precision	AUROC	Average Precision	AUROC	Average Precision
Our Features						
Random Forest	**0.856**	**0.507**	0.971	**0.893**	**0.983**	**0.913**
Logistic Regression	0.734	0.230	0.675	0.171	0.656	0.153
Support Vector Machine (SVM)	0.555	0.171	**0.980**	0.823	0.556	0.172
ORES (Random Forest)	0.844	0.424	-	-	-	-
Our Features + ORES (Random Forest)	0.905	0.597	0.974	0.877	0.981	0.907

Figure 4.1 shows the importance of content features in detecting undisclosed paid articles.

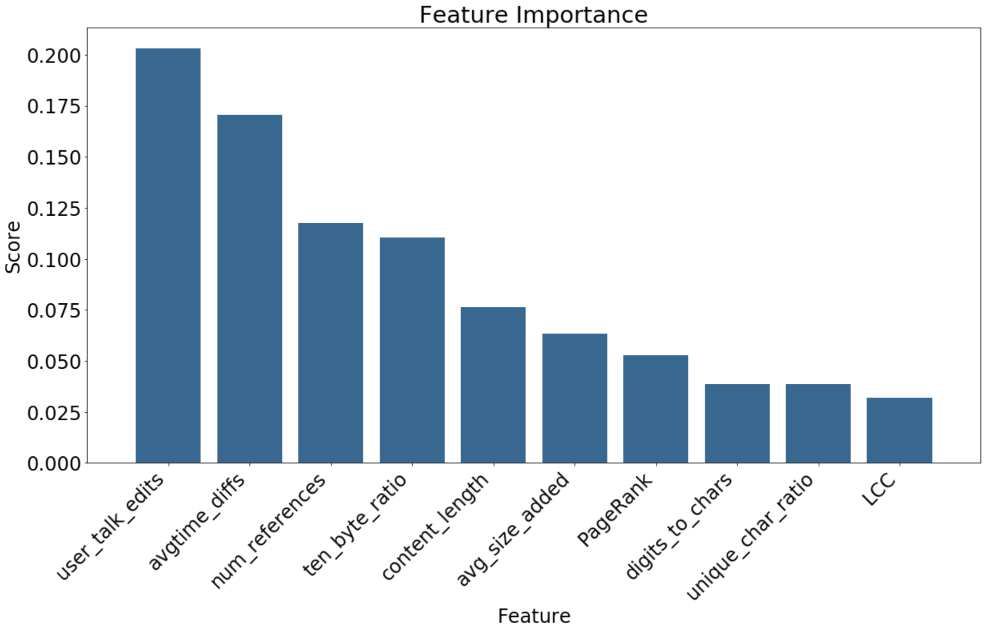

Figure 4.1: Top-10 most important features for detecting undisclosed paid articles.

4.2.1 Feature analysis

To analyze our features, we computed feature importance via a forest of randomized trees. Let F be a set of features. The relative importance (for the classification task) of a feature $f \in F$ is given by the depth of f when it is used as a decision node in a tree. Features used at the top of the tree contribute to the final prediction decision of a larger fraction of the input samples. The expected fraction of the samples they contribute to can thus be used as an estimate of the relative importance of the features. Figure 4.1 shows the importance of our set of features for the undisclosed paid articles classification task. The red bars in the plot show the feature importance using the whole forest. The variability of feature importance scores across the trees in the forest is minimal (less than 0.0001).

Among all the features we defined (article and user-based), the top four most important features are: *percentage of edits on User or Talk pages*, *average time* *difference*, *number of references*, and *ten-byte ratio.* We observe that, on average, the value of the percentage of edits on User or Talk pages feature is higher for positive articles (0.008) than for negative ones (0.0007). These values confirm our hypothesis that users who create undisclosed paid articles are more engaged in editing User and Talk pages than genuine users. Further, we see that, on average, users who create undisclosed paid articles edit more slowly than genuine users: the value of the average time difference feature is 2.8 days for regular articles and 9.2 days for undisclosed paid articles. Also, the percentage of edits that are less than 10 bytes in size is higher for users who created undisclosed paid articles: the value of the ten-byte ratio feature is, on average, 0.38 for positive articles and 0.34 for negative ones. This pattern aligns with the typical behavior of UPEs who make around ten minor edits, then remain quiet for a few days waiting for becoming *autoconfirmed* users (the process takes four days), and then create a promotional article followed by the account going silent [3].

The third most important feature is the number of references in the newly created article. We observe that, on average, positive articles have more references than negative ones: 7.06 vs. 4.88. As explained in Section 4.1.1, this aligns with the fact that regular Wikipedia articles have more missing references than undisclosed paid ones that instead use references to the promoted item.

4.2.2 Comparison with ORES

To compare our preliminary work with ORES, we retrieved the draft quality scores for the positive and negative articles in our dataset by using the ORES publicly available API [3] and used them in input to a classifier to predict undisclosed paid articles. Our preliminary work shows that our approach outperformed ORES, which

[3] `https://ores.wikimedia.org`

Table 4.2: Class averages of each feature.

Feature	Positive Average	Negative Average
user_age	0.439 (days)	0.467 (days)
num_categories	2.06	2.154
infobox	0.57	0.479
num_references	7.064	4.881
num_photos	0.416	0.95
content_length	4455.876	6184.849
user_talk_edits	0.008	0.0007
avgtime_diffs	9.2 (days)	2.8 (days)
num_digits	0.475	0.917
digits_to_chars	0.061	0.173
leading_digits	0.021	0.02
unique_char_ratio	0.815	0.834
ten_byte_ratio	0.38	0.34
avg_size_added	1093.85	1080.372
PageRank	1.15e-05	1.17e-05
LCC	0.966	0.974

achieved an AUROC of 0.844 and average precision of 0.424 in detecting undisclosed paid articles. We further observed that our approach, when combined with ORES features, significantly improved both AUROC (0.905) and average precision (0.597). Table 4.1 shows experimental results from our preliminary work.

4.2.3 Robustness of Model

One of the areas of concern in using a select set of features in detecting undisclosed paid articles is the possibility of such users evading some features and degrading the performance of our model and its ability to detect undisclosed paid articles. We analyzed the stalwart performance of our model, in case the user tries to evade some features, by understanding the robustness of our model. For that, we used a forest of random trees along with feature ablation (removing one of the 16 features at a time and performing the classification with the remaining 15 features). When we exclude our top-1 feature, we notice the largest decrement of the performance of our model. However, even with such drop, our model still achieves an AUROC of 0.962 and Average Precision of 0.834. Table 4.3 shows the performance of our model when we remove one of the 16 features at a time and perform the classification with the remaining 15 features.

Table 4.3: Analysis of model robustness in detecting undisclosed paid articles.

Excluded Feature	AUROC	Average Precision
user_talk_edits	**0.962**	**0.834**
num_references	0.974	0.895
ten_byte_ratio	0.975	0.877
avg_size_added	0.975	0.897
avgtime_diffs	0.976	0.881
PageRank	0.980	0.902
unique_char_ratio	0.980	0.905
num_photos	0.980	0.908
num_digits	0.980	0.905
content_length	0.981	0.908
leading_digits	0.982	0.911
num_categories	0.982	0.910
digits_to_chars	0.982	0.914
infobox	0.982	0.906
user_age	0.983	0.914
LCC	0.984	0.915

4.2.4 False Positives Analysis

Another area of concern when predicting UPEs is the false prediction of genuine unpaid article edits as UPEs. We investigated such false positives from our proposed approach by understanding each feature's contribution to the resulting false positives and feature values when there is a false positive prediction. To investigate the contribution of a feature towards false positive predictions, we performed feature ablation by removing one of the 16 features at a time and performing the classification with the remaining 15 features. By performing feature ablation and subsequent classification, we can identify which feature contributes to the least false positives when excluded from model training and classification. From a multitude of experimental runs, we computed the average of false-positive counts summed up over each stratified sampling fold. We found out that digits_to_chars feature contributes most towards the false positives in identifying UPEs as false positives count when this feature is excluded is the least. Also, user_talk_edits contributes least in false-positive predictions of UPEs as excluding this feature results in the highest number of false-positives. Figure 4.2 shows contribution of each feature towards false-positive prediction of UPEs.

False-positive counts showed us the extent to which each feature contributed to false positives. However, we can also understand the conditions that led to such false positive predictions by investigating feature values and its proximity to class averages. To investigate feature values associated with false-positive predictions, we compared feature values of each sample that resulted in false-positive predictions against positive and negative average feature values and computed the counts of cases when feature value from test sample was closer to positive average feature value from train data than the negative average feature value from the train data. With

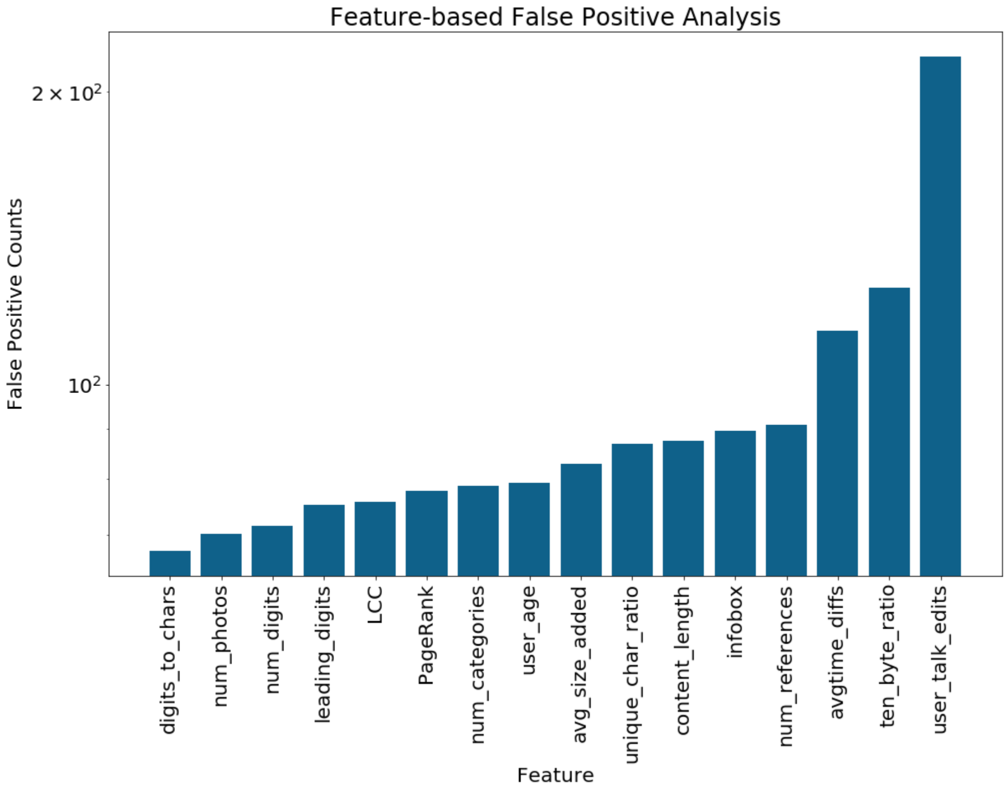

Figure 4.2: False positive contribution averaged over 10 experimental runs.

this determination, we can understand the relation between feature and its value that results with false-positive predictions when test feature values associated with negative ground truth value are closer to average feature value of positive train data. In doing so, we found out that with the false positive predictions, avgtime_diff and digits_to_chars had its feature values closer to the average of positive train data than the average of negative train data. We also notice that the user_talk_edits feature had test values that are mostly closer to negative average train feature values. Figure4.3 shows how often each feature value was closer to positive train average when the model made false-positive predictions.

Based on our false-positive analysis, we can remove our top false-positive con-

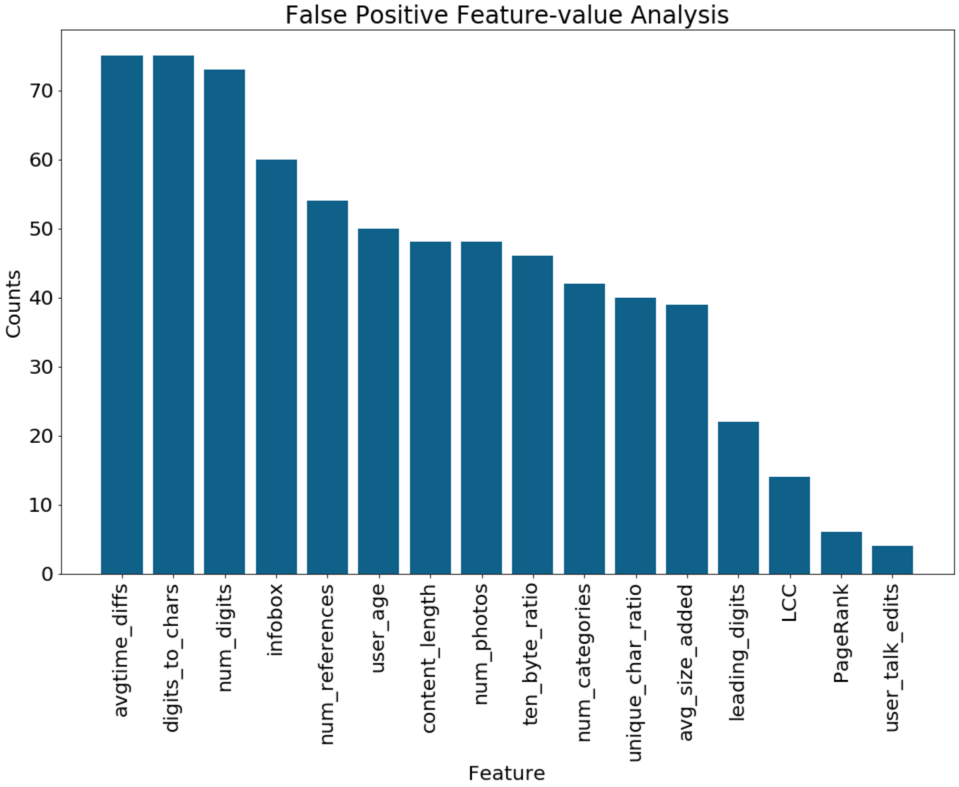

Figure 4.3: False positive contribution averaged over 10 experimental runs.

tributing feature, i.e., digits_to_chars, to reduce false positives from our proposed approach while still maintaining robust performance with 0.982 AUROC and 0.914 average precision as shown in Table 4.3.

CHAPTER 5

EARLY DETECTION OF UNDISCLOSED PAID EDITORS

In this chapter, we address the problem of early detecting undisclosed paid editors in English Wikipedia as a binary classification task using edit history data of the users. This chapter is organized as follows. Section 5.1 introduces our proposed methodology. Section 5.2 discusses user-based features we utilize for the early detection task of undisclosed paid editors. Section 5.3 discuses experimental setup, results, and baseline comparisons. Additionally, Section 5.3.1 discusses results and baseline comparisons using a different dataset used to address the similar task of detecting Wikipedia vandal editors.

5.1 Methodology

In order to achieve our objective of classifying user edit history data and detect Undisclosed Paid Editors, we used Long Short-Term Memory (LSTM), which is a special kind of Recurrent Neural Network capable of learning long-term temporal dependencies, to study the dependability of temporal sequence of user's edit behavior. We made the following considerations in deciding architecture selection:

- The problem of detecting Undisclosed Paid Editors constitute task of classification of user edit history data that are time-series sequence data

- To predict user sequence at any given time-step, we need to learn from its behavior or action from previous time-steps, i.e., accessibility of feedback from previous time-steps at each time-step
- Ability to relay constant flow of feedback without it vanishing or exploding for the long sequences

Even though LSTM itself is a form of Recurrent Neural Network (RNN), unlike RNN, LSTM incorporates input, forget, and output gates (Hochreiter and Schmidhuber [10]) that effectively resolves the issue of vanishing or exploding gradient. Utilizing these gates, LSTM can maintain hidden states of previous time-steps that allows for a constant flow of feedback in sequential data (Gehring et al. [6]). With this, LSTM can decide to remember or forget the feedback in recurrent layers and allow for the learning of long-term dependencies in sequential data (Huang et al. [11]) and thus are more useful in sequence classification. As the user edit history data is a time-series sequence data, LSTM once again makes a better approach for the task of detecting undisclosed paid editors.

With LSTM selected as our preferred architecture for our approach, we now present a general overview of Long Short-Term Memory (LSTM) which forms the basis of our proposed methods. The rest of this section then describes our proposed methods for early detection of Undisclosed Paid Editors.

5.1.1 LSTM: Overview

An LSTM consists of a repeating module of a basic unit called an LSTM cell. Each LSTM cell consists of three gates, namely input, forget, and output gates, to decide retention of memory from previous cell, details to be discarded in current cell, and output of current cell respectively. In each LSTM cell, h_{t-1} is the output of the last

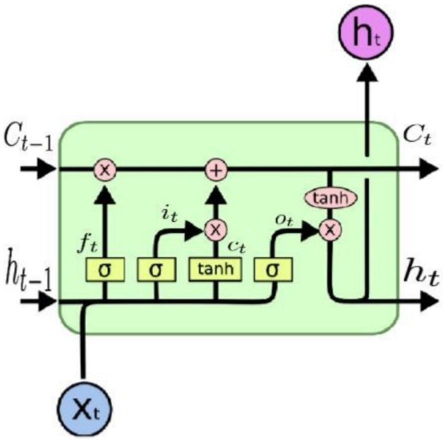

Figure 5.1: Visual representation of a single LSTM cell

LSTM unit, c_{t-1} is the memory from the last LSTM unit, x_t is the current input, c_t is the new updated memory, and h_t is the current output. Input gate uses h_{t-1} and x_t to decide which values from current input should pass through and gives weight to those inputs passing through. Forget gate discards the memory to be discarded and updates current memory c_t. And finally, output gate uses updated memory c_t and input x_t and decides new output h_t. Hence, at any given step, inputs to the LSTM cell are x_t, h_{t-1} and c_{t-1} and outputs from the LSTM cell are h_t and c_t. Figure 5.1 shows components of individual LSTM cells and Figure 5.2 shows three gates of LSTM.

5.1.2 LSTM: Loss function

A model's performance is evaluated by considering how close its predicted values are from the ground truth. Difference between predictions from actual results, which signifies extent by which predicted values deviate from ground truth, is measured as a loss value and function that computes this loss value is called loss function. An LSTM optimizer works by minimizing this loss value in order to improve its performance.

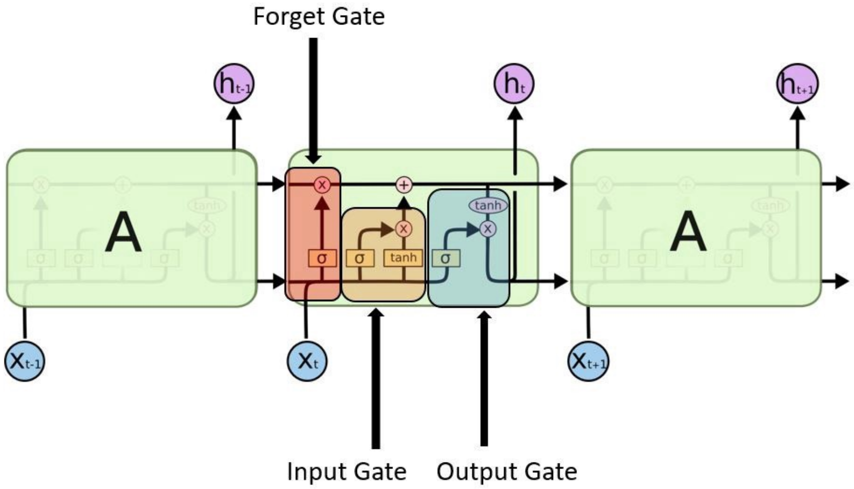

Figure 5.2: LSTM gates

For classification problems, cross entropy loss function is used which is determined as the product of the log of the actual predicted probability for the ground truth class at each time instance as shown in the equation 5.1.

$$loss = -(y_t log(\widehat{y_t}) + (1 - y_t) log(1 - \widehat{y_t})) \tag{5.1}$$

where,

y_t = *actual results or ground truth*

$\widehat{y_t}$ = *probability of predicted class*

Out of two factors used to compute loss, ground truth is fixed. However, we use $\widehat{y_t}$ (probability of predicted class) to define any variations to loss function as required. $\widehat{y_t}$ at time-step t (t $\in (1, .., \ell(u))$ is computed using weight vector of dense LSTM layer $\overline{w}$ and hidden layer activation output h_t as:

$$\widehat{y_t} = W^T.h_t \tag{5.2}$$

where,

W^T *= weight matrix*

h_t *= activation (hidden layer) output at time-step t*

$t = 1, .., \ell(u)$*, i.e. length of edit sequence of user u*

5.1.3 LSTM: Loss function for user edit sequence

For a given user u with edit sequence activity from time-step 1 to t, cross entropy loss function for edit sequence is given by:

$$loss = CE(W^T.h_t) \tag{5.3}$$

where,

W^T *= weight matrix*

h_t *= activation (hidden layer) output at time-step t*

$t = 1, .., \ell(u)$

$\ell(u)$ *= length of edit sequence of user u*

$u \in$ *user in set of users U*

And, cross entropy loss function for edit sequences of entire user set, therefore, is given by taking the sum of cross entropy of all users as shown below:

$$loss = \sum_{u \epsilon U} CE(W^T.h_t) \tag{5.4}$$

where,

$u \in$ *user in set of users U*

W^T *= weight matrix*

h_t *= activation (hidden layer) output at time-step t*

$t = 1, .., \ell(u)$*, i.e. length of edit sequence of user u*

In sections 5.1.4 and 5.1.5, we propose variations of LSTM model architecture based on the way we consider hidden layer outputs for cross entropy loss determinations.

5.1.4 LSTM1: Using hidden layer output from last layer

In our first approach, we use the LSTM1 model, an LSTM model architecture with a many-to-one setup, as shown in the Figure 5.3 where we only use the last hidden layer output at the end of the sequence. In this approach, we use class-specific weighting to deal with class imbalance. The detailed architecture of our LSTM1 model implementation is shown in Figure 5.4. This approach allows the model to consider the entire sequence before classifying a sequence. With such model architecture, our standard cross entropy loss function takes the form as shown in the equation 5.5:

$$loss = \sum_{u \epsilon U} CE(W^T.h_L) \tag{5.5}$$

where,

$u \in$ *user in set of users U*

L *= length of edit sequence of user u*

5.1.5 LSTM2: Using all hidden layer outputs

In our second approach, we use an LSTM model architecture with a many-to-many setup, as shown in figure 5.5, where we use hidden layer output at each time-step

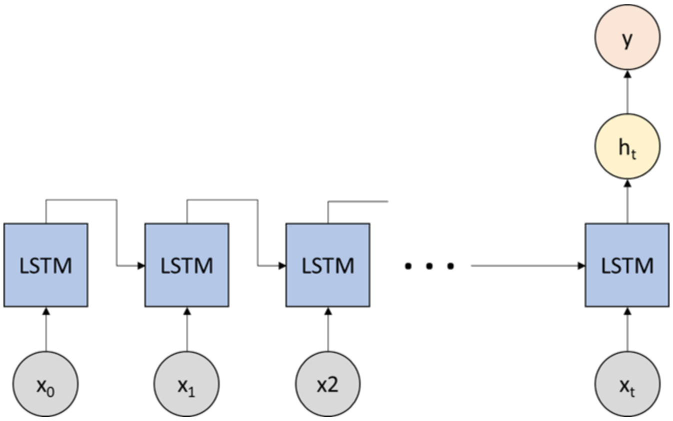

Figure 5.3: LSTM1 overview

of sequences resulting in hidden layer output sequence with the same length as that of the input sequence. The final prediction for the given sequence considers hidden layer output at each and every time-step. In this approach, once again, we use class-specific weighting to deal with class imbalance. However, in LSTM2, we use varying weights for different h-output from different LSTM cells. Such weights vary from highest to lowest as we move from first h-output towards final h-output in sequence. The detailed architecture of our LSTM2 model implementation is shown in Figure 5.6. This approach allows the model to consider the hidden layer outputs at each time-step before classifying a sequence. With such model architecture, our standard cross entropy loss function takes the form as shown in the equation 5.6:

$$loss = \sum_{u \epsilon U} \sum_{t=1}^{\ell(u)} CE(W_t^T . h_t) \tag{5.6}$$

where,

$u \in$ *user in set of users* U

$t =$ *sequence length (1, .., $\ell(u)$)*

$\ell(u)$ = *length of edit sequence of user* u

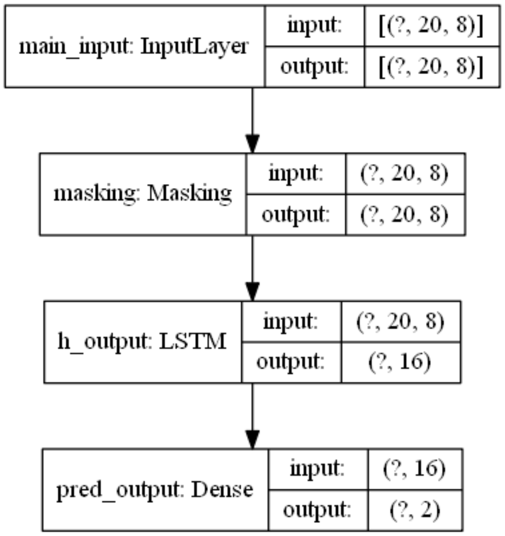

Figure 5.4: Visualization of LSTM1 model architecture

Figures 5.4 and 5.6 show the detailed architectures of LSTM1 and LSTM2 respectively.

5.2 Features for Identifying Undisclosed Paid Editors

With the details of our proposed modeling approaches discussed, we now describe the group of features we use with our models. As we have seen in the previous chapter, the user-based features described in Section 4.1.2 are better than content-based ones in detecting undisclosed paid articles. Thus, in this section, we investigate the effectiveness of these features on the different but related task of early-detection of undisclosed paid editors. The group of features we use to early-detect undisclosed paid editors are similar to the ones we discussed in Section 4.1.2. However, we compute some of the features in different ways that are more compatible with our approach in early-detecting undisclosed paid editors. Each revision in user edit history represents an action or event, and the user's overall edit history represents a sequence of such

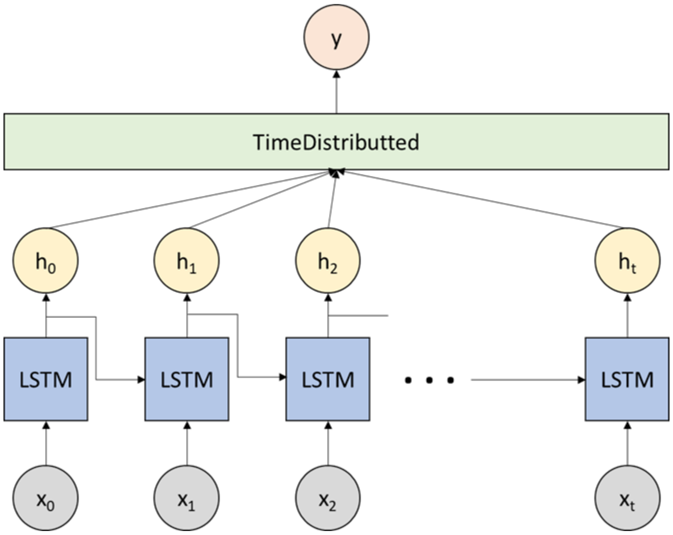

Figure 5.5: LSTM2 overview

actions. As we want our models to learn from each successive action for the task of early-detecting undisclosed paid editors, feature values from each one of users' actions are used without any aggregation.

As discussed in Section 4.1.2, once again, we consider the *number of leading digits*, the *number of digits*, the *ratio of digits to characters*, and the *ratio of unique characters* in username as features indicating a suspicious account.

Size difference (sizediff) - Given an editor, this feature computes the size of any modification or update corresponding to a given revision to an article by the editor. This is the difference between the size of an article before and after the revision is made.

Time difference (time_diff) - This feature indicates the time between two consecutive edits made by the same user.

Under ten bytes - This feature is similar to Ten-byte ratio feature described in Section 4.1.2. However, here we consider this as a binary feature indicating whether an edit made by a user is less than 10 bytes or not.

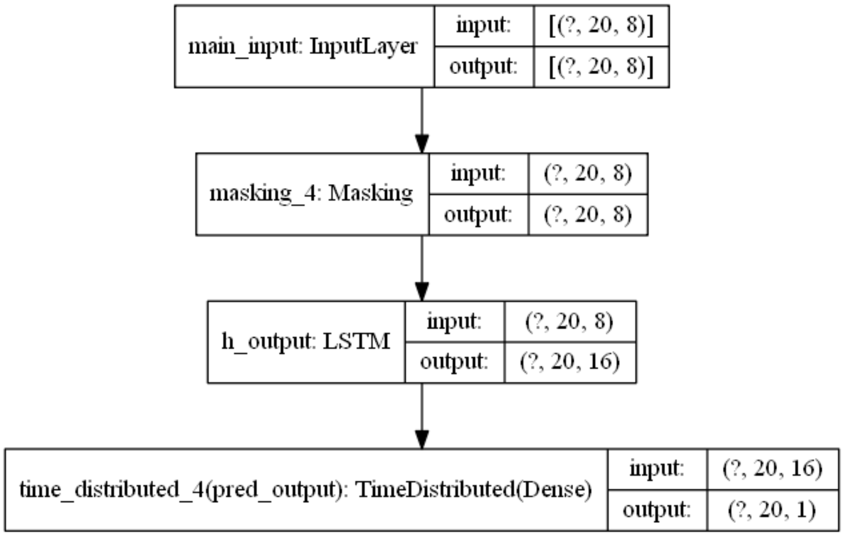

Figure 5.6: Visualization of LSTM2 model architecture

Edits on User or Talk pages (edited) - This feature is similar to Percentage of edits on User or Talk pages feature described in Section 4.1.2. However, for the task of early detecting undisclosed paid editors, we consider this as a binary feature indicating whether an edit made by a user is on User/Talk page or an article.

5.3 Experimental Results

In this section, we look at experimental results for our neural-network-based approaches discussed in Section 5.1. For this, we utilize features described in Section 5.2 and consider only up to the first 20 user-edits from our dataset. In our experiments, both LSTM variants consist of a single LSTM layer with an output space dimension of 16. As our objective is to catch undisclosed paid editors close to the beginning of their action sequence, we consider only up to the first 20 user-edits in our dataset, LSTM cell count, or time-step in both our models are 20 each. As LSTM1 uses hidden outputs from hidden layer output only at the end of the sequence, its output shape is a single hidden output sequence of length that is equal to the output dimension of the LSTM layer. However, LSTM2 uses hidden outputs at each time-step and therefore has LSTM layer output with a shape of (20, 16), i.e., twenty different hidden output sequences of length that are equal to the number of output dimensions in an LSTM layer. As reported in Table 3.1, we have 1,104 UPEs and 1,557 benign users in our dataset.

Experiment1: Using first 20 user-edits. In our first experiment, we used two of our approaches, LSTM1 (Section 5.1.4) and LSTM2 (Section 5.1.5) to investigate capabilities and performance of our proposed models in detecting Undisclosed Paid Editors. Results are reported in Table 5.1 that show our approaches achieving AUROC scores of 0.926 and 0.901 for LSTM1 and LSTM2, respectively. Similarly, the average precision for LSTM1 and LSTM2 is 0.897 and 0.845, respectively. When comparing our approaches individually, the results also show that LSTM1 performed better than LSTM2 both in terms of AUROC and average precision scores.

Comparison with related work. In order to evaluate experimental results based on our LSTM1 and LSTM2 approaches, we once again considered work by Yamak et al. [30] and Zheng et al. [33] (SAFE) discussed in Section 2. Even though SAFE makes baseline comparisons with Support Vector Machine (SVM), Cox proportional hazard (CPH) model and Multi-source LSTM (M-LSTM) described in Section 2, we limit comparison of our experimental results with only the best performing approach in work by Zheng et al. [33], i.e., SAFE. We also compared our experimental results from LSTM1 and LSTM2 with those using ORES scores, discussed in Section 2. As we can see from Table 5.1, our LSTM1 and LSTM2 approaches to detect Undisclosed Paid Editors has a comparable AUROC score but achieves a better average precision (+0.031) than those based on Yamak et al.'s [30] approach. The table also shows that our LSTM1 and LSTM2 approaches achieve far better AUROC scores and average precision when compared with the same using ORES scores as features. And finally, AUROC scores and average precision from both of our approaches (LSTM1 and LSTM2) easily outperform corresponding scores using an approach based on the SAFE. Figure 5.7 shows a comparison of AUROC and average precision using the first 20 user-edits.

Table 5.1: Performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC and average precision.

	AUROC	Average Precision
LSTM1	**0.926**	**0.897**
LSTM2	0.901	0.845
Yamak et al. [30]	0.914	0.866
ORES	0.724	0.632
SAFE [33]	0.575	0.525

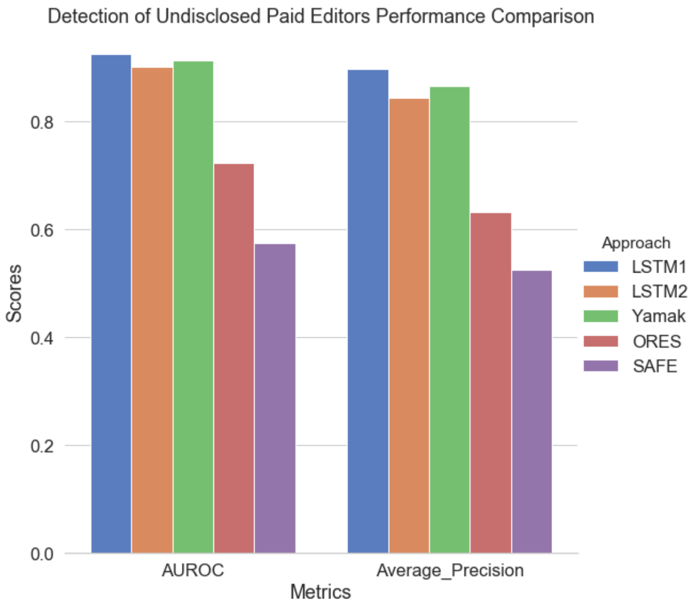

Figure 5.7: AUROC and average precision comparison using first 20 user-edits

Experiment2: Using first K user-edits. In our second experiment, we again use two of our approaches, LSTM1 (Section 5.1.4) and LSTM2 (Section 5.1.5), in order to investigate capabilities and performance of our proposed models in detecting Undisclosed Paid Editors using first K user-edits for each user in our dataset where K ranges from 1 to 10. Results are reported in Table 5.2 that shows our approaches achieving AUROC scores of 0.877 and 0.896 for LSTM1 and LSTM2 respectively when up to 10 user-edits are considered. The table also shows average precision of 0.830 and 0.819 for LSTM1 and LSTM2 respectively.

Comparison with related work. Once again, we considered work by Yamak et al. [30] and the SAFE approach by Zheng et al. [33] discussed in Sections 2, and experimental results from our approach using ORES scores, discussed in Section 2, as features in order to evaluate our experimental results. As we can see from Table 5.2, our approaches (LSTM1 and LSTM2) to early detect Undisclosed Paid Editors easily outperform both AUROC and average precision scores using approaches based on the study by Yamak et al. [30], SAFE, and scores based on experimental results based on ORES scores as features using as little as two edits. Figure 5.8 and 5.9 give visual demonstration of superior performances of our model as compared to the those based on related works. When our approaches are compared individually, the results also show that LSTM2 performed better than LSTM1 when we consider first K edits. Overall, our experimental results show that the LSTM2 is the best approach in the early detection of undisclosed paid editors in terms of both AUROC and average precision scores.

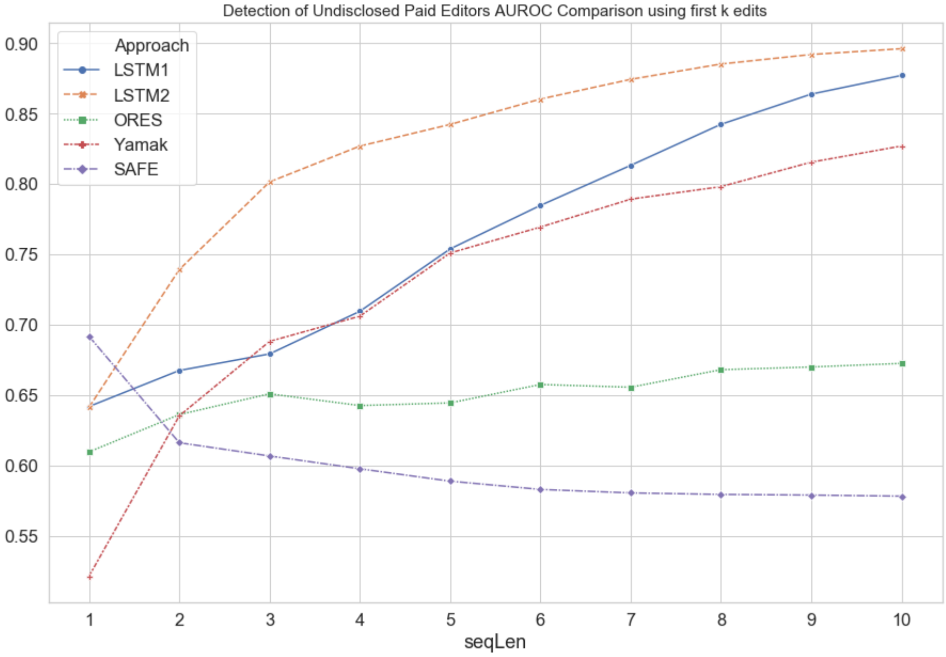

Figure 5.8: Comparison of approaches using AUROC

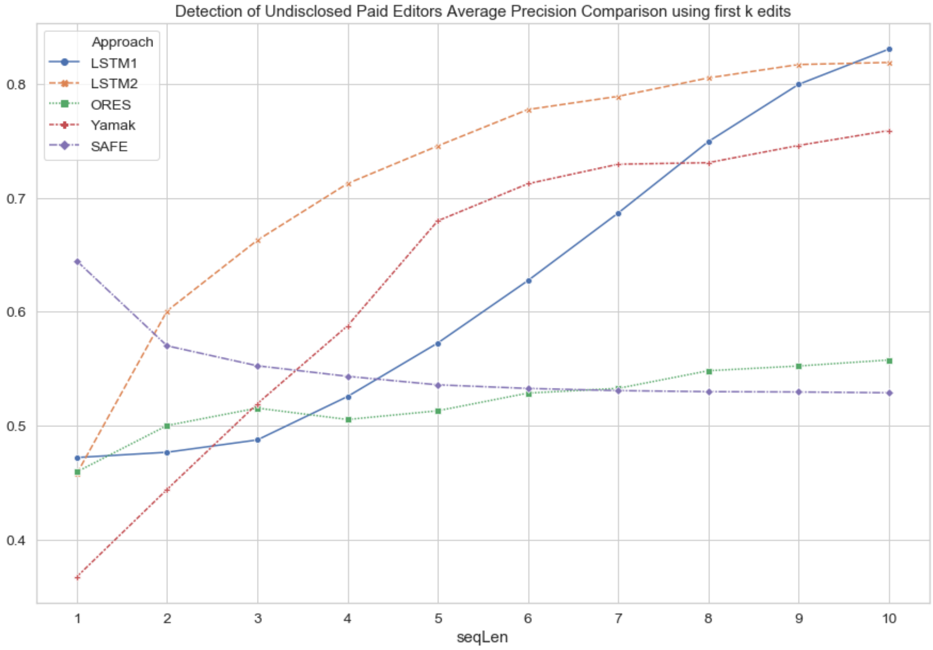

Figure 5.9: Comparison of approaches using average precision.

Table 5.2: Early detection performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC and average precision.

K	Approach	AUROC	Average Precision	K	Approach	AUROC	Average Precision
1	LSTM1	0.642	0.472	**6**	LSTM1	0.785	0.627
	LSTM2	0.641	0.458		LSTM2	0.860	0.777
	ORES	0.610	0.460		ORES	0.657	0.529
	Yamak [30]	0.521	0.368		Yamak [30]	0.769	0.712
	SAFE [33]	0.692	0.645		SAFE [33]	0.583	0.533
2	LSTM1	0.667	0.477	**7**	LSTM1	0.813	0.687
	LSTM2	0.739	0.601		LSTM2	0.874	0.789
	ORES	0.636	0.500		ORES	0.655	0.533
	Yamak [30]	0.635	0.444		Yamak [30]	0.789	0.729
	SAFE [33]	0.616	0.570		SAFE [33]	0.580	0.531
3	LSTM1	0.679	0.488	**8**	LSTM1	0.842	0.749
	LSTM2	0.801	0.663		LSTM2	0.885	0.805
	ORES	0.651	0.515		ORES	0.668	0.548
	Yamak [30]	0.688	0.519		Yamak [30]	0.798	0.731
	SAFE [33]	0.607	0.553		SAFE [33]	0.579	0.530
4	LSTM1	0.709	0.526	**9**	LSTM1	0.864	0.799
	LSTM2	0.827	0.712		LSTM2	0.892	0.817
	ORES	0.642	0.505		ORES	0.670	0.552
	Yamak [30]	0.706	0.588		Yamak [30]	0.815	0.746
	SAFE [33]	0.598	0.543		SAFE [33]	0.579	0.530
5	LSTM1	0.754	0.573	**10**	LSTM1	**0.877**	**0.830**
	LSTM2	0.842	0.745		LSTM2	0.896	0.819
	ORES	0.644	0.513		ORES	0.672	0.558
	Yamak [30]	0.751	0.680		Yamak [30]	0.827	0.759
	SAFE [33]	0.589	0.536		SAFE [33]	0.578	0.529

5.3.1 Early Detection of Vandal Users

In the previous section, we evaluated the experimental results based on our approaches (LSTM1 and LSTM2) to demonstrate its superiority in early detecting Undisclosed Paid Editors compared to results from approaches based on related work. However, we also evaluated the performance of our approaches when tested on a similar task of early detecting Wikipedia vandal editors and, once again, made a comparison of results from approaches based on related work. For that, we used the UMD-

Wikipedia [16] dataset, collected for the task of detecting Wikipedia vandal editors. The dataset consists of 17,027 users that were blocked by Wikipedia for vandalism and 16,549 randomly selected benign users collected between January 01, 2013, and July 31, 2014. The dataset also consists of edit meta-data comprised of user, page-type (normal or meta-page), title of page edited, time of edit, and category for the total of 770,040 edits out of which 160,651 edits were made by vandals, and 609,389 were made by benign users. The dataset provides edit sequences for each user based on characteristics of consecutive edits actions, such as speed, frequency, page-type, etc., that are encoded with unique string representation with a maximum sequence length of 500 edits for the total of 33,576 users. For our experimental setup, we considered edit sequences with up to 20 (inclusive) edits. As the UMDWikipedia [16] dataset is balanced, we were also included an accuracy metric, along with AUROC and average precision metrics, in comparing our results.

Experiment3: Using first 20 user-edits. First, we used two of our approaches, LSTM1 (Section 5.1.4) and LSTM2 (Section 5.1.5), in order to investigate capabilities and performance of our proposed models in detecting Wikipedia vandal users using UMDWikipedia [16] dataset. Results are reported in Table 5.3 that shows our approaches achieving AUROC scores of 0.951 and 0.955, average precision of 0.969 and 0.972, and accuracy scores of 0.880 and 0.889 for LSTM1 and LSTM2 respectively.

Comparison with related work In order to evaluate experimental results based on our LSTM1 and LSTM2 approaches, once again, we considered the SAFE approach by Zheng et al. [33] (discussed in Section 2) using the UMDWikipedia [16] dataset. As we can see from Table 5.3, LSTM1 achieved an AUROC score of 0.951, i.e. +0.376, average precision of 0.969, i.e., +0.444, and accuracy score of 0.880, i.e. +0.240, as compared to corresponding metrics based on the SAFE approach. Similarly, LSTM2 achieved an AUROC score of 0.955, i.e. +0.380, average precision of 0.972, i.e. +0.447, and accuracy score of 0.889, i.e. +0.249, as compared to corresponding metrics based on the approach by SAFE [33]. The results show that LSTM2 was the best performing approach. Both LSTM1 and LSTM2 are comparable to one another and perform far better than SAFE [33]. Figure 5.10 shows a comparison of our approaches with related work according to AUROC and average precision.

Table 5.3: Performance of our user-based features to detect undisclosed paid editors and comparison with related work according to AUROC, average precision, and accuracy

	AUROC	Average Precision	Accuracy
LSTM1	0.951	0.969	0.880
LSTM2	**0.955**	**0.972**	**0.889**
SAFE [33]	0.575	0.525	0.640

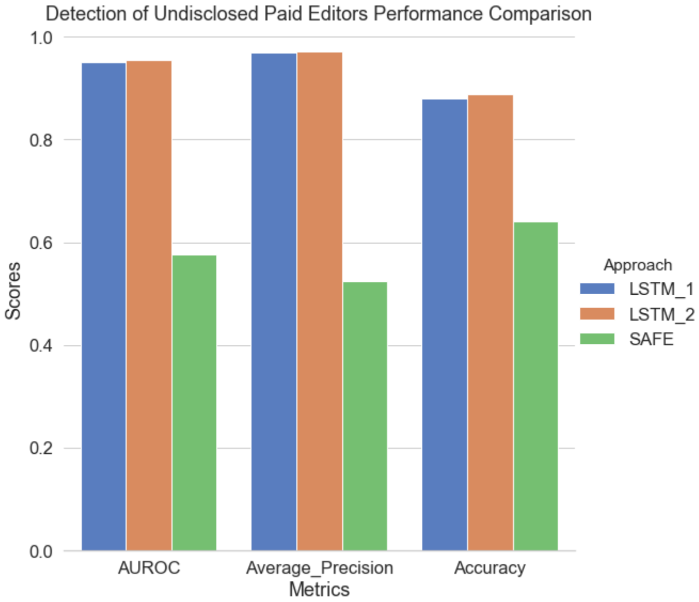

Figure 5.10: Comparison of approaches using AUROC, average precision, and accuracy on UMDWikipedia [16]

Experiment4: Using first K user-edits. Finally, we used two of our approaches, LSTM1 (Section 5.1.4) and LSTM2 (Section 5.1.5), in order to investigate capabilities and performance of our proposed models in early detection of Wikipedia vandal users using the UMDWikipedia [16] dataset. Results are reported in Table 5.4 that shows our approaches achieving AUROC scores of 0.948 and 0.952 for LSTM1 and LSTM2 respectively, when the first up to ten user-edits are considered. The table also shows an average precision of 0.891 and 0.935 and accuracy scores of 0.878 and 0.886 for LSTM1 and LSTM2, respectively. When we compare our two approaches, we can see that the LSTM2 performed slightly better than LSTM1. In terms of evaluation metrics improvement with LSTM2, compared to those with LSTM1, accuracy score achieved the biggest improvement while average precision score achieved the smallest improvement.

Comparison with related work. Once again, we considered SAFE [33], discussed in Section 2, to evaluate our approaches using the UMDWikipedia [16]. As we can see from Table 5.4, our two approaches to detect Undisclosed Paid Editors, LSTM1 and LSTM2, easily outperform the results from experiments using an approach based on study by SAFE [33] in all three metrics (AUROC, average precision, and accuracy scores) using as little as two edits. Figures 5.11 through 5.13 shows the obvious superiority of our approaches highlighting the fact that LSTM1 and LSTM2 curves for all three metrics are significantly above those based on the approach by Zheng et al. [33].

Table 5.4: Early detection performance of our user-based features to detect Wikipedia vandal editors and comparison with related work according to AUROC and average precision.

K	**Approach**	**AUROC**	**Average Precision**	**Accuracy**
1	LSTM1	0.796	0.579	0.579
	LSTM2	0.795	0.759	0.768
	SAFE	0.692	0.645	0.661
2	LSTM1	0.859	0.776	0.807
	LSTM2	0.886	0.842	0.845
	SAFE	0.616	0.570	0.675
3	LSTM1	0.899	0.834	0.850
	LSTM2	0.911	0.877	0.862
	SAFE	0.607	0.553	0.667
4	LSTM1	0.920	0.856	0.862
	LSTM2	0.929	0.899	0.871
	SAFE	0.598	0.543	0.661
5	LSTM1	0.930	0.871	0.870
	LSTM2	0.938	0.914	0.878
	SAFE	0.589	0.536	0.661
6	LSTM1	0.939	0.878	0.873
	LSTM2	0.944	0.922	0.880
	SAFE	0.583	0.533	0.651
7	LSTM1	0.944	0.883	0.874
	LSTM2	0.948	0.927	0.883
	SAFE	0.580	0.531	0.651
8	LSTM1	0.946	0.886	0.876
	LSTM2	0.950	0.929	0.884
	SAFE	0.579	0.530	0.651
9	LSTM1	0.947	0.889	0.877
	LSTM2	0.951	0.934	0.886
	SAFE	0.579	0.530	0.651
10	LSTM1	0.948	0.891	0.878
	LSTM2	**0.952**	**0.935**	**0.886**
	SAFE	0.578	0.529	0.651

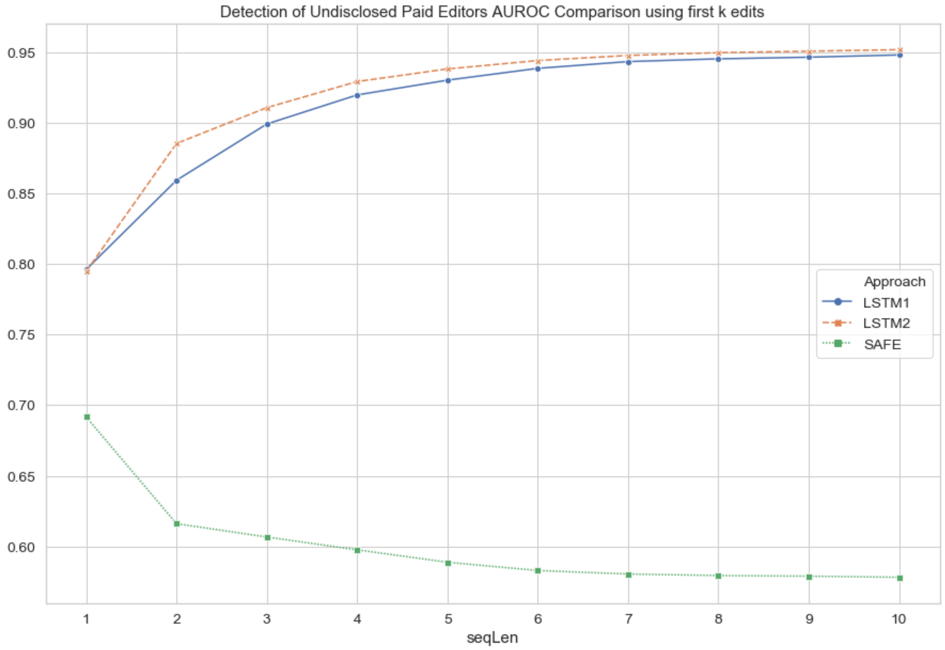

Figure 5.11: Comparison of approaches using AUROC.

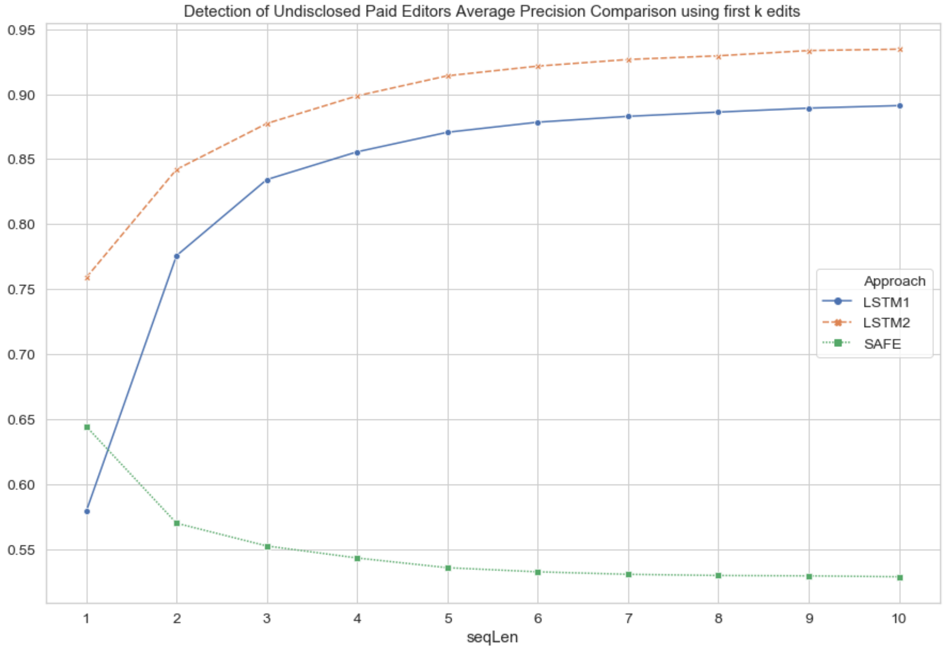

Figure 5.12: Comparison of approaches using average precision.

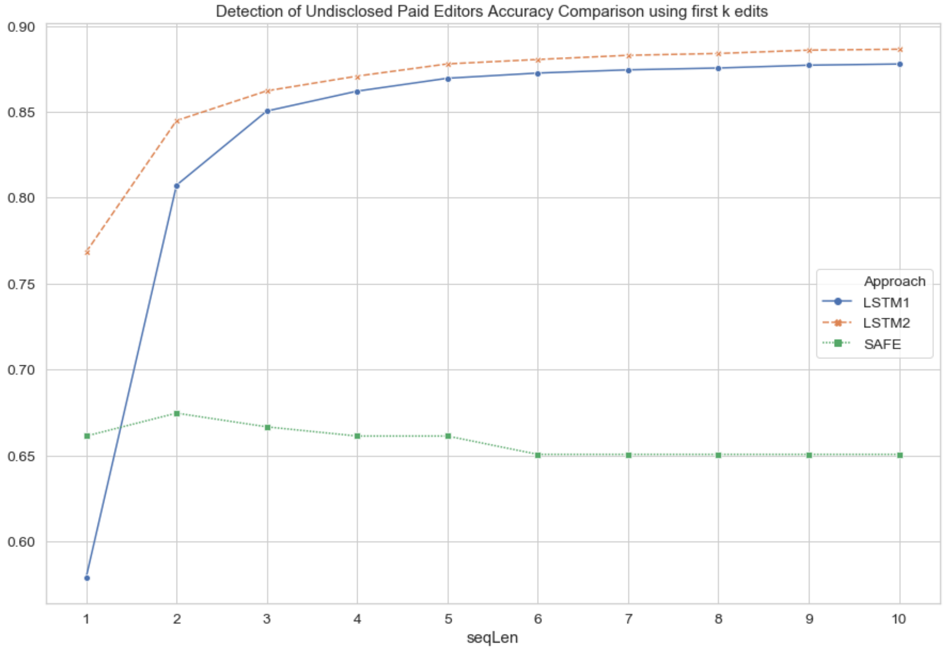

Figure 5.13: Comparison of approaches using Accuracy.

CHAPTER 6

CONCLUSIONS

6.1 What have we done so far?

In this thesis, we addressed the problem of identifying undisclosed paid articles and editors in English Wikipedia. Our proposed approach relies on article-based and user-based features that describe potential malicious behavior. Through evaluation of our approaches, we demonstrated the following:

- We showed that we can detect undisclosed paid articles with an AUROC of 0.98 and an average precision of 0.91. As our features are independent of linguistic barriers, our proposed approach can work on any Wikipedia language version.

- Through our article network analysis, we highlighted the complexity in investigation of editing behavior of sockpuppet groups. We showed that undisclosed paid editors contribute to a limited number of Wikipedia titles, possibly focused on promotional articles, and the number of editors contributing to such articles can vary from one undisclosed paid editors groups (sockpuppets) to another.

- We showed that our user-based features can be used to identify undisclosed paid editors with 0.93 AUROC and 0.90 average precision. Our results in detecting undisclosed paid articles and editors improve over state-of-the-art approaches.

- We showed that our proposed LSTM models are well capable of early-detecting undisclosed paid editors with an AUROC of 0.88 and .90 and average precision of 0.83 and 0.82 respectively for LSTM1 and LSTM2. Furthermore, we also showed that our LSTM approaches are well capable of early-detecting undisclosed paid editors with our LSTM2 approach outperforming performance based on the related works using as little as two edits.

- We demonstrated the reliability of our LSTM approaches in addressing other similar tasks such as early detection of vandals in Wikipedia using the UMD-Wikipedia [16] dataset set up to study the problem of detecting vandals in Wikipedia. We showed that our proposed LSTM approaches can be used to identify vandals with up to 0.96 AUROC and 0.97 average precision using our best performing approach (LSTM2) far surpassing the performance of a similar approach by SAFE [33].

6.2 Future directions

Our proposed approaches can be further improved and extended in the future. Wikipedia has shown significant interest in our published work with the possibility of deployment of our system directly into their platform. For that reason, we are now collaborating with one of the teams from Wikipedia that will provide us with access to additional user edit history data. With these additional data, we can improve our working dataset to further improve the model-training and its performance in detecting undisclosed paid editors. Our approaches can also be extended in the future for the study of sockpuppets in general. In the future, it is also possible to include content analysis of Wikipedia edits and investigate the socio-behavioral aspect of editors. One such aspect that Wikipedia is interested in investigating is the study of user interactions to detect detrimental behavior such as harassment. Finally, our thesis work can also be extended beyond the study of early detection of undisclosed paid editors to address the problem of early detection of sockpuppets in general.

REFERENCES

[1] B. Thomas Adler, Luca de Alfaro, Santiago Moisés Mola-Velasco, Paolo Rosso, and Andrew G. West. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part II*, pages 277–288, 2011.

[2] B. Thomas Adler, Luca de Alfaro, and Ian Pye. Detecting wikipedia vandalism using wikitrust - lab report for PAN at CLEF 2010. In *CLEF 2010 LABs and Workshops, Notebook Papers, 22-23 September 2010, Padua, Italy*, 2010.

[3] Tony Ballioni, James Heilman, Brian Henry, and Aaron Halfaker. Known Undisclosed Paid Editors (English Wikipedia). 4 2018.

[4] Zhan Bu, Zhengyou Xia, and Jiandong Wang. A sock puppet detection algorithm on virtual spaces. *Knowledge-Based Systems*, 37:366–377, 2013.

[5] David R Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

[6] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org, 2017.

[7] GitHub. https://github.com/.

[8] Thomas Green and Francesca Spezzano. Spam users identification in wikipedia via editing behavior. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 532–535, 2017.

[9] Aaron Halfaker, R Stuart Geiger, Jonathan T Morgan, and John Riedl. The rise and decline of an open collaboration system: How wikipedia's reaction to popularity is causing its decline. *American Behavioral Scientist*, 57(5):664–688, 2013.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Minlie Huang, Yujie Cao, and Chao Dong. Modeling rich contexts for sentiment classification with lstm. *arXiv preprint arXiv:1605.01478*, 2016.

[12] Abhik Jana, Pranjal Kanojiya, Pawan Goyal, and Animesh Mukherjee. Wikiref: Wikilinks as a route to recommending appropriate references for scientific wikipedia pages. *arXiv preprint arXiv:1806.04092*, 2018.

[13] Nikesh Joshi, Francesca Spezzano, Mayson Green, and Elijah Hill. Detecting undisclosed paid editing in wikipedia. In *Proceedings of The Web Conference 2020*, pages 2899–2905, 2020.

[14] Michael Kemmler, Erik Rodner, Esther-Sabrina Wacker, and Joachim Denzler. One-class classification with gaussian processes. *Pattern recognition*, 46(12):3507–3518, 2013.

[15] Srijan Kumar, Justin Cheng, Jure 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, Perth, Australia, April 3-7, 2017*, pages 857–866, 2017.

[16] Srijan Kumar, Francesca 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, Sydney, NSW, Australia, August 10-13, 2015*, pages 607–616, 2015.

[17] Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. Vews: A wikipedia vandal early warning system. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 607–616, 2015.

[18] Srijan Kumar, Robert West, and Jure Leskovec. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 591–602, 2016.

[19] Dong Liu, Quanyuan Wu, Weihong Han, and Bin Zhou. Sockpuppet gang detection on social media sites. *Frontiers of Computer Science*, 10(1):124–135, 2016.

[20] Martin Potthast, Benno Stein, and Robert Gerling. Automatic vandalism detection in wikipedia. In *Advances in Information Retrieval , 30th European*

Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings, pages 663–668. 2008.

[21] Thamar Solorio, Ragib Hasan, and Mainul Mizan. A case study of sockpuppet detection in wikipedia. In *Proceedings of the Workshop on Language Analysis in Social Media at NAACL HTL*, pages 59–68, 2013.

[22] Thamar Solorio, Ragib Hasan, and Mainul Mizan. Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities. *arXiv preprint arXiv:1310.6772*, 2013.

[23] David MJ Tax and Robert PW Duin. Uniform object generation for optimizing one-class classifiers. *Journal of machine learning research*, 2(Dec):155–173, 2001.

[24] David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

[25] Michail Tsikerdekis and Sherali Zeadally. Multiple account identity deception detection in social media using nonverbal behavior. *IEEE Transactions on Information Forensics and Security*, 9(8):1311–1321, 2014.

[26] Bimal Viswanath, Ansley Post, Krishna P Gummadi, and Alan Mislove. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review*, 41(4):363–374, 2011.

[27] Andrew G. West, Avantika Agrawal, Phillip Baker, Brittney Exline, and Insup Lee. Autonomous link spam detection in purely collaborative environments. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration, 2011, Mountain View, CA, USA, October 3-5, 2011*, pages 91–100, 2011.

[28] Andrew G. West, Jian Chang, Krishna K. Venkatasubramanian, Oleg Sokolsky, and Insup Lee. Link spamming wikipedia for profit. In *The 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference, CEAS 2011, Perth, Australia, September 1-2, 2011, Proceedings*, pages 152–161, 2011.

[29] Andrew G. West, Sampath Kannan, and Insup Lee. Detecting wikipedia vandalism via spatio-temporal analysis of revision metadata? In *Proceedings of the Third European Workshop on System Security, EUROSEC 2010, Paris, France, April 13, 2010*, pages 22–28, 2010.

[30] Zaher Yamak, Julien Saunier, and Laurent Vercouter. Detection of multiple identity manipulation in collaborative projects. In *Proceedings of the 25th International Conference Companion on World Wide Web (Companion)*, pages 955–960, 2016.

[31] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Yang Xiang. Wikipedia vandal early detection: from user behavior to user embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 832–846. Springer, 2017.

[32] Reza Zafarani and Huan 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, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 423–431, 2015.

[33] Panpan Zheng, Shuhan Yuan, and Xintao Wu. Safe: A neural survival analysis model for fraud early detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1278–1285, 2019.

[34] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1286–1293, 2019.

APPENDIX A

REPRODUCING EXPERIMENTS

A.1 Getting the code

The code can be downloaded from GitHub [7] using the URL below. The repository will be made public after publication of this thesis. However, dataset will not be available given the restricted nature of our dataset provided to us by Wikipedia.

URL: `https://github.com/nikeshjoshi/DetectingUPEinWikipedia.git`

We used Jupyter Notebook to implement our thesis work.

A.2 Repository Structure

The repository is structured in same was as the directory structure used in running all of our experiments as shown below:

1. ./graphOutput/ stores graph output
2. ./nbDetectUPArticles/ includes notebooks required for detection of undisclosed paid articles experiment
3. ./nbEarlyDetectUPEditors/ includes notebooks required for early detection of undisclosed paid editors experiments
4. ./nbRetrieveORES/ includes notebooks required to retrieve ORES scores

5. ./nbSAFE/ includes notebook required for baseline experiment using SAFE
6. ./nbThesisResults/ includes notebooks to run all the early detection of undisclosed paid editors experiments using single notebook and tabulate and plot all the experimental results together
7. ./pickles/ stores pickle output
8. ./plots/ stores plot outputs
9. ./wikiData/ includes data files provided by Wikipedia administrator. Note: Two files in this folder have been added as zip file because of GitHub size limit. To re-run the experiments, those files must be unzipped.
10. ./vewsData/ includes data files used by Kumar et al. [17] that we use for baseline comparison

A.3 Getting the data

Additional data files that exceed size limit on GitHub repository are privately stored on google drive and will not be publicly available given the restricted nature of our dataset provided to us by Wikipedia. Authorized users can access these data using google drive link (**Additional Data Link**).

To re-run the experiments, these data folders must be placed in same directory that contains other folders listed in Section A.2.

A.4 Pre-processing of data

Wikipedia provided us their data in three different pairs of data files in binary form.

- Content data
  - positive_content (inc short)
  - negative_content
- User registration data
  - positive_user_registration
  - negative_user_registration
- Edit history (contribution) data
  - positive_contrib_full_v2
  - negative_contrib_full_v2

Beside these data files that were provided by Wikipedia, we also generate secondary data files that are used in our notebooks that run our experiments.

- positive_content_with_ORES.csv
- negative_content_with_ORES.csv
- content_data_with_ORES.csv
- user_data_without_ORES.csv
- graph.pkl
- user_and_content_data_with_ORES.csv
- posAll_Data_ORES_Scores.csv
- negAll_Data_ORES_Scores.csv
- userEdits.csv

Due to the restricted nature of Wikipedia dataset, these data will not be publicly available from GitHub [7] repository.

A.5 Running the experiments

In this section, we discuss procedures for running our all of our experiments including the ones for Detection of Undisclosed Paid Articles and Early Detection of Undisclosed Paid Editors.

A.5.1 Detection of Undisclosed Paid Articles

Follow hierarchical procedure can be followed for complete re-run of experiments for detection of undisclosed paid articles by running Jupyter notebooks in order as listed below. Figure A.1 shows procedural flow diagram for detection of undisclosed paid articles.

1. *RetrieveORESScores_AllContents.ipynb* - Retrieve ORES scores for each articles in Wikipedia content data.
   - Outputs:
     (a) positive_content_with_ORES.csv
     (b) negative_content_with_ORES.csv
2. *dupa_ParseContentData.ipynb* - Parse content data.
   - Outputs:
     (a) content_data_with_ORES.csv
3. *dupa_ParseUserData.ipynb* - Parse user data.
   - Outputs:
     (a) user_data_without_ORES.csv

4. *dupa_Generate_Graph.ipynb* - Generate article-article network graph (where Wikipedia articles are nodes, and there is an edge between two articles if the same user has edited them) to extract article *PageRank* and *Local Clustering Coefficient (LCC)*.

   - Outputs:

     (a) graph.pkl

5. *dupa_runme.ipynb* - Merge content data with user data.

   - Outputs:

     (a) user_and_content_data_with_ORES.csv

6. *dupa_DetectUndisclosedPaidArticles.ipynb* - Train and test model.

   - This notebook generates experiment results for detection of undisclosed paid articles. We can also directly rerun this notebook using pre-existing user_and_content_data_with_ORES.csv and graph.pkl files to reproduce experiment results.

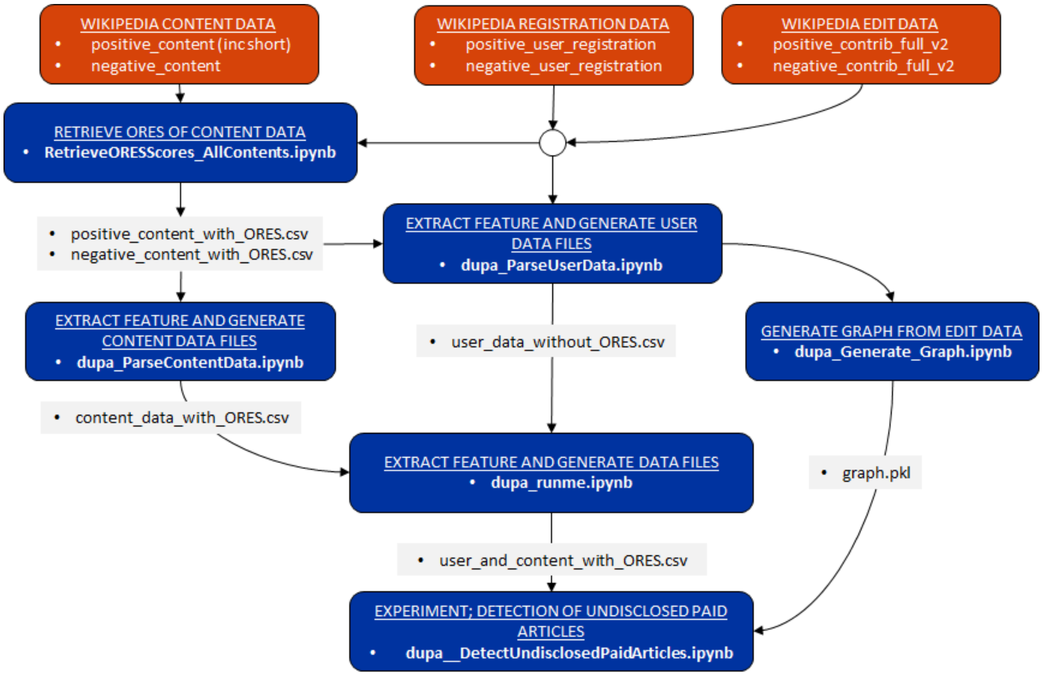
WIKIPEDIA CONTENT DATA
positive_content (inc short)
negative_content
WIKIPEDIA REGISTRATION DATA
positive_user_registration
negative_user_registration
WIKIPEDIA EDIT DATA
positive_contrib_full_v2
negative_contrib_full_v2
RETRIEVE ORES OF CONTENT DATA
RetrieveORESScores_AllContents.ipynb
positive_content_with_ORES.csv
negative_content_with_ORES.csv
EXTRACT FEATURE AND GENERATE USER DATA FILES
dupa_ParseUserData.ipynb
EXTRACT FEATURE AND GENERATE CONTENT DATA FILES
dupa_ParseContentData.ipynb
user_data_without_ORES.csv
GENERATE GRAPH FROM EDIT DATA
dupa_Generate_Graph.ipynb
content_data_with_ORES.csv
EXTRACT FEATURE AND GENERATE DATA FILES
dupa_runme.ipynb
graph.pkl
user_and_content_with_ORES.csv
EXPERIMENT; DETECTION OF UNDISCLOSED PAID ARTICLES
dupa__DetectUndisclosedPaidArticles.ipynb

Figure A.1: Detection of Undisclosed Paid Articles Process.

A.5.2 Early Detection of Undisclosed Paid Editors

Follow hierarchical procedure can be followed for complete re-run of experiments for detection of undisclosed paid articles by running Jupyter notebooks in order as listed below. Figure A.2 shows procedural flow diagram for detection of undisclosed paid articles.

1. *RetrieveORESScores_AllEdits.ipynb* - Retrieve ORES scores for each edits in Wikipedia contribution data.

   - Outputs:

     (a) posAll_Data_ORES_Scores.csv

     (b) negAll_Data_ORES_Scores.csv

2. *ParseUserData.ipynb* - Parse user edit data.

- Outputs:

    (a) userEdits.csv

3. These notebooks generate experiment results for early detection of undisclosed paid editors with corresponding approach and dataset. Run following notebooks (order not important):

    (a) *lstm1_max20_overall_and_firstKedits.ipynb* - Run experiment using LSTM1 (discussed in Section 5.1.4).

    (b) *lstm2_max20_overall_and_firstKedits.ipynb* - Run experiment using LSTM2 (discussed in Section 5.1.5).

    (c) *Yamak_max20_overall_and_firstKedits.ipynb* - Run experiment using Yamak et al. [30] approach.

    (d) *vewsData_lstm1_max20_overall_and_firstKedits.ipynb* - Run experiment using LSTM1 (discussed in Section 5.1.4) approach with UMDWikipedia [16] dataset.

    (e) *vewsData_lstm2_max20_overall_and_firstKedits.ipynb* - Run experiment using LSTM2 (discussed in Section 5.1.5) approach with UMDWikipedia [16] dataset.

    (f) *safe_max20_overall_and_firstKedits.ipynb* - Run experiment using SAFE [33] approach.

4. Alternative method:

    (a) *earlyDetectUPE_runAll.ipynb* - Run all early detection of undisclosed paid editors experiments.

5. Generate results and plots:

   (a) *earlyDetectUPE_Plots_and_Tables_only.ipynb* - Once all early detection of undisclosed paid editors experiments have been executed, this notebook tabulates all the results and plots together.

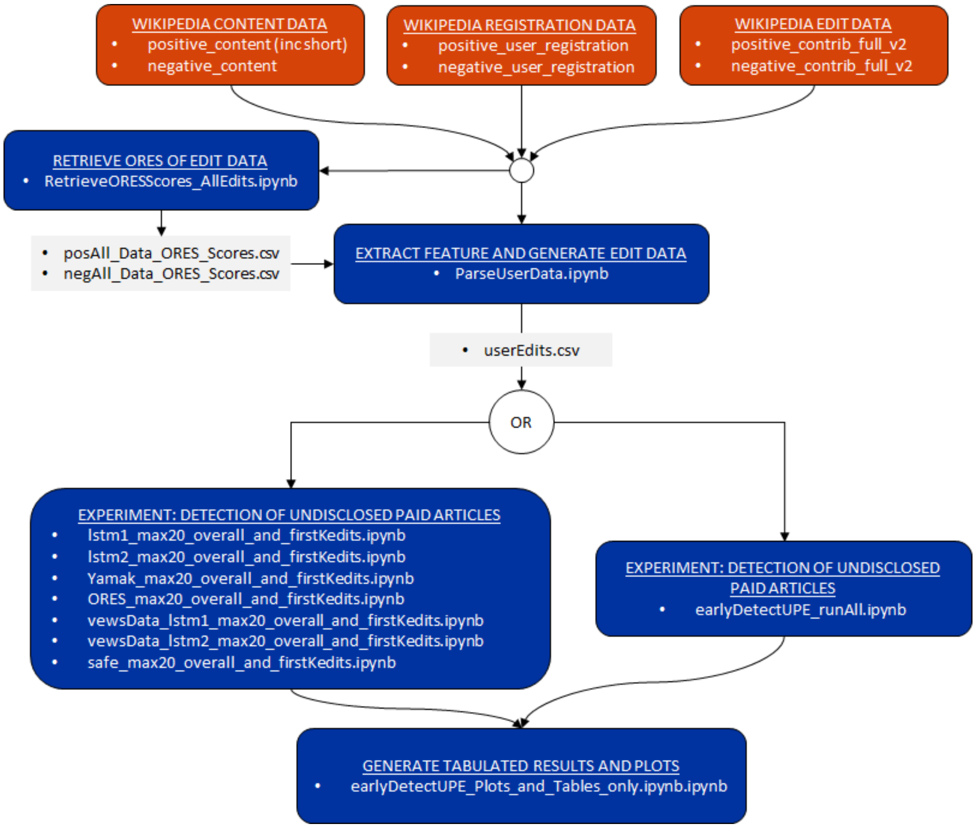
WIKIPEDIA CONTENT DATA
• positive_content (inc short)
• negative_content
WIKIPEDIA REGISTRATION DATA
• positive_user_registration
• negative_user_registration
WIKIPEDIA EDIT DATA
• positive_contrib_full_v2
• negative_contrib_full_v2
RETRIEVE ORES OF EDIT DATA
• RetrieveORESScores_AllEdits.ipynb
• posAll_Data_ORES_Scores.csv
• negAll_Data_ORES_Scores.csv
EXTRACT FEATURE AND GENERATE EDIT DATA
• ParseUserData.ipynb
• userEdits.csv
OR
EXPERIMENT: DETECTION OF UNDISCLOSED PAID ARTICLES
• lstm1_max20_overall_and_firstKedits.ipynb
• lstm2_max20_overall_and_firstKedits.ipynb
• Yamak_max20_overall_and_firstKedits.ipynb
• ORES_max20_overall_and_firstKedits.ipynb
• vewsData_lstm1_max20_overall_and_firstKedits.ipynb
• vewsData_lstm2_max20_overall_and_firstKedits.ipynb
• safe_max20_overall_and_firstKedits.ipynb
EXPERIMENT: DETECTION OF UNDISCLOSED PAID ARTICLES
• earlyDetectUPE_runAll.ipynb
GENERATE TABULATED RESULTS AND PLOTS
• earlyDetectUPE_Plots_and_Tables_only.ipynb.ipynb

Figure A.2: Early Detection of Undisclosed Paid Editors Process.