Detecting Sockpuppet Accounts in Wikipedia: A Quantitative Evaluation of Different Models

Skaistė Mielinytė

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

Sockpuppetry problem, where the same user creates multiple accounts to engage in some malicious activity, significantly undermines the quality of Wikipedia. Past research addressed this problem by presenting many candidate models for automated sockpuppet accounts' detection in Wikipedia, however the suggested models were never directly compared, which prevented from knowing which model has the highest potential of achieving good practical results in Wikipedia. This study filled the gap by recreating the models (textual, non-textual and combined), described in the past research, evaluating them on the standard dataset formed from users' comments in Wikipedia's talk pages and directly comparing the accumulated results. In order to make the comparison up-to-date and reflecting the latest innovations, transformers were also fine-tuned to detect sockpuppet accounts in Wikipedia and included in the comparison after evaluating them on the same dataset. Findings of this study show that transformers, more precisely the RoBERTa transformer, are the most compatible with the sockpuppetry problem in Wikipedia - the RoBERTa transformer achieved 84.4% recall, 9.8% FPR, 89.3% precision and 86.7 f-score. It was also found that the RoBERTa transformer can achieve comparable results when generalising on comments from unseen topics and unseen time periods, which makes RoBERTa even a more promising candidate. Because of the high FPR, the currently proposed RoBERTa can only offer a semi-automated sockpuppet detection, however it is still sufficient to considerably improve the current completely manual sockpuppet detection mechanism in Wikipedia. Having a more effective sockpuppet detection system should make Wikipedia a more accurate, objective and welcoming website.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 34872
Date when approval was obtained: 2023-01-27

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Skaistė Mielinytė*)

Acknowledgements

I would like to thank my supervisor Dr Björn Ross for providing me continuous guidance and help while I was working on this project. I am especially grateful for his suggestion to use transformers in this study, which, I believe, highly increased the value of my study.

I would also like to thank my family for always supporting me in every possible way and my friends for their companionship. Moral support of my family and friends really helped not only while working on this project, but also throughout all four academic years.

Table of Contents

1 Introduction **1**

2 Background **3**
2.1 Sockpuppetry problem in Wikipedia 3
2.1.1 Motivations 3
2.1.2 Reporting a sockpuppet account 4
2.1.3 Reviewing a reported sockpuppet account 4
2.1.4 Automation need 4
2.1.5 Current state of the sockpuppetry problem in Wikipedia . . . 5
2.2 Related work 5
2.2.1 Models based on textual features 6
2.2.2 Models based on non-textual features 6
2.2.3 Models combining different features 7
2.2.4 Recent works 8
2.3 Research questions 8

3 Data **10**
3.1 Motivation for a new dataset 10
3.2 Gathering data from Wikipedia 10
3.2.1 Source 10
3.2.2 Sockpuppets 11
3.2.3 Non-sockpuppets 12
3.2.4 Metadata 15
3.3 Cleaning data 16
3.3.1 Cleaning main files 16
3.3.2 Cleaning metadata files 16

4 Comparing different model groups on standard metrics (RQ1) **17**
4.1 Methodology 17
4.1.1 Recreating models 17
4.1.2 Evaluating models 22
4.2 Results 23
4.3 Interpretation of results 23

5 Sockpuppet detection with transformers (RQ2) **26**
5.1 Motivation 26

5.2 Methodology . . . 27
5.2.1 Selecting transformers . . . 27
5.2.2 Fine-tuning transformers . . . 27
5.2.3 Evaluating transformers . . . 28
5.3 Results . . . 28
5.4 Interpretation of results . . . 29

6 Practical evaluation of top-performing models (RQ3) 30
6.1 Motivation . . . 30
6.2 Methodology . . . 31
6.2.1 RQ3a . . . 31
6.2.2 RQ3b . . . 33
6.3 Results . . . 34
6.4 Interpretation of results . . . 34

7 Discussion 36
7.1 Implications . . . 36
7.2 Comparison with existing literature . . . 37
7.3 Critical evaluation . . . 38
7.3.1 Design limitations . . . 38
7.3.2 Implementation limitations . . . 39
7.4 Possible future work . . . 39
7.5 Conclusion . . . 40

Bibliography 41

A First appendix 45
A.1 Formulas of metrics . . . 45
A.2 Extended results of RQ1 . . . 46
A.3 Additional data for RQ2 . . . 48

Chapter 1

Introduction

Wikipedia can be written by anyone, but not everyone contributing to it has good intentions. One of the major sources undermining Wikipedia's quality are sockpuppet accounts. Sockpuppets are a type of fake identity where the same person creates multiple accounts and uses them for malicious purposes, i.e. sockpuppetry is "the misuse of multiple Wikipedia accounts" as originally defined in Wikipedia [46]. It is important to note that some user having multiple Wikipedia's accounts do not automatically count as a sockpuppetry case - secondary accounts are referred to as sockpuppets only when a person uses them to violate the established policies [47, 46]. Examples of policy violation include using secondary accounts to spread fake news, to circumvent a block or sanction of a primary account, to create the illusion of greater support for a primary account's position in discussion pages, etc [46].

Sockpuppet accounts can cause a severe damage for Wikipedia. Wikipedia is built on 5 fundamental principles, also known as "Five pillars" [41], and sockpuppets' activity can easily violate even 3 of them. The first principle is "Wikipedia is an encyclopedia ... Wikipedia is not a soapbox, a battleground, or a vehicle for propaganda, advertising and showcasing." Users often use sockpuppets to carry on ideological battles, spread propaganda and, in general, foster prejudice, hatred, or fear, which clearly does not comply with this principle. The second violated pillar is "Wikipedia is written from a neutral point of view". Users can create the illusion that their viewpoint is superior by posting favorable comments from their sockpuppet accounts. This tactic may lead to the acceptance of their position instead of objectively presenting multiple perspectives, i.e. writing from a neutral point of view. The last violated principle is "Wikipedia's editors should treat each other with respect and civility". The usage of sockpuppet accounts often results in edit wars, conflicts and overall makes it harder to reach consensus. All of this creates tension between users and makes Wikipedia a less welcoming place.

Even though Wikipedia has a method to detect and ban sockpuppet accounts, it is arguably insufficient given the severity of the problem. Sockpuppets are caught and blocked in a 2-stage process: suspected sockpuppets are firstly reported by other users [35] and then manually reviewed by administrators [44], clerks [45] and, if requested, checkusers [39]. Arguably, the current procedure involves too many parties, requires too much manual effort and is prone to human error. Researchers have tried to address these

problems by suggesting automated sockpuppet detection solutions based on machine learning. They can be grouped into 3 categories based on the features used: textual features [23], non-textual features [25, 50] and a combination of several types of features [51, 53]. Even though several of these models were reported to have achieved good results, none of them have actually been integrated into Wikipedia. One of the reasons none of the models were adopted could be because it is not possible to directly compare the reported results, as different evaluation datasets were used, and consequently know which model is the most promising. This creates a paradox - even though many solutions were suggested to solve a severe problem, no effort was made towards actually analysing which solution has the highest potential of achieving good practical results.

Here comes this study whose aim is to directly compare models from different groups on standard metrics as well as from practical dimensions and ultimately determine the best candidate model for sockpuppet detection in Wikipedia. This study directly compared 3 model categories by firstly collecting the standard dataset of comments made by Wikipedia users in talk pages and recreating a representative model for each of the groups. After that, the recreated models were evaluated on this dataset and accumulated results were directly compared in order to find the best solution. Also, it was noticed that recreated models are limited to ML technologies and thus do not reflect the latest innovations in the text technologies field. For this reason, transformers were also fine-tuned to detect sockpuppet accounts and included in the direct comparison. Using transformers to detect sockpuppet accounts was, in fact, a novel approach in Wikipedia's context. Summarising the work completed, the most important contributions of this study are as follows:

- This study enabled the direct comparison of 3 sockpuppet detection model groups based on different types of features (textual, non-textual, combined) and outlined the most promising group, as well as its specific implementation.
- This research presented a novel approach of using transformers to catch sockpuppet accounts in Wikipedia. To the best of our knowledge, transformers have never been used before to catch sockpuppets specifically in Wikipedia.
- This paper evaluated the recreated models on more practical aspects such as topic generalisability (generalising on unseen topics) or temporal generalisability (generalising on unseen time periods). This evaluation was not found in the original papers, even though it is important if models were to be actually integrated into Wikipedia.
- A new dataset from users' comments in Wikipedia's talk pages was produced. It is sufficient in size: 146,886 contributions from 4,966 users (50% sockpuppets, 50% real accounts), and can be used to analyse the sockpuppetry problem from other dimensions or to analyse related problems such as user commenting behaviour.

Chapter 2

Background

The background, important for this study, can be divided into 2 sections. The first one is the sockpuppetry problem in Wikipedia which covers motivations for creating a sockpuppet account, current procedures of reporting and reviewing suspicious accounts and overall relevance of the problem. The second topic is literature review. Papers about the sockpuppetry problem are discussed in terms of their methodology, results and individual limitations.

2.1 Sockpuppetry problem in Wikipedia

2.1.1 Motivations

Sockpuppetry problem, where the same person (often called sockmaster, puppetmaster, or sockpuppeteer) creates multiple accounts and uses them for malicious purposes, is a well-known fake identity problem in social platforms. Common reasons for creating a sockpuppet account includes defending or supporting a sockmaster in online discussions, participating in some malicious activity, e.g. spreading fake news, or simply regaining access to the platform after the main sockmaster's account was blocked. All these reasons apply to Wikipedia as well, however, its unique setup adds even more motivations for sockpuppetry [34]. For instance, in Wikipedia sockpuppet accounts are used to cast additional votes in deletion debates or talk pages. In these pages the outcome is generally determined by consensus - having more accounts gives more votes for a sockmaster and theoretically increases the probability of some article, which a sockmaster cares about, not being deleted or some controversial edit, which a sockmaster made and which is being discussed in talk pages, being accepted. One more Wikipedia-specific motivation for creating sockpuppets is to gain more reverts in edit wars [40]. Edit wars happen when two editors have opposing views on some topic and hence constantly override each other's changes in an article. Current Wikipedia's rules allow up to 3 reverts on the same article within a 24-hour period, however having multiple accounts consequently increases this number and disturbs the balance between 2 opposing editors. As a final note, it is also important to mention that Wikipedia's design makes it very easy to create multiple accounts. One of the main properties of Wikipedia is that its setup is anonymous. When creating accounts, users do not have to

specify their personal information such as email address (it is only recommended, but not mandatory) which would allow linking accounts of the same user.

2.1.2 Reporting a sockpuppet account

In Wikipedia, any user (even a not registered user) who suspects that some account is a sockpuppet, can report it. A reporting user, however, must have sufficient evidence that 2 accounts are connected, for instance, they should list edited pages where they noticed signs of sockpuppetry. The current procedure of reporting a sockpuppet account is as follows: a user needs to go to the reporting page [35], enter sockmaster's name and then list all the sockpuppet accounts they noticed and provide all the evidence they collected. They can optionally request this case to be reviewed by a checkuser whose functionality is described in the following section.

2.1.3 Reviewing a reported sockpuppet account

Sockpuppet investigations (SPI) are a manual action normally performed by administrators and clerks, and, if requested, by checkusers. Administrators [44] are responsible for carefully and neutrally assessing the provided evidence and deciding the outcome for SPI which includes completely or temporarily blocking an account or extending an existing block. It is important to note that even though there are many signs that some account is a sockpuppet [34], none of them guarantees that sockpuppetry is taking place and in reality the administrator needs to use their subjective judgement and intuition. Arguably, the decision making in SPI is subject to human biases and errors.

Checkusers [39] get involved in SPI when an administrator thinks that the provided evidence is not sufficient for a decision or when the reporting user initially requested a case to be reviewed by a checkuser. Checkusers are a small group of trusted Wikipedia users who have access to IP addresses used by Wikipedia user accounts. By comparing IP addresses of accounts under investigation, checkusers provide their decision of how likely it is that these accounts are controlled by the same sockmaster. After a checkuser provides their decision, an administrator can re-assess the case. Importantly, checkusers never reveal the IP addresses themselves as sharing the IP addresses with others would break Wikimedia Foundation's privacy policy [7].

Even though clerks [45] don't have much decision power in SPI or have access to non-public information such as IP addresses, they are very important when investigating sockpuppetry cases. Clerks ensure that cases have sufficient evidence and if not, they request such evidence. They also actively participate in SPI "housekeeping" tasks which include formatting, merging the same cases, closing and archiving resolved cases.

2.1.4 Automation need

Presumably, Wikipedia's methods of resolving SPIs require too much manual effort and potentially are error-prone. Up to 3 different roles of Wikipedia's users can be involved in the SPI case, namely, administrators, clerks and checkusers. Administrators, clerks and checkusers are Wikipedia's editors who got their corresponding role by completing

various courses and maintaining a good reputation [44, 45, 39]. Consequently, these are the people who are very knowledgeable of Wikipedia and have spent a considerable amount of time in the platform writing or editing articles and contributing to Wikipedia's welfare in other ways. Since obtaining a role requires significant effort, these individuals are a limited resource, e.g. currently, the English Wikipedia has 909 administrators [37] and only 49 checkusers [39]. It is debatable if involving such people in the SPI review process, which potentially could be automated, is a good use of their time and knowledge. If the SPI review process required less manual effort, these people could continue to work on articles which probably would bring more benefit to Wikipedia, as knowledge sharing is its main purpose. Also, manual SPI reviews may lead to human error. Some evidence might be overlooked or mistakenly mixed up between different cases which may result in inaccurate decisions. As already mentioned in 2.1.3, a substantial part of the SPI review process requires individual judgement. Hence, some sockpuppet investigations might get stricter examination than others, which leads to inconsistencies. Having an automated solution for sockpuppets detection should provide a more universal and consistent way of investigating sockpuppet cases.

2.1.5 Current state of the sockpuppetry problem in Wikipedia

Even though Wikipedia does not provide any statistics for the sockpuppetry problem (e.g. average number of new SPIs each day), it is evident that sockpuppetry is an ongoing problem as people actively report sockpuppets in the reporting tool till this day [35]. In fact, sockpuppetry is not only an ongoing problem but it falls into a category of important problems that Wiki community actively tries to solve. Sockpuppet detection problem is being mentioned in various Wiki-related conferences including Wiki Workshop which is an annual "forum bringing together researchers exploring all aspects of Wikimedia projects" [49]. Looking specifically at Wiki Workshop, it has been including sockpuppet's problem in the "topics of interest" list for 5 years now (2018-2023). However, regarding the most recent literature of this topic, not much was done for investigating the sockpuppetry problem in Wikipedia after 2019 (see 2.2). This imbalance between demand and supply shows that more contribution is needed towards analysing the sockpuppetry problem in Wikipedia, especially its automated solutions.

2.2 Related work

The topic of automating sockpuppet account detection has been widely researched for many social media platforms [1], especially for Twitter, Wikipedia, Facebook and microblogging websites like Sina Weibo. Since Wikipedia has a unique setup, functionalities and a different purpose than the rest of these websites (Wikipedia is a knowledge sharing, not opinion sharing platform as opposed to Facebook, Twitter and microblogs), methods applied to other websites cannot always be applied to Wikipedia. Consequently, the most relevant work in the area is that which used data specifically from Wikipedia.

This section reviews and evaluates work on sockpuppet detection in English Wikipedia by grouping it into 3 sections based on the type of features used. The most recent

work in each of the 3 sections is reviewed in detail because these works are closely related to this study. The remaining works are briefly mentioned without evaluating them critically just to present the diversity of methods.

2.2.1 Models based on textual features

One of the first papers in this field is Solorio et al. "A Case Study of Sockpuppet Detection in Wikipedia" [23] which dates back to 2013. In this paper, Authorship Attribution (AA) techniques were used to match sockpuppet accounts with their sockmaster by analysing their comments in various talk pages. Researchers used 239 textual features such as total number of characters, total punctuation count or emoticons count. Dataset consisted of 77 collected SPI cases each of which required a separate training and testing phase - for each SPI case, researchers trained the model on comments made by the sockmaster account, and afterwards tested it on the comments made by other accounts (including sockpuppet accounts) in the same talk page. The proposed 239 textual AA features proved to be somewhat effective as the model achieved accuracy of 69%, recall of 75%, precision of 68% and F-measure of 72%. Nevertheless, separate training and testing phase setting results in a high computational cost, which is a limiting factor, and raises a question if this model can actually be used in Wikipedia because retraining a model on each new SPI is probably not feasible. Other drawbacks of this study include limited data in terms of the dataset itself (only 77 entries) and its content. In the training set, each sockmaster wrote around 80 comments on average, while in the testing set, the average number of comments made by one user was 8. Such numbers are not very compatible with the authorship attribution method which expects long pieces of text in order to match text with its author.

The same group of scientists presented a new dataset of 632 cases in their further research called "Sockpuppet Detection in Wikipedia: A Corpus of Real-World Deceptive Writing for Linking Identities" (2013) [24]. Applying this dataset to the previous model [23] changed the F-measure from 72% to 73%. So increasing the dataset did not have much effect on the SVM model's performance. Interestingly enough, the effect on other metrics was not reported. However, one cannot claim that the model would work well on the majority of SPI cases, because 632 cases is still a low number and might not reflect the variety of SPI cases. As a final note, it is important to remember that this model can only be used to match sockpuppet accounts with pre-approved sockmaster accounts (i.e. accounts which were known to be sockmaster accounts in advance), but it cannot identify new sockmaster accounts and find their sockpuppets.

2.2.2 Models based on non-textual features

Tsikerdekis et al. model, described in "Multiple Account Identity Deception Detection in Social Media Using Nonverbal Behavior" (2014) [25] paper, used non-textual features to binary classify Wikipedia users into sockpuppets and non-sockpuppets. This model used 2 types of non-textual features, namely, time independent (e.g., time difference between account registration and the first edit) and time dependent ones (e.g. the total number of bytes added or removed by a user for a time window of 30 days since their initial registration with the website).

In 2016, Yamak et al. presented another binary classification model, which used non-textual features, to detect Wikipedia's sockpuppets in their study "Detection of Multiple Identity Manipulation in Collaborative Projects" [50]. In some sense, this study was the continuation of Tsiderkekis' study because some of the features from Tsiderkekis study [25] were reused such as the interval between the user's registration and their first contribution. However, some new features were also added, e.g. the number of user's contributions in each Wikipedia's namespace, the average number of contributions in the same article, etc. A number of ML models, namely, Support Vector Machine (SVM), Random Forest (RF), Naive Bayesian (NB), K-Nearest Neighbor (KNN), Bayesian Network (BN) and Adaptive Boosting (ADA), were evaluated on the dataset of around 10,000 user accounts, active between 2004 and 2015. RF and NB gave an extremely high precision of 98% and 97% respectively, and also an extremely low FPR of 2% and 3% respectively. However, such good results are questionable because the sockpuppet detection problem is not a simple classification problem, so some model having accuracy of almost 100% sounds unrealistic. In the "Conclusion and Future Work" section authors claim: "We plan to study more closely how these algorithms perform with the use of a third (development) dataset", which suggests that only training and testing sets were used. Hence, it might be the case that parameter tuning was done on the testing set, which would mean that reported results cannot be trusted. Also, the reported TPR and FPR always add up to 1 for all the models, even though TPR and FPR adding up to 1 is not true in general. However, TPR and FNR always add up to 1, so there is a slight possibility that authors mistakenly reported FNR instead of FPR, which also makes reported results less trustworthy. Despite these inaccuracies, this study surpassed the previous Solorio's study [23] by having a setup which requires lower computational cost, by being evaluated on a considerably bigger dataset, by relying on non-textual features which cannot be manipulated easily by sockpuppets and by being able to identify new sockpuppet accounts (i.e. not only sockpuppet accounts of some predefined set of sockmasters). The single aspect on which Yamak's model remains weaker than Solorio's is the complexity of features - to calculate features for the Yamak's study one needs high loads of data taken from various Wikipedia's namespaces, whereas Solorio's model's features can be calculated from talk pages only. It would be interesting to investigate if it is possible to restrict the data to only a few namespaces and still get satisfactory results.

2.2.3 Models combining different features

Several studies tried to combine different types of features. In 2018, Yamak et al. in study "SocksCatch: Automatic detection and grouping of sockpuppets in social media" [51] used non-textual features to detect sockpuppet accounts and then applied graph theory and community detection algorithms to group sockpuppets which belong to the same sockmaster.

Yu et al. in paper "Sockpuppet Detection in Social Network Based on Adaptive Multi-source Features" (2021) [53] described combining verbal and non-verbal features for sockpuppets detection. In this study, verbal features (e.g., text length, question frequency, and number of emotional words) were combined with non-verbal features (e.g., speaking time of each user, the number of comments per user per day) using adaptive

multi-source feature fusion. The dataset consisted of 685 users: 370 sockpuppets and 315 legitimate. Several ML algorithms, namely, Support Vector Machine (SVM), Random Forest (RF) and Adaptive Boosting (ADA), were used to verify the effectiveness of the method. All models performed relatively well: the average precision of models was 79%, average recall 82% and average F-1 score 80%. Even though these are quite good results, FPR wasn't reported. FPR is a very important metric for the sockpuppetry problem because accusing a legitimate user with sockpuppetry would mean blocking a completely innocent account and presumably losing that user's trust in Wikipedia. The dataset of this study is similar to the dataset used in Solorio's study [24] and has the same pros and cons. Since it only consists of data from talk pages, it makes it easy to calculate the needed features which is a plus, however as it contains only 685 users, it does not have the potential to represent many different Wikipedia's users, and the study's evaluation is weaker than the evaluation of Yamak's study [50]. This study shares some similarities with the Yamak's study [50] as well - both studies do not require to know the predefined set of sockmasters to do the classification, however both models are limited in a way that they cannot link sockpuppets of the same sockmaster.

2.2.4 Recent works

Looking at the previous sections (2.2.1, 2.2.2, 2.2.3), it is apparent that there are not many recent studies done about sockpuppetry in Wikipedia. In recent years, sockpuppetry and fake identity problems still remained a relevant topic in literature, however, researchers' focus shifted to other platforms especially social network and microblogging sites [3, 19, 28, 55, 27] . For instance, in 2019, Zhou et al. study [55], which analysed Sina Weibo website, observed the dynamic growth of the social interactions between 2 users. This study concluded that once blocked, a sockmaster tries to recover its prior connections as soon as possible to maintain the same influence in the website. In 2022, Du et al. [3] analysed user sentiment with the help of deep learning methods in 2 Chinese social network websites, namely, Sina Weibo and Douba. Researchers found that user sentiment can be effectively used to link accounts belonging to the same person. One of the most recent studies in the field is Verma et al. work [27]. They tried several combinations of different ML and deep learning models and found that the most effective model to classify an account as real or fake in OSN (online social network) is the fusion of RoBERTa transformer, Bi-LSTM (Bidirectional LSTM) and RF (Random Forest). The findings were confirmed by the consistent results from the evaluation on 2 datasets: OSN and Twitter. Summing up, there is a variety of newer sockpuppet detection methods tested on other social network platforms, but not on Wikipedia.

2.3 Research questions

Considering the relevance of the sockpuppetry problem in Wikipedia (2.1.5) and benefits which come from automating sockpuppet detection procedure (2.1.4), it becomes clear that there is a high need for automated or partially automated sockpuppet detection mechanism. Even though literature suggests many different ML solutions (2.2) for the sockpuppet detection, it is not clear which solution is the most promising and has the highest potential of achieving good practical results. One cannot simply compare

the reported metric results of different models because models were evaluated on completely different datasets which varied in source and size - some datasets were formed from talk pages and included only a few hundreds of cases [23, 24, 53], while others were generated from various namespaces' data and included several thousands of cases [50, 51, 25]. Also, the models themselves differ in their classification task - for some it is binary classification [53, 50, 25], for others it is matching sockpuppets with their sockmaster [23, 51]. Overall, it is clear that models cannot be compared as-is and a study which would enable comparing different models and choose the most compatible one is needed. Here comes this research study and the first research question:

RQ1 **How do different models compare in terms of their classification precision, recall, f-score and FPR? Based on these metrics, which model is the most compatible with the sockpuppetry problem in Wikipedia?**

The main goal of this question is to recreate 3 binary classification model archetypes from different feature groups (textual features, non-textual features and combination of textual and non-textual features) and to evaluate them on the uniform dataset. However, if one wants the comparison to be up-to-date and reflecting the current state of the field, analysing only the existing literature is not sufficient. As already noted in 2.2.4, sockpuppetry in Wikipedia has not gotten much attention since 2019, which means that the existing literature around sockpuppetry problem in Wikipedia did not try using state-of-the-art models. Considering how powerful some of the state-of-the-art models are (GPT, BERT, etc.), there is a high chance that they might achieve better results and turn out to be more compatible with the sockpuppetry problem than the recreated models which used traditional ML techniques. Hence, my research study also tries to bring some innovation by introducing transformers (belong to textual features group) to tackle this problem. This is what the second research question is about:

RQ2 **Does switching to transformers increase classification scores for textual models? Revisiting previous scores from RQ1 and transformers' scores, which model is the most suited for the sockpuppetry problem in Wikipedia?**

Answering RQ1 and RQ2 would reveal the most compatible model based on the standard metrics only. As already stated, one of the main goals of this study is to analyse which model has the highest potential of achieving good practical results. This invites evaluating models from a more practical perspective:

RQ3 **How well do selected models generalise to unseen data:**

a) **Can models easily generalise to unseen topics?**

b) **Can models easily generalise to data taken from unseen time periods, especially to the most recent data?**

Answering RQ1-RQ3 will reveal which feature group's model (textual, non-textual or combined) is the most likely to achieve good practical results on detecting sockpuppet accounts in Wikipedia. Depending on the fall-out (FPR) ratio, this will either reveal a candidate for a fully automated or a semi-automated solution (i.e. still requiring some human intervention). Regardless of the automation level, this study is expected to mitigate the Wikipedia's sockpuppetry problem by suggesting a real practical solution.

Chapter 3

Data

3.1 Motivation for a new dataset

In order to conduct this study, I created a completely new dataset formed of data gathered from Wikipedia. A new dataset was needed because datasets used in related studies (see 2.2) were not publicly available. Recreating datasets used in previous studies was also not feasible because their dataset collection lacked detailed descriptions how to reproduce the process [24, 23, 50, 53], required too much manual inspection [23, 24] or produced too few sockpuppetry cases [23, 24, 53]. These problems led to designing an alternative data collection methodology. By following this new methodology, I created a dataset of 4,966 users (50% sockpuppet accounts, 50% non-sockpuppets acounts) and their contributions to the discussion area (i.e. talk pages) of English Wikipedia's articles. The final dataset consisted of 2 main files: contributions from sockpuppet accounts ("socks_with_contribs.json"), contributions from non-sockpuppet accounts ("control_with_contribs.json"), and some metadata files: accounts registration information ("sock_registrations.json", "control_registrations.json"), timestamps of first contributions ("control_first_contribs.json") and talk pages' category information ('title_categories.pickle').

3.2 Gathering data from Wikipedia

3.2.1 Source

MediaWiki API [8] was used as the source to gather the needed data. MediaWiki API is a web service that provides a number of endpoints to interact with Wikipedia including endpoints to retrieve information. The benefits of gathering data via the MediaWiki API are that HTTP requests are highly customizable and can be restricted to return only the data that is actually needed. This resulted in memory-light main files of ≈ 27 MB and ≈ 68 MB for sockpupppets' contributions and non-sockpuppets' contributions respectively. However, getting data via the MediaWiki API resulted in many HTTP requests and posed a risk of overwhelming the MediaWiki API's servers. To mitigate this risk, information was retrieved by adhering to the rules described in MediaWiki

API's "Etiquette" page [9], for example, requests were serialised rather than sent in parallel and several requests were combined into one by using the pipe character ("|"). Even though this sometimes resulted in a longer waiting time, a safe request rate was maintained and a fair usage of the API was ensured.

3.2.2 Sockpuppets

3.2.2.1 Getting sockpuppets

Since accounts participating in sockpuppetry eventually get blocked, usernames of sockpuppet accounts were collected by looking at blocked Wikipedia's users and filtering those accounts which were blocked because of sockpuppetry.

To get blocked accounts, MediaWiki API:Blocks endpoint [10] was used. A Python's script was written which sent a request [1] of the form:

```
https://en.wikipedia.org/w/api.php?action=query&list=blocks&
bkstart=<start_timestamp>&bkend=<end_timestamp>&bkdir=newer
&bklimit=500&format=json
```

for each month from 2010 January to 2022 December. The request frequency was set to once a month because of the "bklimit" parameter. The "bklimit" field has the upper limit of 500 which means that one request can return at most 500 blocked accounts in the response. This limitation required to experiment with different request frequency and pick the one which finds the majority of blocked accounts but does not overwhelm the servers. The decision was made to set the request frequency to once a month, because this frequency collects the majority of blocked accounts (doubling the frequency and querying twice a month increased the number of collected sockpuppets only by 1.56%) and does not overload the servers as it results in the total of 156 requests only.

As already mentioned, account filtering was also needed because API:Blocks endpoint returns accounts which were blocked for any reason, not only for sockpuppetry. Conveniently enough, the "reason" field in the API:Blocks endpoint's response includes the reason why each account was blocked. Hence, the Python script saved only the usernames of those accounts whose block reason mentioned being involved in sockpuppetry (e.g. [[WP:SOCK]], [[WP:SOCK—Abusing Multiple Accounts]], etc).

All the aforementioned steps were completed on the 22nd of November 2022 and resulted in getting 20,361 usernames of sockpuppet accounts.

3.2.2.2 Getting contributions of sockpuppets ("socks_with_contribs.json")

Once usernames of sockpuppet accounts were found, the aim was to leave only those accounts which made some contribution (adding/editing/deleting comments) to the talk pages of English Wikipedia's articles and then to get the specific contributions of these accounts.

To look at the contributions of each account, MediaWiki API:Usercontribs [14] endpoint was used in the form:

[1] Arguments written in <> are variable (applicable for all the following HTTP requests).

```
https://en.wikipedia.org/w/api.php?action=query&list=
usercontribs&ucuser=<sockpuppet_usernames>&uclimit=500&
ucnamespace=1&ucprop=ids|sizediff|title|timestamp&format=json
```

The field "ucuser" contained usernames (combined with the "|" symbol in order to decrease the number of requests) of the accounts, for which contributions were retrieved, "uclimit" was set to the maximum value of 500 meaning that for each account at most 500 contributions were returned and "ucnamespace" was set to the numeric code of 1 of the "Talk" namespace which ensured that the retrieved contributions were only taken from the talk pages. If this request returned no contributions for some account, this means that this account did not contribute to talk pages and hence this user was ignored and not included in the final "socks_with_contribs.json" file.

For accounts which had contributed to the talk pages, the "ucprop" field shows what kind of information was returned about each of their contributions: IDs (page's ID, revision's ID and previous revision's ID), size of the contribution's diff, title of the talk page an account was contributing to and finally the timestamp of the contribution. This information was saved to the final "socks_with_contribs.json" file. However, the MediaWiki API:Usercontribs endpoint is not capable of returning the content of a contribution, i.e. the comment itself. For this reason, for every contribution which had a positive "sizediff" MediaWiki API:Compare endpoint [12], which compares 2 versions of a page, was called. By using the revision ID of each contribution and comparing it with the previous revision of that talk page, this endpoint returned comments an account has added or edited. These comments were saved to the "socks_with_contribs.json" file. The final file mapped each sockpuppet account with the list of their contributions and each of the contributions had the following content:

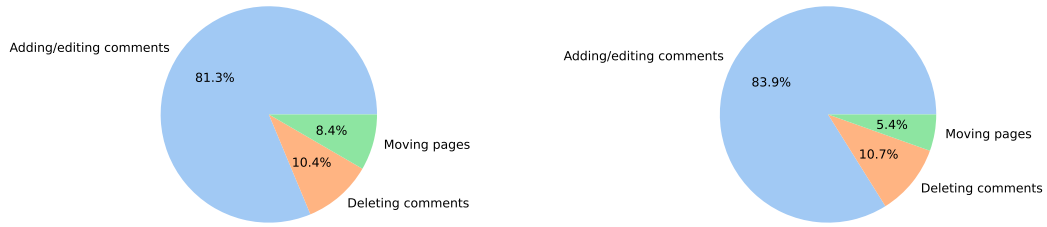
```
{ "pageid": int,
  "revid": int,
  "parentid": int,
  "title": string,
  "timestamp": timestamp,
  "sizediff": int,
  "comments": string }
```

The final version of "socks_with_contribs.json" file was generated on the 15th of December 2022 and contained 40,090 contributions to talk pages from 2,483 sockpuppet accounts. Contributions were made between 2003 February and 2022 October. Out of 40,090 contributions, 32,877 contributions were added or edited comments, 3,990 contributions were about deleting comments and hence had no "comments" content and 3,223 contributions were about moving pages, hence they also had no "comments" content. The content of the sockpuppet data is summarised in the figure 3.1a.

3.2.3 Non-sockpuppets

3.2.3.1 Getting non-sockpuppets

In order to train the classifier, the control group of non-sockpuppets was needed. Even though the majority of related studies (see 2.2) collected non-sockpuppet accounts



(a) Sockpuppets' contributions (b) Non-sockpuppets' contributions

Figure 3.1: Types of contributions in the final dataset.

without controlling for any variables, the decision was made to control for an account's activity time when collecting non-sockpuppet accounts and control for an account's activity range when collecting specific non-sockpuppet contributions.

More precisely, to get the usernames of non-sockpuppets, for each sockpuppet a matching unique non-sockpuppet, which was active around the same time as a sockpuppet, was found. This was done by taking some contribution from a sockpuppet account and looking at the 15 former and 15 subsequent contributions in the same talk page made by other accounts with the MediaWiki API:Revisions endpoint [13]. From the potential matches, only 1 not blocked (prevents from mistakenly including sockpuppet accounts into non-sockpuppet files) and not already used (ensures that the final "control_with_contribs.json" file contains the same number of 2,483 accounts) non-sockpuppet account was added to the "control_with_contribs.json" file. The final list of usernames of non-sockpuppet accounts contained 2,483 usernames as expected.

Controlling for the activity time was needed because if non-sockpuppets and sockpuppets were active at completely different times, this might influence the topics they are contributing to or the linguistic style they are using when commenting since the popularity of topics and writing style might change over time. For example, in the final data for the sockpuppets, 80% of the contributions were made before 2019 and hence topics like Covid-19 are not expected to be dominating topics among the sockpuppets. However, if only those non-sockpuppet accounts, which were active for the last few years, were sampled, the probability of them mentioning Covid-19 increases and hence the model might become biassed and classify accounts mentioning Covid-19 as non-sockpuppets. The effect of controlling for the activity time is depicted in the diagram 3.2. Distributions of contributions of both groups, namely, sockpuppets and control group, follow similar patterns - there aren't many contributions until 2006, spikes (2011, 2018) and dips (2019-2020) happen around the same time. This proves that the timing of contributions of both groups are comparable, however the height of bars is very different which is explained in the following section.

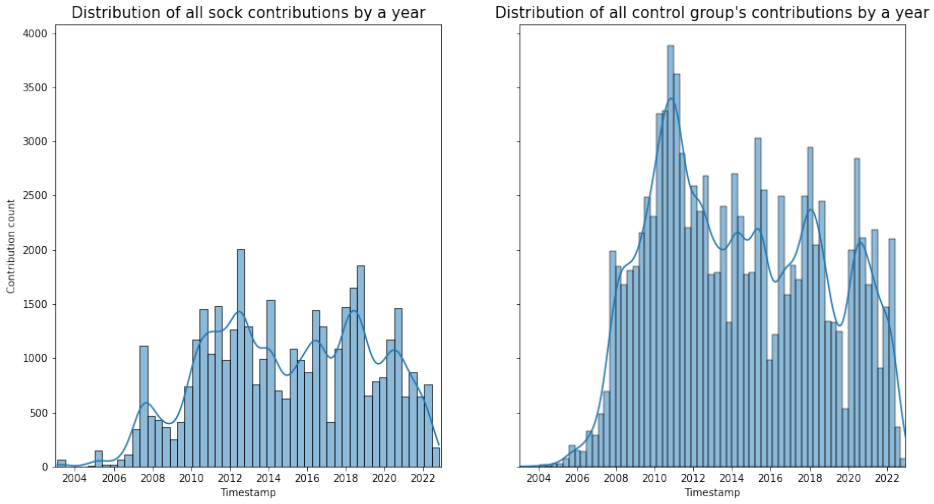


Figure 3.2: Distribution of all contributions by their timestamp.

3.2.3.2 Getting contributions of non-sockpuppets ("control_with_contribs.json")

To get the contributions of non-sockpuppet accounts, the identical process was followed as when retrieving the contributions for sockpuppet accounts (see 3.2.2.2): MediaWiki API:Usercontribs endpoint retrieved each user's contributions and then MediaWiki API:Compare endpoint was used to get the content of each comment. The final fields saved for each contribution also followed the same json structure depicted in 3.2.2.2. However, as already mentioned, each non-sockpuppet account was additionally controlled for their activity range when collecting the contributions. This means that non-sockpuppet account's contributions were retrieved only for the time period when a matching sockpuppet was active (time period between sockpuppet's first and last contribution in the "socks_with_contribs.json" file). This was needed because non-sockpuppet accounts have an unfair advantage of being active for a longer period of time as they are not eventually blocked unlike the sockpuppet accounts. Naturally, the probability of non-sockpuppet accounts having more contributions increases and hence the model might classify the accounts hugely depending on the number of contributions an account has. In order to decrease the significance of this non-textual feature when classifying an account, especially for the textual models, controlling for activity range was introduced. Nevertheless, even when controlling for it, this still resulted in non-sockpuppet accounts having a higher number of contributions as bars are significantly higher for this group in the RHS table of 3.2. This shows that legitimate accounts are naturally more active which is reasonable because users are expected to use their primary accounts, i.e. legitimate accounts, more than their secondary accounts, i.e. sockpuppet accounts.

The final version of the "control_with_contribs.json" file for non-sockpuppet accounts was generated on the 13th of January 2023 and contained 106,796 contributions to talk pages from 2,483 non-sockpuppet accounts. The first contribution was made on 2001 December and the last one was made on 2023 January. Out of 106,796 contributions, 89,625 contributions were added or edited comments, 11,364 contributions were

about deleting comments and hence had no comment content and 5,807 contributions were about moving pages, hence they also had no comment content. The content of contributions of non-sockpuppet accounts is summarised in the figure 3.1b.

3.2.4 Metadata

3.2.4.1 Account registrations ("sock_registrations.json", "control_registrations.json")

Yamak's non-textual model's [50] features, more precisely "The interval between the user's registration and their first contribution" feature, required to know the registration time of each account. The registration time of each account was retrieved by using MediaWiki API:Users endpoint [15] in the form:

```
https://en.wikipedia.org/w/api.php?action=query&list=users&ususers
=<account_usernames>&usprop=registration&format=json
```

As before, the field "ucuser" contained 50 (max limit) usernames combined with the "|" symbol in order to decrease the number of requests. Account names mapped with the retrieved registration times were saved into "sock_registrations.json" and "control_registrations.json" files.

3.2.4.2 First contribution's timestamp ("control_first_contribs.json")

The same feature "The interval between the user's registration and their first contribution" required to know the timestamp of the first contribution of each user. For sockpuppet accounts, this information was already present in the "socks_with_contribs.json" file - since for each account contributions were gathered by looking at 500 contributions ordered from the oldest to the newest ("bkdir=newer"), the very first contribution was always included. However, the situation was different for non-sockpuppet accounts - their contributions were retrieved only for the time period when a matching sockpuppet account was active (effect of controlling for activity range). This means that the very first contribution might not be included in the "control_with_contribs.json" file. Because of this reason, MediaWiki API:Usercontribs [14] endpoint was used again, this time without controlling for activity range and retrieving only 1 oldest contribution for each control account. The usernames mapped to the retrieved timestamps were saved in the "control_first_contribs.json" file.

3.2.4.3 Talk page's category ("title_categories.pickle")

RQ3a) which examined how well models generalise on unseen topics, required to know the topic of every talk page. The majority of talk pages are assigned some topic category(-ies) in Wikipedia [38]. Topic categories of a page can be obtained via MediaWiki API:Categories endpoint [11]. The request of the form:

```
https://en.wikipedia.org/w/api.php?action=query&prop=categories
&cllimit=500&titles=<talkpage_titles>&format=json
```

returned all the categories talk pages belong to. Talk page names mapped to all the categories they belonged to was saved in the "title_categories.pickle" file. Note that talk pages which didn't have any category were not included in this file.

3.3 Cleaning data

3.3.1 Cleaning main files

Main files, namely, "socks_with_contribs.json" and "control_with_contribs.json", were cleaned only minimally in order not to lose important information. Initially, all the contribution fields (pageid, revid, parentid, title, timestamp, sizediff and comments) were checked if they contain the expected type and if all the values are not null. All fields in both files passed this test. However, the "comments" field still needed some cleaning because it contained raw people's comments. Several stages of cleaning were applied to this field:

1. Substituting actual signatures with "<signature>" token. Comments in Wikipedia can be signed with the user signature whose standard format is "[[User:username| username]] ([[User talk:username|talk]]) + date". As it is seen, signature depends on the account's username and hence it might influence comment's textual features like length of the comment, frequency of certain characters, etc. Hence, the account's username might become a hiding variable for signed comments in textual models. In order for this not to happen, signatures in standard and some common alternative forms were substituted with the "<signature>" token which is independent from the account's username.
2. Substituting automatic signatures with "<auto-signature>" token. Comments can also be signed by the SineBot [33] which automatically adds a signature to unsigned comments. The format of the added signature is "Preceding [[Wikipedia: Signatures| unsigned]] comment added by [[User:username]]". Again, this signature might influence textual comment's features as it includes a username, hence these signatures were substituted with the "<auto-signature>" token.

3.3.2 Cleaning metadata files

No cleaning was needed for the metadata files. Looking more specifically:

- "sock_registrations.json" and "control_registrations.json" - it was checked if all accounts have their registration timestamp recorded. Even though some of the accounts had "invalid" or "null" values for their registration timestamp, a decision was taken to account for this issue when doing the computations (see 4.1.1.4), rather than to change it directly in files.
- "control_first_contribs.json" - no cleaning was needed as all the control accounts had the timestamp of their first contribution registered.
- "title_categories.pickle" - no cleaning was needed because only those titles which had at least one contribution were saved to the file.

Chapter 4

Comparing different model groups on standard metrics (RQ1)

This chapter describes RQ1, more precisely what methodology was used to answer RQ1 and what the accumulated results were. The aim of this chapter (as well as the RQ1 itself) is to directly compare sockpuppet detection models described in literature and taken from different feature groups: textual features, non-textual features and combination of textual and non-textual features. The final goal is to select the most promising model group based on the most relevant metrics for the sockpuppetry problem.

4.1 Methodology

The process of answering RQ1 can be split up into 2 main stages: models' recreation and models' evaluation. Since direct comparison was the main goal, models were recreated by ensuring the identical setup for all the models and they were evaluated by creating the identical evaluation environment for all the models. Without this "uniformity principle", the final metrics of different models would not be directly comparable because they might be influenced by the other factors (different model type, different data split, etc).

4.1.1 Recreating models

4.1.1.1 Selected models

As already stated, 3 different feature groups were selected for the direct comparison: textual features, non-textual features and combination of textual and non-textual features. Each of the groups needed a representative model, i.e. archetype. The decision was made to recreate the newest model described in each of the groups: for the textual features group - Solorio's model [23], for the non-textual features group - Yamak's model [50], for the combined features group - Yu's model [53]. The main reason for selecting to recreate the newest model was because newer studies are expected to use newer methods and account for previously conducted studies by oftentimes building on these studies and consequently improving them. Hence, in general, newer studies are expected to be more promising.

Since none of the studies, described in the 2.2 section, had published the full code of their models, I needed to recreate all of the selected models from scratch. However, it is important to note that none of the models were recreated fully accurately because of the modifications needed to ensure the "uniformity principle" (see 4.1.1.2) and individual limitations (see 4.1.1.3, 4.1.1.4, 4.1.1.5).

4.1.1.2 Identical setup

As already mentioned in 2.3, one of the reasons why models described in the literature were not directly comparable was varying model type. Looking at the models chosen to be recreated, one can indeed notice that they are of different types: Yamak's and Yu's models are binary classification models, whereas Solorio's model matches sockpuppet accounts with their sockmaster (i.e. authorship attribution). RQ1 requires all the models to have the identical model type. The decision was made to convert all of the models to binary classification models, i.e. convert the Solorio's model to the binary classification model and leave the remaining 2 models as-is. The main reason why binary classification was chosen over account matching was because converting the model from account matching to binary classification is feasible, whereas the reverse is not really doable - if a model can only output if an account is a sockpuppet or not, there is no straightforward way to match accounts belonging to the same person. Additionally, looking at the Solorio's study limitations (see 2.2.1), it was outlined that sockpuppetry problem is not really compatible with the used authorship attribution (AA) technique, because AA expects long texts of inputs, whereas people make only limited number of comments in the discussion area. This was an additional motivation to modify Solorio's original model.

4.1.1.3 Textual features model

The textual archetype was mostly based on Solorio's study [23], however, it required some modifications, e.g. when converting it to the binary classification model.

Converting to binary model: There were several ways to convert the original account-matching model to the binary model. One way was to try and match each sockpuppet account with every possible sockmaster. If there is a match, it means that an account was a sockpuppet, otherwise, it is a legitimate account. However, the original Solorio's model already suffered from a high computational cost (see 2.2.1) and this setup increases the runtime to classify one account by n times, where n is the number of sockmasters. Also, if the sockmaster of some sockpuppet account is not in the dataset, the model would incorrectly classify this sockpuppet as a legitimate account. Therefore, this modification version had major flaws and could not be adopted. Instead, a simpler conversion was done. AA features, described in the original study, were simply used to convert each comment to a feature vector. Then a portion of labelled comments were used as a training data. After the training phase, the model was able to classify each comment, converted to the feature vector, as coming from the sockpuppet or a legitimate account. These decisions were afterwards used to classify users to sockpuppets and legitimate accounts (see 4.1.1.3). This setup resulted in a low computational cost and had the potential to catch all sockpuppet accounts regardless if their sockmaster was

in the dataset or not. On the negative side, this setup relied on the assumption that AA features were compatible with the user classification problem into sockpuppets and legitimate, even though this was not verified.

Features: All of the 239 features, used in the original study, were successfully recreated. The used text features can be grouped into sentence, token, character and speech level features and are listed in the table 4.1. Most of the features are self-explanatory and their explanations can also be found at the original Solorio's paper [23]. The only ambiguous features were features which originally included the word "frequency" in their name such as "Question frequency" or "Tab frequency". According to the Cambridge dictionary [2], frequency is "the number of times something happens within a particular period." Applying this definition to the given context, frequency of some feature might mean the number of times that feature happens within a specified number of words, however this number was not outlined for any of the features in the original paper. Because of the lack of information, the decision was made to substitute the word "frequency" with "proportion" in the features, i.e. to divide feature's count by the total count of either sentences, tokens, characters or speech parts depending on which level (sentence, token, character or speech) a feature belonged to. E.g. sentence-level feature "Question frequency" was interpreted as "Question proportion" which equals question sentences count divided by the total number of sentences in the comment.

Sentence level	Token level	Character level	Speech level
Sentence count without capital letter at the beginning	Big "I" pronoun frequency	Frequency of letters (26 features)	Parts of speech (POS) tags frequency (36 features)
Total number of sentences	Total number of tokens	Total number of characters	
Question frequency	Small "i" frequency	Total alphabet count	
Sentence with small letter frequency	Function words frequency (150 features)	Two/three continuous punctuation count	
	All caps letter word count	Total punctuation count	
	Total contraction count	Emoticons count	
	Words without vowels	Happy emoticons count	
	"He" frequency	Quotation count	
	"She" frequency	Parenthesis count	
	"A" error frequency	Alpha frequency	
	"An" error frequency	Digit frequency	
		Uppercase frequency	
		White space frequency	
		Tab frequency	
		Full stop without white space frequency	

Table 4.1: Features used in the textual model.

User classification: The recreated original model actually performed the classification on the comment level. To translate this into user classification, the two-step scheme proposed in the original paper was adopted. In order to classify a user, the first step was to gather all the user's comments and get their individual predictions. In the second step, the majority (i.e. the most common) prediction is calculated and assigned as the final classification for a user. This scheme is summarised in the fig. 4.1.

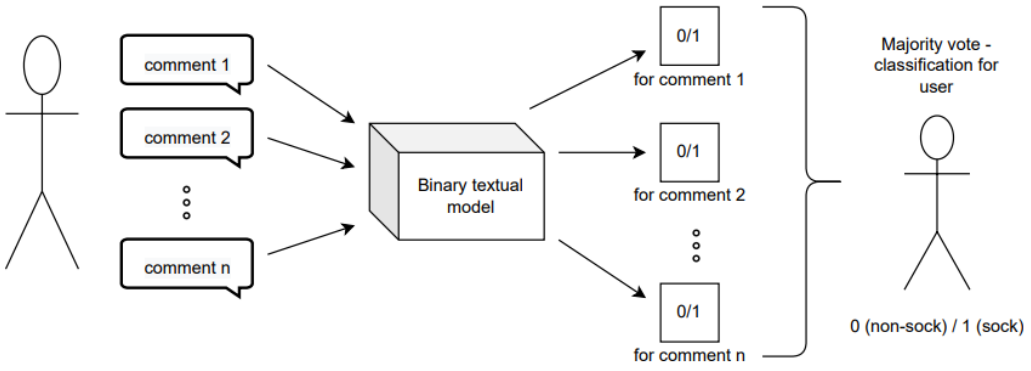


Figure 4.1: User classification scheme of a textual model.

4.1.1.4 Non-textual features model

The non-textual model archetype was highly inspired by the Yamak's model [50] and the main mismatch arose because the dataset used in this study (see 3) was considerably smaller (in terms of the number of namespaces it covers) than dataset originally used by Yamak, hence some of the features could not be recreated.

Features: Originally Yamak used 11 non-textual features: 5 of them were successfully recreated in this model, whereas 6 were lost because of the dataset limited to the discussion area only. 5 recreated features include: the number of user's contributions in the talk pages, the average of bytes added and removed from each revision, the average number of contributions in the same article and the interval between the user's registration and their first contribution. The latter feature required special attention. As already noted in the 3.3.2, some registration dates, stored in the "sock registrations.json" and "control registrations.json" files, were either "null" or "invalid". For such accounts whose registration date is unknown (these usually were not registered accounts), the interval between the user's registration and their first contribution feature should not impact the classification output. Therefore, for these accounts the average feature's value calculated from all the registered accounts (including both sockpuppets and control users) found in the dataset, was reported in this field as this was the most neutral value. Coming back to 6 not recreated values, they include the number of user's contributions in the article/user page/user discussion page/project namespace/all other namespaces combined, and the frequency of revert after each contribution in the articles. One can observe that all these features are connected to namespaces other than the article discussion area (i.e. talk pages), so recreating them would require collecting additional data from various namespaces and would bring back the problem of different evaluation datasets among different models which make the final results not directly comparable. Even though missing some features results in data loss, it also allows performing additional analysis. It was already noted in the 2.2.2 that it would be "interesting to investigate if it is possible to restrict the data to only a few namespaces and still get satisfactory results." If the results are similar, then it is debatable how useful the skipped features are, because collecting them costs much, and if the contribution

from them is only negligible, then it's probably better to abandon them.

User classification: The non-textual model, as well as the original Yamak's model, performed the classification directly on the user level. All the contributions from a user were gathered into one "document" which represented a user and then this document (as well as the user) was classified as a sockpuppet (1) or a legitimate account (0). The scheme is summarised in the 4.2.

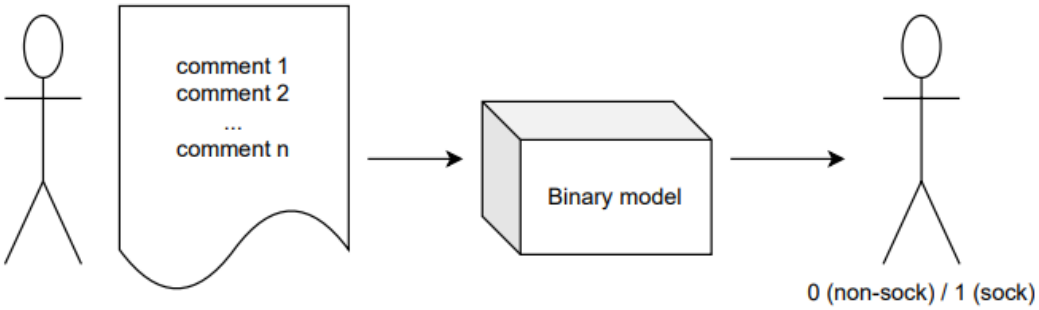
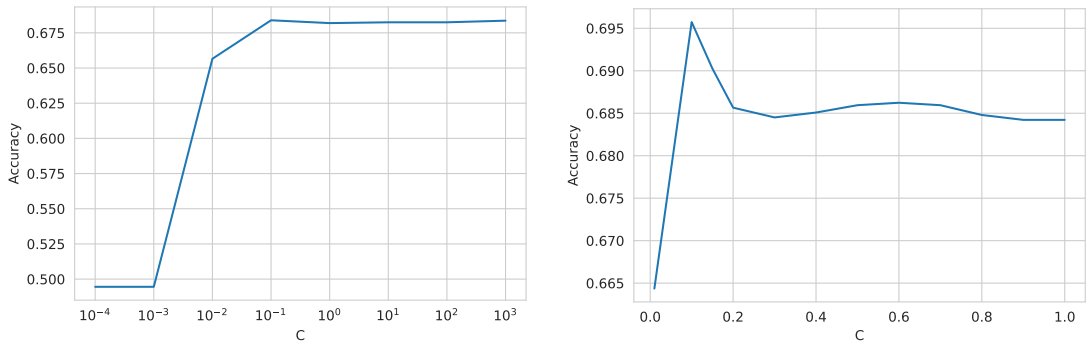


Figure 4.2: User classification scheme of non-textual, combined models and transformers.

4.1.1.5 Combined features model

Combined features model integrated textual and non textual features by following Yu's study [53].

Features: Since Yu's study did not provide the exhaustive list of the combined features or code of the model, it was impossible to know what features originally were combined. Because of this lack of information, the decision was made to simply concatenate features used in the previous textual (see 4.1.1.3) and non-textual models (see 4.1.1.4) of this study. Adding 239 textual features with 5 non-textual features would result in 244 features, however, Yu also completed adaptive feature selection via Logistic Regression with Lasso regularisation (i.e. L1 regularisation), which decreases the feature space. When reading Yu's paper [53], it seems that authors might have used some modification of the traditional L1 regularisation because provided formulas slightly differ from the traditional L1 regularisation, but the provided formulas also contain many mathematical errors. For instance, in (19.5) function in the original paper [53], Y is subtracted from 1, even though Y is a vector and subtracting vectors from scalars is not mathematically defined. Also, sum and product operator's boundaries are incorrect in (19.4) and (19.5) - they should include all the terms as in traditional log-loss function, not only the first term. Taking into account all of the errors, one can claim that authors do not have much expertise in logistic regression with L1 regularisation. Therefore, it becomes hard to trust the provided formulas and, in general, distinguish which alterations in formulas is the author's conscious choice to modify the traditional approach, and which ones were done due to simple lack of expertise. Hence, it seemed safer to complete adaptive multi-source feature selection by executing the traditional Lasso regularisation found



(a) Choosing the best magnitude.

(b) Choosing the best individual value.

Figure 4.3: Lasso regularisation (C) effect on the classification accuracy.

in the sklearn.linear_model package. Regularisation's goal is to prevent overfitting by decreasing model's variance, however there is no standard metric for logistic regression to measure its variance (as opposed to R^2 score in linear regression). Hence, some other metric, connected with overfitting, was needed. Since this is a classification problem, the model's accuracy can signal about overfitting - if the model is overfitted, it generalises bad and has a low accuracy on validation or test dataset. Therefore, C was tuned by maximising accuracy on the validation set in the 4-fold cross validation. In the first stage, the best magnitude of regularisation parameter was found - it was in the order of 10^{-1} (see 4.3a). In the second stage, several values of this order were tried and C = 0.1 yielded the highest classification accuracy (see 4.3b). Applying Lasso regularisation with C = 0.1 removed 72 features and left us with a total of 172 combined features.

User classification: The combined model used the same classification scheme as the non-text model, namely, classification on the user-level which is depicted in the 4.2.

4.1.2 Evaluating models

It was crucial to ensure an identical evaluation environment for all the models, so that differences found in models' results can be explained solely by their different methodologies, not by some confounding variables. Because of this reason, all of the models were trained and evaluated on the same dataset described in chapter 3. The dataset was split 3 times (see 4.2). Each model was trained and tested on each of the splits and afterwards average metrics were reported. This was done in order to check if a model's performance is independent of the data split - ideally, we want a model to be stable and perform equally well regardless of what data it gets.

For every archetype that was recreated, an attempt was made to identify the most suitable machine learning model. Because of this reason, a number of ML models were tried, namely, Support Vector Machine (SVM), Random Forest (RF), K-nearest neighbours (kNN), Naive Bayes (NB) and Adaptive Boosting (ADA). Some of them required scaled data, i.e. normalising the data. Data were scaled for linear models (SVM, kNN) and used as-is for non-linear models (RF, NB, ADA). Importantly, normalisation was completed after the dataset was split into training, development and testing sets.

If data was scaled before it, then some information from the testing set would appear in the scaled training set and thus the model would become aware of it. Incorporating any data from the test set during the model's training phase might result in unfair increase of a model's performance. Using the ML models also required completing hyper-parameter tuning. Different hyper-parameter combinations were exercised with the help of scikit learn's GridSearchCV [18]. Scikit learn's GridSearchCV automatically performs cross-validation. Time-consuming models were evaluated using only 3-fold CV, whereas more time efficient models were evaluated using 5-fold CV.

	Training set (70%)		Testing set (30%)	
	#sockpuppets	#control users	#sockpuppets	#control users
SPLIT 1	1756	1720	727	763
SPLIT 2	1721	1755	762	728
SPLIT 3	1736	1740	747	743

Table 4.2: Description of different data splits.

4.2 Results

The average results from 3 splits are summarised in the following tables: textual models in 4.3, non-textual models in 4.4 and combined models in 4.5. The extended version of all tables, also reporting standard deviation, can be found in the appendix (A.2).

Models were compared on FPR, TPR, precision (macro average) and f-score (macro average), which arguably are the most important metrics for a sockpuppet detection model. Formulas of the metrics can be found at A.1. TPR (recall) is relevant because the ultimate goal of a model is to catch and block the majority of sockpuppet accounts, so the system should have a high recall. Nevertheless, FPR is just as important. The significance of FPR lies in the fact that being accused of sockpuppetry is a serious action, which might result in an account's block. Hence, it is crucial to minimise FPR by preventing blocking legitimate accounts. As the cost of false positives is high, precision also automatically becomes a metric of interest - a model must be certain when making a positive prediction. Additionally, having high recall may lead to a very low precision as recall and precision are often trade-offs. For example, if every account in our dataset (50% sockpuppets, 50% legitimate accounts) is predicted to be a sockpuppet, then this system would have a recall of 100% and precision of only 50%. Hence, precision and recall need to be balanced and f-score is exactly the metric which integrates both precision and recall and harmonises them.

The total time needed to train the model (this does not include parameter tuning) and the total time a model took to classify all the instances was also reported in seconds.

4.3 Interpretation of results

Looking at the results, one can see that recreated models based on textual features (see 4.3) turned out to be incompatible with our problem - since it can only detect at

Textual features models

	SVM	RF	NB	KNN	ADA
TPR AVG	0.329	0.239	0.378	0.385	0.268
FPR AVG	0.235	0.068	0.294	0.208	0.132
Precision AVG	0.559	0.665	0.548	0.604	0.607
F-score AVG	0.524	0.528	0.528	0.569	0.523
Training AVG (in s)	71.520	61.368	0.895	0.065	37.531
Testing AVG (in s)	22.226	1.130	0.395	77.503	1.571

Table 4.3: Results of models based on textual features.

Non-textual features models

	SVM	RF	NB	KNN	ADA
TPR AVG	0.618	0.693	0.912	0.709	0.694
FPR AVG	0.324	0.300	0.781	0.388	0.278
Precision AVG	0.681	0.697	0.627	0.666	0.708
F-score AVG	0.626	0.696	0.506	0.657	0.708
Training AVG (in s)	0.678	0.456	0.003	0.005	0.150
Testing AVG (in s)	0.154	0.029	0.001	0.116	0.013

Table 4.4: Results of models based on non-textual features.

most 38.5% of sockpuppets (TPR of kNN), the majority of sockpuppets would remain unblocked, which basically defeats the purpose and makes the remaining metrics irrelevant. This might have happened because, in order to allow for direct comparison, the original model's type was changed to binary classification and features, which were originally used for AA, now were used for account classification purposes. Even though those features were suitable for the AA problem in the original study (the original model [23] managed to achieve satisfactory results), they might be insufficient when classifying users to sockpuppets and legitimate accounts.

Archetype based on non-textual features (see 4.4) looks more promising. In the original study (see [50]), the best performing ML models were NB and RF. In this study, NB managed to reach a very high recall of 91.2%, however its FPR of 78.1% is unacceptable, therefore, this model cannot be considered as the best. It is interesting to notice that NB had a FPR of 3% in the original study, which is very different from FPR of 78.1% of NB in this study. In general, results are not expected to accurately match the results of the original studies because of all the modifications and different evaluation dataset, however similar trends between 2 studies are still expected to be observed. This mismatch highly increases the possibility of the mistake which was mentioned in the 2.2.2: "there is a slight possibility that authors mistakenly reported FNR instead of FPR." Looking at the second best model in the original study, namely, RF, in this study it achieved good but not the best results between all non-textual ML models. From the remaining models, the ADA model has the lowest FPR (27.8%), the highest precision (70.8%) and f-score (70.8%) and second highest TPR (69.4%). Hence, non-textual archetype achieves the best results when implemented as ADA and non-textual model's implementation via ADA can be considered as a real candidate for the best solution.

Combined features models

	SVM	RF	NB	KNN	ADA
TPR AVG	0.709	0.776	0.910	0.654	0.749
FPR AVG	0.328	0.230	0.730	0.395	0.265
Precision AVG	0.691	0.773	0.653	0.629	0.742
F-score AVG	0.690	0.773	0.543	0.629	0.742
Training AVG (in s)	32.271	17.383	0.012	0.004	1.039
Testing AVG (in s)	0.261	0.282	0.005	0.996	0.026

Table 4.5: Results of models based on combined features.

Even though textual features on its own were not sufficient in the sockpuppetry detection problem, adding them with non-textual features resulted in highly improved results. From the combined features group (see 4.5), RF has the best metrics among all the ML models (NB is instantly rejected because of an unacceptable FPR - identical problem as for non-textual NB). RF model managed to achieve 77.6% recall, 77.3% precision, 77.3% f-score and FPR of 23%. All these results are really satisfactory, similar to the original study's results (see 2.2.3) and improves the results of non-textual model's implementation via ADA by 5-8%. Hence, the combined archetype's implementation via RF is even a better candidate if considering only the standard metrics.

To conclude, this section's aim was to review results of the textual, non-textual and combined model and to identify the highest potential having solutions. The best found solution in terms of recall, precision, f-score and FPR was the combined-features model implemented via RF. Nevertheless, the primary section's aim arguably was not fully achieved because textual archetype's models were not actually included in the comparison. It was only identified that the recreated models turned out to be incompatible with the sockpuppetry problem. However, this raises a question if the archetype, which was selected for recreating, was a good representative for the textual model's group. First of all, as already noted, the original archetype's type was changed to binary classification, even though AA features might not be suitable for the account classification problem. Also, the original study [23] is 10 years old. During this 10-year period, there have been many innovations in the text technologies field such as word2vec model allowing to capture semantic relationships between words [21] or attention mechanisms allowing to focus on specific parts of a sequence when processing it [26] which eventually led to transformers. Transformers (GPT, BERT, etc.) are a deep learning model which demonstrated outstanding results on various NLP tasks and "has brought natural language processing (NLP) to a new era." [22] In RQ2, transformers will be implemented as a representative model for the textual-features group as they are the most promising state-of-the-art solution suitable for the comments' setup. It is anticipated that transformers will turn out to be compatible with the sockpuppetry problem and they will achieve better results for the textual-features group or even among all of the groups.

Chapter 5

Sockpuppet detection with transformers (RQ2)

This chapter describes RQ2 - how transformers were used for the sockpuppet detection problem and what results they managed to achieve when evaluated on the same dataset as models described in RQ1. After transformers' results were accumulated, the process of choosing the most promising model group based on standard metrics was revisited by including transformers in the selection process.

5.1 Motivation

As already noted in the previous section 4.3, textual-features group needed a better representative model. The decision was made to switch to transformers for the textual group because of several reasons. The first criterion when choosing a new model, was that this model should be compatible with the current problem - to classify users as sockpuppets or legitimate accounts by analysing their comments in English Wikipedia's talk pages. Comments belong to sequential data, and transformers are well-suited for analysing sequential data. Transformers are also known to work well on a number of tasks, including the needed task of text classification, because of their universal architecture: "Transformer does not make any assumption about how the data is structured ... This effectively makes Transformer a very universal architecture." [20] Secondly, transformers were chosen due to their promising results on a similar problem of detecting fake accounts in OSN [27], as previously mentioned in 2.2.4. Even though the main emphasis of this work was to present that UCred model incorporating RF, LSTM neural network and RoBERTa transformer is the most effective when catching fake accounts, this study also reported metrics if transformers were used alone. Using the RoBERTa transformer alone caused only 1.5-3% drop in classification accuracy, precision, recall and f1 score if compared with UCred's results. Since transformers were successful in detecting fake accounts, they hold a great potential for detecting sockpuppets which are a subset of fake accounts. Finally, to the best of our knowledge, no study completely relied on transformers to detect sockpuppet accounts specifically in Wikipedia. Using something novel makes this study interesting and contributes to the research world.

5.2 Methodology

5.2.1 Selecting transformers

In this study, the Hugging Face transformers [5] library was used to download the needed transformers and fine-tune them on the collected sockpuppetry dataset. This library was chosen because it is flexible and easy-to-use - the author of this study did not have any prior experience with PyTorch or TensorFlow, so some beginner-friendly library was needed, which is exactly what Hugging Face library is about.

Hugging Face library offers thousands of transformers, however, evidently it was only feasible to test a few transformer models in RQ2. Therefore, a considerate selection procedure was needed. Selection criteria included the popularity of the transformer and its exposure to Wikipedia's texts. The reasoning behind choosing the most popular transformers was that if those models are actively used by other people, it indicates that they perform well on the range of problems and hence there is a high probability that they will succeed in the sockpuppets detection problem as well. Exposure to Wikipedia's texts was needed because the dataset of this study consists entirely of texts taken from English Wikipedia. Hence, out of all the possible transformers, 4 most downloaded (as of 14/02/2023, see A.1) transformers trained on the "Wikipedia" dataset was chosen and downloaded, namely, RoBERTa, DistilBERT, BERT and XLNet.

5.2.2 Fine-tuning transformers

5.2.2.1 Process overview

The downloaded transformers already come pre-trained on large amounts of text in a self-supervised manner. This type of training allows for a model to get a statistical understanding of the text it was trained on, however pre-training does not prepare a model to perform a specific task such as text classification. Therefore, the downloaded general transformer should also go through fine-tuning which is training a model on a dataset specific for a task. However, in order to complete fine-tuning, text in a dataset should be preprocessed with a suitable tokenizer.

5.2.2.2 Preprocessing data

The first step of the data preprocessing was to gather each user's comments into a single document (i.e. 1 user = 1 document). Merging comments into 1 document might arguably reveal the number of comments a user made, which is a non-textual feature, because, if all the comments are of a similar size, the higher the number of comments a user made, the longer the accumulated document should be. However, transformers' tokenizers do padding which makes all of the documents of the same length, therefore the risk of including non-textual features for a textual model is eliminated. Accumulating comments to individual documents enabled transformers to perform direct classification on a user level, i.e. transformers followed the same classification scheme as non-textual and combined models (see 4.2). Following the same classification scheme makes different model groups even more comparable.

Once comments were accumulated to individual documents, they were tokenized with the suitable tokenizer. Each transformer required a different tokenizer, for example, for RoBERTa "RobertaTokenizerFast" was used, for BERT "BertTokenizer" was used, etc. Despite the fact that tokenizers were different, "padding" and "truncation" parameters were always set to true for all the used tokenizers. Setting these parameters to true meant that firstly comments' documents were appended with additional tokens in order to match the length of the longest document in a batch and afterwards the appended documents were truncated to the maximum allowable input's length for a model.

5.2.2.3 Fine-tuning via Trainer class

To complete the fine-tuning, Hugging Face's Trainer class [6], which provides functions to facilitate the fine-tuning of models in PyTorch, was used. This class is highly dependent on the "TrainingArguments" class which is used to specify parameters for the training. For the majority of training parameters, the default values were used, but values for "per_device_train_batch_size", "warmup_steps" and "weight_decay" were set to 4, 500 and 0.01 respectively because these values were reported to achieve satisfactory results on similar tasks [17]. In reality, to find the best results, one should do parameter tuning for the training arguments, however due to time constraints (it took 20-30 min to train 1 model) and limited computing resources (Google Collab's GPU was used to train the models, however, it is only available for at most 12 hours and has an exponentially increasing reset time), the decision was made to choose the arguments from theoretical rather than experimental side. It is important to note, that during training, the model was evaluated on the development set, which was 30% of randomly sampled training data, however during testing, it was evaluated on the unseen testing set.

5.2.3 Evaluating transformers

The most important thing was to recreate the identical evaluation environment which was used to evaluate non-textual and combined ML models (see 4.1.2). This was done in order to directly compare transformers with previously recreated non-textual and combined archetypes and, in case of any performance differences, explain them solely by the different model's architecture, rather than some confounding variables. Therefore, all of the fine-tuned transformers were evaluated on the same dataset (see 3) which was split 3 times (see 4.2). Evaluation, i.e. testing, was completed via the Trainer's "evaluate" method. Identically as for non-textual and combined models, transformers were trained and tested 3 times (on each of the splits) and, subsequently, the metrics obtained from each split were averaged and reported.

5.3 Results

Transformers' results are summarised in the table 5.1 (the extended version of results can be found at A.4). The table reports the same metrics of interest, namely, averaged recall, FPR, precision (macro average) and f-score (macro average). Average time to train the transformer (Training AVG) and time to classify all test instances (Testing AVG) are also reported because they hold a significant importance for transformers.

Transformers				
	RoBERTa	DistilBERT	BERT	XLNet
TPR AVG	0.844	0.766	0.744	0.632
FPR AVG	0.098	0.278	0.274	0.385
Precision AVG	0.893	0.727	0.725	0.646
F-score AVG	0.867	0.746	0.734	0.566
Training AVG (in s)	738.605	371.902	726.550	1321.469
Testing AVG (in s)	42.261	22.729	45.376	139.208

Table 5.1: Results of transformers (belong to textual group).

5.4 Interpretation of results

Looking at the 5.1, one can immediately notice that transformers performed significantly better than previous textual models (see 4.3) which proves that previously selected models were poor representatives for the textual model group and that textual features in general are compatible with the sockpuppet detection problem. Performance of all the transformers, except for maybe XLNet, was very satisfactory. RoBERTa demonstrated outstanding results. So far, the best model was the combined model's implementation via RF with 77.6% recall, 77.3% precision, 77.3% f-score and FPR of 23%. RoBERTa outperformed this model in all the aspects - RoBERTa reached 84.4% recall (improvement by 6.8%), 9.8% FPR (improvement by 13.2%), 89.3% precision (improvement by 12%) and 86.7% f-score (improvement by 9.4%). These results are very promising and the standard deviation of all the metrics is only 0.5-1% (see A.4), which shows that the achieved results are stable, i.e. they are not accidentally caused by a very good split between training and testing data or some coincidental factors, and that the reported results can be trusted.

When looking at the reported metrics, one could immediately claim that RoBERTa's model turned out to be the best performing one on sockpuppetry problem and has the highest potential to be integrated into Wikipedia. However, arguably comparing models on recall, FPR, precision and f-score is a very theoretical comparison suitable for any type of problem. Relying only on standard metrics might not be sufficient to select the best model for sockpuppet detection in Wikipedia. Therefore, it's important to evaluate the best found models from a more practical perspective that takes into account factors such as limitations of selected models and models' maintenance problem. This leads to RQ3. RQ3 analyses how well 2 so far the best models, namely, RoBERTa and combined model via RF, classify data taken from unseen topics (addresses limitation of both models as they both rely on textual features) and unseen time periods (addresses maintenance problem) which were not present during the training (or fine-tuning) phase. It is anticipated that, performing additional evaluation on more practical dimensions, will allow proposing a solution, which has the highest probability to achieve good practical results if integrated into Wikipedia for sockpuppet detection, with a greater confidence.

Chapter 6

Practical evaluation of top-performing models (RQ3)

In this chapter, models that were found to be the best when evaluated solely on the standard metrics, are evaluated further from a more practical perspective. The final goal of this chapter, as well as this study, is to find a model which has the greatest potential of achieving satisfactory results in sockpuppet detection in Wikipedia.

6.1 Motivation

As already mentioned in 5.4, it is crucial to evaluate models not only on standard metrics, but also from a more practical perspective. When choosing additional practical dimensions to evaluate models on, the goal was to foresee what factors could have a significant influence on their performance and to check model's performance after simulating unfavourable conditions for those factors.

The first factor to consider is that both RoBERTa's model and the combined model rely on the textual features when classifying users - RoBERTa's model entirely, combined model partially. Textual data is very dynamic and easily manipulable, hence it is impossible to introduce the model with all the possible variations of texts during training. For instance, new variations of texts can be created by introducing new topics which were not present when training the model. As time advances, new article topics continue to be generated - from 2018, each year around 200,000 new articles are added to Wikipedia [43]. Given the great number of articles, many of them apparently introduced new topics. Since the textual data is of very dynamic nature, the ideal sockpuppet detection model should be able to correctly classify users whose comments content is different from content of comments found in the training set. This invites to test how well 2 best models (RoBERTa and combined model via RF) generalise on unseen topics, i.e. topics a model was not exposed to during the training phase (RQ3a)). In this study, this feature will be interchangeably called "topic generalisability". As reflected in [54], topic generalisation, also known as domain generalisation, is a crucial dimension to evaluate learning algorithms on as "generalization to out-of-distribution data is a capability natural to humans yet challenging for machines to reproduce."

Another practical problem is retraining the models. In the ideal world, these models should be periodically retrained, however retraining models requires human experts, much computing power and time, all of which are limited resources. Time concern is especially applicable for RoBERTa because transformers take much time for fine-tuning - RoBERTa has 43 times longer training (i.e. fine-tuning) time than the combined model via RF (see 4.4). Because of these practical issues, updating models may occur less often than it is recommended and models trained on old data inevitably will have to classify more recent data. This leads to RQ3b) which analyses how well RoBERTa and combined model, trained on data from one period, can classify more recent data, i.e. model's "temporal generalisability" will be tested. Temporal generalisation is a classical dimension to evaluate models working with textual data, for example, see [16].

6.2 Methodology

6.2.1 RQ3a

6.2.1.1 Getting topic of a talk page

For each talk page, categories it belongs to were retrieved via MediaWiki API:Categories [11] endpoint as already described in 3.2.4.3. It was observed that not all the retrieved categories are topic-oriented, some of them are only concerned with talk page's quality or importance. For instance, "Talk:Donald Trump" [36] has the category "B-Class politics articles" which indicates that this page is of B-Class quality and also the category "High-importance politics articles" which informs about the page's importance. Therefore, the goal was to find a category type which is only concerned about the page's topic. WikiProject [48] type categories precisely embodied that characteristic as WikiProject type categories connect talk pages of the same topic. Some examples of WikiProject titles include "WikiProject Television articles", "WikiProject Biography articles", "WikiProject United States articles", etc. Since WikiProjects are clearly centred around a specific topic, only WikiProject type categories were filtered out and used as a topic(-s) of the talk page.

It is important to note that a vast majority of the pages were given a topic, although not all of them - of the 67,814 distinct talk pages, 55,459 (81.8%) were eventually assigned a topic. Out of 18.2% of pages without a topic, 7.6% could not be assigned a topic because they were not assigned to any of the categories in Wikipedia. Reasons why some talk page did not have a category include the fact that talk pages are not required to have a category, e.g. "Talk:ARM DynamIQ" [31] has no categories, or because the talk page was moved and the current title did not exist, e.g. "Talk:Michael Bell Cox" was moved to "Talk:Archdeacon of Raphoe" [32]. The latter reason is the limitation of the MediaWiki API:Categories endpoint - this endpoint can only search for current titles but it cannot redirect to a new title. The remaining 10.6% of unassigned pages could not be assigned a topic because even though they belonged to some of the categories, none of them were of the WikiProject type. One alternative topic source for unassigned talk pages would be to analyse categories of the corresponding article, however categories of articles are too specific and that would create an imbalance between 81.8% of talk pages which were already assigned a few broad topics and the remaining 18.2% of talk pages

which would be assigned to many very specific topics. Therefore these 12,355 talk pages (18.2%) ultimately were left without a topic because alternatives topic sources seemed insufficient.

6.2.1.2 Creating training and testing sets

Once the majority of talk pages were assigned a topic, the next step was to divide users between training and testing sets. It was impossible to create a testing set which would consist of users contributing only to topics which a model was not introduced to during the training, i.e. some intersection between the training set's and the testing set's topics was inevitable. A simple example proving the difficulty of this task is as follows: out of 3 users, 2 users (i.e. 66% users) need to be assigned to the training set and 1 user (i.e. 33% users) to the testing set. The contributions of the users are as follows: user A contributed to "Politics" and "USA" topics, user B to "Politics" and "Food" topics and user C to "Nature" and "USA" topics. Regardless of which user is put to the testing set, it is impossible to create a testing set which consists only of topics which a model was not introduced to during training.

One way to solve this problem was to assign each user a dominating topic, in which the majority of the user's contributions were made, instead of considering all of the topics for which a user have made contributions. In the aforementioned example, if user A and user B dominating topic was "Politics" and user C dominating topic was "Nature", then putting user C to the testing set would result in the smallest topic-intersection between testing and training sets. However, in reality users tend to make their contributions in a number of topics instead of concentrating on one topic only. Histogram (see 6.1), which shows what percentage of contributions were made in the user's majority category (most common category for that user's contributions), is skewed to the right. This indicates that the majority of users made only a small portion, specifically 3-30%, of their contributions in their majority topic. Hence the concept of linking each user with the topic they made the most contributions at is not an accurate representation of that user. Relaxing this and allowing a user to be assigned to multiple topics when X percentage of their contributions were made in those topics would also not work. If the goal is to assign each user with at least one topic, then histogram 6.1 shows that the threshold should be set to 3%. If it is acceptable to have a subset of users without any topic assigned to them, then the threshold could be somewhat higher: with 10% threshold, 7% of users would be left with no topic, but with 20% threshold already 30% of users would be left with no topic. However, setting up the main topic(-s) for a user in reality is just applying filtering for categories. If the threshold is too low, most topics will remain, and filtering will have little effect. Therefore, no filtering was applied and topics were considered as-is, i.e. if a user contributed to some topic, they were a representative of that topic, no matter how actively they contributed to it.

Once this was established, topics were randomly sampled one by one until the testing set was of the needed size ($\approx$ 30% users) and of the needed content ($\approx$ 50% sockpuppets and $\approx$ 50% of control users). After sampling one topic, all of the users who made at least one contribution to that topic were added to the testing set. After the testing set was created, the remaining users were added to the training set. From the topic-intersection

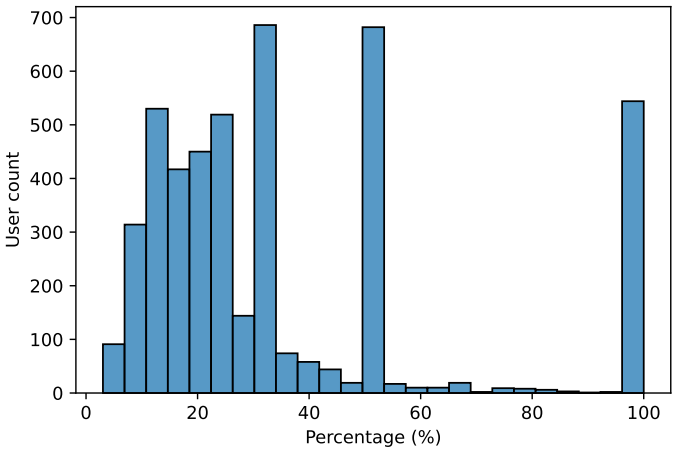


Figure 6.1: Percentage of contributions that were made in the user's majority category.

perspective this means that when forming a testing set, some set of topics were selected and it was ensured that those topics appear only in the testing set, however the remaining topics were not controlled and they could appear both in the training as well as the testing set. This procedure was repeated 3 times, each time with different topics in the random sample - 3 different training and testing sets were created allowing to run models 3 times and report average metrics.

6.2.2 RQ3b

6.2.2.1 Creating training and testing sets

Since timestamp of each contribution was present in the dataset from the very beginning (see 3.2.2.2), answering RQ3b only required to divide users between training and testing sets. The sets were created by simulating the following scenario: models were released at some point in time (let's call it T) and then they were used to classify future data without ever retraining models on newer data. Hence, the testing set consisted of the "future data", i.e. users who contributed after the time T, and the training set consisted of the remaining users. Since the goal was to still ensure that testing set contains only 30% of users (50% of sockpuppets and 50% of control group), some of the users, who contributed after time T, were still assigned to the training set, however all of their contributions which happened after time T were deleted. On the other hand, contributions, which happened before time T, weren't deleted from the testing set to mimic the real word - when classifying some user as a sockpuppet or a legitimate account, one examines all of their contributions, not only the most recent ones.

The date of "01/01/2016" was selected for the value of T. This date has no semantic meaning - it was selected simply because it allowed splitting the users in the most efficient way. Advancing T to more recent years would have introduced the problem of not having enough users in the testing set - testing set required around 750 of each control users and sockpuppets, who had at least one contribution after time T, and the most recent date that had the similar amount of both type of users was "01/01/2016".

If the earlier date was chosen, on the other hand, this would have resulted in deleting many contributions, which happened after this date, from the training set because as the date becomes earlier, the number of deleted contributions increases proportionally.

To conclude, each user was reviewed separately and those users who had comments after "01/01/2016" were added to the testing set. When the testing set became of the desired size and content ($\approx$ 30% of all users equally split between sockpuppets and legitimate users), the remaining users as well as the users, who had no contributions after "01/01/2016", were added to the training set, however without the contributions happening after "01/01/2016". Since the number of sockpuppets who had contributions after this date was slightly more than the needed 30%, this allowed creating multiple sets. Hence, similarly as before, by creating 3 distinct sets for training and testing, the models were executed three times, enabling the derivation of mean metrics.

6.3 Results

Results of both research questions (RQ3a) and RQ3b)) for both combined and RoBERTa models are summarised in the table 6.1. The same metrics of interest, namely, TPR, FPR, Precision and F-score, were averaged from 3 separate runs and reported. The reported metrics were compared with the baseline metrics calculated in the previous research questions - for combined model in RQ1 (see 4.5), for RoBERTa in RQ2 (see 5.1). The observed difference between new and baseline metrics was expressed in brackets in percentages: (new - baseline) * 100%. Differences indicating decline in performance are bolded - for TPR, precision and f-score negative percentage means performance drop, for FPR positive percentage means performance drop.

	Combined model (RF)			RoBERTa		
	baseline	RQ3a	RQ3b	baseline	RQ3a	RQ3b
TPR AVG	0.776	0.796 (+ 2%)	0.786 (+ 1%)	0.844	0.916 (+ 7.2%)	0.860 (+ 1.6%)
FPR AVG	0.230	**0.273 (+ 4.3%)**	**0.543 (+ 31.3%)**	0.098	**0.134 (+ 3.6%)**	**0.128 (+ 3%)**
Precision AVG	0.773	**0.761 (- 1.2%)**	**0.637 (- 13.6%)**	0.893	**0.852 (- 4.1%)**	**0.867 (- 2.6%)**
F-score AVG	0.773	**0.759 (-1.4%)**	**0.614 (- 15.9%)**	0.867	0.882 (+ 1.5%)	**0.866 (- 0.1%)**

Table 6.1: RQ3 results.

6.4 Interpretation of results

In general, one can clearly see that simulating unfavourable conditions resulted in performance drop for both of the models, as expected. All of the metrics, except for TPR, experienced deterioration. Since both TPR and FPR increased, one can claim that models started to classify accounts as sockpuppets in a more lennient manner - this is not ideal, because even high TPR is one of the goals, however the model needs to be confident when claiming that some account is a sockpuppet as the price of FP is high.

Looking specifically at RQ3a), which analysed topic generalisability of models, it is debatable which model experienced a more substantial decline in performance. As for the combined model via RF, this experiment caused a minimal 1.2-2% change

in almost all metrics, except for FPR which experienced 4.3% deterioration. Small effect can be explained by the fact that this experiment was based on alternating textual features and combined model relies on textual features only partially. The combined model's ability to achieve similar performance when generalising on unseen topics underscores its stability, which is a desirable feature for a model. Effect for RoBERTa, which completely relies on textual features, was of a higher magnitude, although not always negative. RoBERTa's TPR increased by 7.2%, however at a price of FPR which regressed by 3.6% and precision which dropped by 4.1%. So even though RoBERTa's model was less stable, some of the changes were positive which somewhat compensated for deteriorated metrics.

On the other hand, it is evident that the combined model's drop in performance is more significant than that of RoBERTa in RQ3b), which analysed temporal generalisability. Looking at the RoBERTa's decreased metrics, its metric deterioration of 0.1-3% seems negligible when compared with the combined model's metric deterioration of 13.6-31.3%. The latter deterioration is unexpectedly high and some analysis was completed in order to understand why. In general, this might be caused either by unsatisfactory dataset, shifts in user commenting behaviour over time or model's incompatibility with the experiment's setup. Out of these reasons, the first 2 seemed improbable, because the RoBERTa model managed to achieve satisfactory results indicating that the dataset was diverse and unbiased towards any specific type of comments and that the user commenting behaviour did not change at least in terms of textual features. Even if commenting behaviour changed in terms of non-textual features, this should not have that drastic effect on the model's performance as non-textual features constitute the minority of the features in the combined model. Hence, the model's incompatibility seemed like the most probable reason and to test this, the experiment's setup was slightly changed - contributions that happened after "01/01/2016" were not deleted from the training dataset. Rerunning the model gave the following averaged results: TPR - 0.807 (+3.1%), FPR - 0.372 (+14.2%), precision - 0.726 (-4.7%) and f-score - 0.717 (-5.6%). The significant improvement in results by including "future" contributions in the training set implies that the combined model needs to see the entire commenting history of a user when learning, whereas the RoBERTa model can effectively learn from partial data. This is one more advantage of RoBERTa. Nevertheless, even when including the entire history, the combined model's results are worse and FPR of 0.372 is quite intolerable, hence the combined model cannot perform well without regular retraining. To conclude, RQ3b shows that retraining model on a newer data is a mandatory condition for the combined model to achieve satisfactory performance and the whole user's commenting history is required when learning, whereas RoBERTa can perform relatively well even without retraining and only given partial commenting data in the learning phase.

To conclude, RoBERTa already presented better results when evaluated on standard metrics and this additional analysis proved that RoBERTa is, in general, more compatible with the practical issues that arise in the sockpuppetry problem. RoBERTa demonstrates comparable performance when handling comments from previously unencountered topics and it does not require regular retraining on more recent data. Therefore, RoBERTa's model exhibits the highest likelihood of achieving good practical results in sockpuppets detection in Wikipedia.

Chapter 7

Discussion

7.1 Implications

Through the various stages of evaluation, this study outlined RoBERTa transformer as the best model for catching sockpuppet accounts in Wikipedia. Even though RoBERTa presented very satisfactory results, the demonstrated behaviour still does not allow it to be a fully automated solution which regularly scans all the accounts contributing to Wikipedia, classifies them and automatically blocks accounts classified as sockpuppets. The main reason why RoBERTa cannot be fully automated is its high FPR of 9.8%. Although 9.8% FPR theoretically looks small, in reality the majority of Wikipedia accounts are legitimate, hence it would translate into a significant number of incorrectly blocked accounts. Since incorrectly blocking some account might result in serious problems, such as a decrease in the user's trust towards Wikipedia or their discontinuation of its use, the sockpuppet detection tool should be used very carefully. One way to mitigate the unwanted effects is to include a human in the process. There are 2 ways to make RoBERTa as a semi-automated solution involving a human in the decision making process. As the current procedure of detecting and blocking sockpuppet accounts is a 2-stage process, one can either automate the first or the second stage.

To reiterate, the first stage of the current procedure is other Wikipedia users detecting and reporting suspicious accounts through Wikipedia's sockpuppet reporting tool (see 2.1.2). This stage depends on other users' good will and such problems as the reporting tool being time-consuming or temporarily unavailable might lead to not reporting some account which was suspected as a sockpuppet. Clearly, this stage could be improved by using the RoBERTa model to flag the suspected accounts instead. RoBERTa could periodically scan the accounts which made some contribution and perform account classification behind-the-scenes. Those accounts which were classified as sockpuppets could be flagged and sent for further review to Wikipedia's administrators, which would make the final decision.

On the other hand, the second stage, where Wikipedia's administrators make the final decision (see 2.1.3), could be automated instead. Then reporting the suspected accounts would be done by all the Wikipedia's users as before, but the final solution if the account should be blocked would be automatically completed by the RoBERTa tool. In this

case, RoBERTa would make the decision on accounts which are already suspected not to be legitimate, thus 9.8% FPR should result in a significantly reduced number of incorrectly blocked legitimate accounts, if compared with making the classification on all the accounts.

To conclude, the found RoBERTa model cannot be used in a fully automated manner, but it can successfully be used as a semi-automated solution for sockpuppet detection automating either the first or the second stage of the current process. Deciding which stage to automate is a trade-off. Automating the first stage would result in flagging suspected accounts more accurately and free from faults caused by human action, however it would still require a significant effort from Wikipedia's administrators when making the final decision. Automating the second stage would reduce the workload of administrators, who could use their expertise and time towards other Wikipedia's problems, but the process of flagging suspected accounts would still depend on other Wikipedia users. Regardless of which stage was automated, incorporating the RoBERTa model to sockpuppet detection in Wikipedia should result in a highly improved and more objective process and greatly benefit all of the Wikipedia's community.

7.2 Comparison with existing literature

RQ1 found that the combined model is the most effective ML model to detect sockpuppet accounts in Wikipedia. Looking at the existing literature (see 2.2), the combined model group looked the most promising from the very beginning because it had the most recent works, it integrated the strengths of both approaches and also because of the primary comparison completed in the Yu et al. study [53]. In order to prove the effectiveness of integrating textual and non-textual features together, Yu's study also performed account classification when using these features in isolation. It found that non-textual features are slightly more effective than textual features (1-2% difference on metrics) and integrating both of them together can increase standard metrics by 6% on average if compared with results from using non-textual features only. The latter conclusion matches findings of this study - for the majority of the models, except for kNN, the combination of textual and non-textual models improved metrics by 1-8% if compared with using non-textual features alone (see 4.5 and 4.4) and thus resulted in the most effective ML based approach. Importantly, Yu's comparison doesn't diminish our study's importance as it's very basic: it only uses SVM (whereas our findings show that archetype's performance depend very much on the selected ML implementation, so one specific implementation is not a good representative of the archetype), lacks crucial metrics such as FPR, and ignores practical evaluation.

As for the RQ2, it found that transformers perform significantly better than previously analysed ML models. On the one hand, the fact that a model from a textual group turned out to be the most compatible one is somewhat unexpected. Looking at the literature about sockpuppetry problem in Wikipedia (see 2.2), the textual model group appears to have been largely overlooked - there is only Solorio et al. model in this group, which is 10 years old and suffers from high computational cost and poor evaluation, and after that no significant contribution was made in this area. On the other hand, transformers were deemed to be superior to ML models on a similar problem of detecting fake accounts

in OSN [27]. When these 2 types of models were evaluated in isolation, transformers reached higher scores on all the metrics, e.g. ML models reached only 89-91% recall, whereas transformers had recall of 94-97%. These findings provided us with a good premise to try transformers on Wikipedia's sockpuppetry problem and the outcomes not only proved to be successful but also revitalised the previously neglected field.

Regarding RQ3 findings, they cannot be compared with the existing literature. Even though topic and temporal generalisation are classical evaluation axes for models, none of the sockpuppet detection models for Wikipedia or other sites were found to be evaluated on these axes. The closest studies evaluated on these axes were those analysing bots [4, 52], however bots are conceptually different from sockpuppets as they are not controlled by humans, hence those studies cannot be used for comparison.

7.3 Critical evaluation

7.3.1 Design limitations

The first design limitation is that models were evaluated on the "Talk" namespace only. In reality, Wikipedia has a number of namespaces [42] and some studies used all of the namespaces to evaluate their model ([50], [51]). However, when working with all of the namespaces, fetching data from MediaWiki:API becomes not doable as the data quantity is huge and hence one has to work with Wikimedia database dumps [30]. Working with Wikimedia dumps requires much storage - the needed English Wikipedia dump ("All pages with complete edit history") is around 24 TB uncompressed [29]. Storage of that quantity was unavailable when completing this study and hence the models were evaluated on the "Talk" namespace only. However, in reality, only a small portion of sockpuppets contribute to the discussion area. After revisiting sections 3.2.2.1 and 3.2.2.2, one can observe that initially 20,361 of unique sockpuppets were retrieved, however only 2,483 (12%) of them contributed to talk pages. This means that even though this study found that the RoBERTa model is very effective when catching sockpuppets, its effectiveness was only proven for the small number of sockpuppets. On the positive side, the remaining namespaces is of the textual nature as well, e.g. "Main" namespace contains Wikipedia articles, "User" namespace is dedicated for interpersonal discussion, etc. This means that the RoBERTa model is applicable to other namespaces as well, however, its effectiveness in other Wikipedia areas still needs to be tested.

Another limitation comes from the methodology of collecting control group's contributions, more specifically controlling the control group's activity range when collecting the contributions (see 3.2.3.2). Even though this ensured that models do not rely too much on the number of contributions a user made, which was crucial especially for the textual models as this is a non-textual feature, it also resulted in losing some information. It means that for the control users, not all of their contributions, that have been made since the account registration, were analysed, but only those contributions which were made at the time window when the matching sockpuppet account was active. This might result in practical problems if the RoBERTa model was actually integrated into Wikipedia. Additional research would be required in order to find the optimal number of contributions which are needed to decide if an account is a sockpuppet or not.

7.3.2 Implementation limitations

Implementation limitations were already discussed when presenting the methodology, however, for the sake of completeness they are briefly mentioned again in this section. When recreating non-textual models, features, which are related with namespaces other than ”Talk”, could not be recreated (see 4.1.1.4). This resulted in a model with 5 features only. Such a small number of features might not reveal the full potential of a non-textual model and the situation might be improved by adding additional features. However the primary goal of this study was to compare models from different feature groups and just a minimal archetype already allows that. If some improvements were made towards the non-textual model, then to provide equal treatment for all the models, arguably other models should also be improved which would have resulted in a different study. Secondly, no hyperparameter tuning was completed for the transformers because of the time and computing resources constraints (see 5.2.2.3). Even though this did not prevent transformers from achieving satisfactory results, possibly models have the potential to attain even more optimal outcomes. Finally, as already mentioned in 6.2.1.1, 18.2% of talk pages were not assigned a topic when completing RQ3a. This meant that the inclusion of these articles in the testing dataset was deemed improbable. However, it did not cause any major implications as the testing set already had many candidate topics to sample from.

7.4 Possible future work

This study provides many possible directions for the future work.

One of the most important findings of this study was the innovative approach of using a RoBERTa transformer to catch sockpuppet accounts. Even though the RoBERTa model achieved outstanding results when compared with previously published ML models, it still cannot be implemented as a fully automated solution because of the high FPR. RoBERTa has an FPR of 9.8% which translates into a high number of incorrectly blocked accounts as the majority of users are legitimate in reality. Future research could investigate this problem and try to decrease the number of false positives. Some suggestions on how to achieve better results include tuning hyper-parameters for the transformer and finding the most optimal setting or including more data of a user, e.g. an account’s edits to the “Main” namespace (i.e. articles). Regarding other namespaces, as already mentioned in the 7.3.1 section, RoBERTa’s performance in them still needs to be tested, which is yet another direction for the future’s work.

What is more, since the RoBERTa transformer proved effective in Wikipedia’s discussion area, it would be interesting to test if transformers can also provide a satisfactory performance in other websites which suffer from the sockpuppetry problem such as Twitter, Facebook, Reddit, etc. Upon revisiting 2.2.4 section, one can observe that transformers were already tried on OSN and Twitter datasets to detect fake accounts by using transformers on its own, as well as incorporating transformers to the multi-fusion process [27]. Even though both methods proved effective in this study, they were tested on a dataset which included all types of fake accounts such as bots, compromised profiles, etc. As the proportion of sockpuppets within this dataset was not reported, one

cannot know if the proposed methods would work equally well if tested on sockpuppets only. Hence, some future study detecting only sockpuppets using transformers alone in other platforms might be a useful contribution to the research.

Also, the collected dataset is useful itself and it can be used in similar or related studies. The dataset contains a rich collection of 146,886 contributions to English Wikipedia's talk pages made by 4,966 users (50% sockpuppets, 50% legitimate accounts) between December 2001 and January 2023. Arguably, it is superior to the datasets used in other studies - Solorio's [23, 24] and Yu's [53] studies used datasets that included only around 700 users, whereas Yamak's studies [50, 51] used datasets about 10,000 users, however the data spanned a smaller time frame (only from 2004 to 2015). The dataset of this study contains a sufficient number of users and it covers a commenting history of more than 20 years. Consequently, it can be used in studies analysing if the commenting behaviour of users change over time or in studies analysing the sockpuppet problem from a different angle, for instance analysing if sockpuppets tend to contribute to some topics more than others as the information what topic a comment belongs to is present in the dataset.

7.5 Conclusion

The aim of this study was to outline the best candidate model to be implemented in Wikipedia for catching sockpuppet accounts. The primary analysis (i.e. RQ1), which considered only traditional ML models recreated from papers of past research, revealed that the model combining textual and non-textual features has the highest probability of achieving success as it scored the highest on standard metrics (77.6% recall, 77.3% precision, 77.3% f-score and FPR of 23%). However, the poor representation of the textual model group in RQ1, encouraged including more state-of-the-art textual models, namely transformers, in the comparison (RQ2). Detecting sockpuppet accounts solely based on transformers was not only an innovative approach, but it also turned out to be the most powerful solution - the RoBERTa transformer reached 84.4% recall, 9.8% FPR, 89.3% precision and 86.7% f-score on the very same dataset. RoBERTa's effectiveness was also proven from the practical point of view (RQ3) - it was found that RoBERTa can effectively generalise on unseen topics and unseen time periods without experiencing dramatic deterioration in its performance, opposite to the combined model whose performance dropped quite significantly when generalising on unseen time periods. RoBERTa achieving the highest results on standard evaluation, as well as on additional evaluation on practical dimensions allows to confidently present RoBERTa as the most promising model which has the highest potential of delivering positive results if integrated into Wikipedia for sockpuppet detection.

It is anticipated that having outlined the most promising sockpuppet detection model will accelerate the process of implementing semi-automated sockpuppet detection in Wikipedia. Given the limitations of the current manual solution and the number of aspects sockpuppet accounts can be detrimental to Wikipedia, this is a crucial and useful contribution to Wikipedia. Having more effective sockpuppets detection system would result in more accurate, more objective and more welcoming Wikipedia to all of us.

Bibliography

[1] Ahmed Alharbi, Hai Dong, Xun Yi, Zahir Tari, and Ibrahim Khalil. Social media identity deception detection: A survey. *ACM Comput. Surv.*, 54(3), apr 2021.

[2] Cambridge Dictionary. Meaning of frequency in English. https://dictionary.cambridge.org/dictionary/english/frequency, 2023.

[3] Xin Du, Siyuan Chen, Zhiyue Liu, and Jiahai Wang. Multiple userids identification with deep learning. *Expert Systems with Applications*, 207:117924, 2022.

[4] Juan Echeverrï£¡a, Emiliano De Cristofaro, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, and Shi Zhou. Lobo: Evaluation of generalization deficiencies in twitter bot classifiers. In *Proceedings of the 34th Annual Computer Security Applications Conference*, ACSAC '18, page 137–146, New York, NY, USA, 2018. Association for Computing Machinery.

[5] Hugging Face. Hugging Face transformers. https://huggingface.co/, 2023.

[6] Hugging Face. Trainer. https://huggingface.co/docs/transformers/main_classes/trainer, 2023.

[7] Wikimedia Foundation. Privacy policy. https://foundation.wikimedia.org/wiki/Privacy_policy, 2021.

[8] Wikimedia Foundation. MediaWiki API. https://www.mediawiki.org/wiki/API:Main_page, 2023.

[9] Wikimedia Foundation. MediaWiki API Etiquette. https://www.mediawiki.org/wiki/API:Etiquette, 2023.

[10] Wikimedia Foundation. MediaWiki API:Blocks. https://www.mediawiki.org/wiki/API:Blocks, 2023.

[11] Wikimedia Foundation. MediaWiki API:Categories. https://www.mediawiki.org/wiki/API:Categories, 2023.

[12] Wikimedia Foundation. MediaWiki API:Compare. https://www.mediawiki.org/wiki/API:Compare, 2023.

[13] Wikimedia Foundation. MediaWiki API:Revisions. https://www.mediawiki.org/wiki/API:Revisions, 2023.

[14] Wikimedia Foundation. MediaWiki API:Usercontribs. `https://www.mediawiki.org/wiki/API:Usercontribs`, 2023.

[15] Wikimedia Foundation. MediaWiki API:Users. `https://www.mediawiki.org/wiki/API:Users`, 2023.

[16] Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

[17] Jesus Leal. Using RoBERTA for text classification. `https://jesusleal.io/2020/10/20/RoBERTA-Text-Classification/`, 2020.

[18] Scikit Learn. GridSearchCV. `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`, 2023.

[19] Jiacheng Li, Wei Zhou, Jizhong Han, and Songlin Hu. Sockpuppet detection in social network via propagation tree. In João M. F. Rodrigues, Pedro J. S. Cardoso, Jânio Monteiro, Roberto Lam, Valeria V. Krzhizhanovskaya, Michael H. Lees, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2019*, pages 507–513, Cham, 2019. Springer International Publishing.

[20] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[22] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[23] 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*, pages 59–68, 2013.

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

[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] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] Pawan Kumar Verma, Prateek Agrawal, Vishu Madaan, and Charu Gupta. Ucred: fusion of machine learning and deep learning methods for user credibility on social media. *Social Network Analysis and Mining*, 12(1):54, 2022.

[28] Mudasir Ahmad wani, Nancy Agarwal, Suraiya Jabin, and Syed Zeeshan Hussain. Analyzing real and fake users in facebook network based on emotions. In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, pages 110–117, 2019.

[29] Wikimedia. Wikimedia Database Dump of 2023/01/01. https://dumps.wikimedia.org/enwiki/20230101/, 2023.

[30] Wikimedia. Wikimedia Downloads. https://dumps.wikimedia.org/, 2023.

[31] Wikipedia. Talk:ARM DynamIQ. https://en.wikipedia.org/wiki/Talk:ARM_DynamIQ, 2018.

[32] Wikipedia. Talk:Archdeacon of Raphoe. https://en.wikipedia.org/wiki/Talk:Archdeacon_of_Raphoe, 2020.

[33] Wikipedia. User:SineBot. https://en.wikipedia.org/wiki/User:SineBot, 2021.

[34] Wikipedia. Wikipedia:Signs of sockpuppetry. https://en.wikipedia.org/wiki/Wikipedia:Signs_of_sockpuppetry, 2022.

[35] Wikipedia. Wikipedia:Sockpuppet investigations. https://en.wikipedia.org/wiki/Wikipedia:Sockpuppet_investigations, 2022.

[36] Wikipedia. Talk:Donald Trump. https://en.wikipedia.org/wiki/Talk:Donald_Trump, 2023.

[37] Wikipedia. Wikipedia:Administrators. https://en.wikipedia.org/wiki/Wikipedia:Administrators#Involved_admins, 2023.

[38] Wikipedia. Wikipedia:Categorization. https://en.wikipedia.org/wiki/Wikipedia:Categorization, 2023.

[39] Wikipedia. Wikipedia:CheckUser. https://en.wikipedia.org/wiki/Wikipedia:CheckUser, 2023.

[40] Wikipedia. Wikipedia:Edit warring. https://en.wikipedia.org/wiki/Wikipedia:Edit_warring, 2023.

[41] Wikipedia. Wikipedia:Five pillars. https://en.wikipedia.org/wiki/Wikipedia:Five_pillars, 2023.

[42] Wikipedia. Wikipedia:Namespace. https://en.wikipedia.org/wiki/Wikipedia:Namespace, 2023.

[43] Wikipedia. Wikipedia:Size of Wikipedia. https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia, 2023.

[44] Wikipedia. Wikipedia:Sockpuppet investigations/SPI/Administrators instructions. https://en.wikipedia.org/wiki/Wikipedia:Sockpuppet_investigations/SPI/Administrators_instructions, 2023.

[45] Wikipedia. Wikipedia:Sockpuppet investigations/SPI/Clerks. https://en.wikipedia.org/wiki/Wikipedia:Sockpuppet_investigations/SPI/Clerks#List_of_clerks, 2023.

[46] Wikipedia. Wikipedia:Sockpuppetry. https://en.wikipedia.org/wiki/Wikipedia:Sockpuppetry, 2023.

[47] Wikipedia. Wikipedia:Username policy. https://en.wikipedia.org/wiki/Wikipedia:Username_policy#Using_multiple_accounts, 2023.

[48] Wikipedia. Wikipedia:WikiProject. https://en.wikipedia.org/wiki/Wikipedia:WikiProject, 2023.

[49] Wiki Workshop. Wiki Workshop 2023. https://wikiworkshop.org/2023/, 2023.

[50] 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*, pages 955–960, 2016.

[51] Zaher Yamak, Julien Saunier, and Laurent Vercouter. Sockscatch: Automatic detection and grouping of sockpuppets in social media. *Knowledge-Based Systems*, 149:124–142, 2018.

[52] Kai-Cheng Yang, Onur Varol, Pik-Mai Hui, and Filippo Menczer. Scalable and generalizable social bot detection through data selection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1096–1103, 2020.

[53] Hang Yu, Feng Hu, Li Liu, Ziyang Li, Xiangpeng Li, and Zhimin Lin. Sockpuppet detection in social network based on adaptive multi-source features. In Victor Chang, Muthu Ramachandran, and Víctor Méndez Muñoz, editors, *Modern Industrial IoT, Big Data and Supply Chain*, pages 187–194, Singapore, 2021. Springer Singapore.

[54] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2023.

[55] Wei Zhou, Jingli Wang, Junyu Lin, Jiacheng Li, Jizhong Han, and Songlin Hu. A time-series sockpuppet detection method for dynamic social relationships. In Guoliang Li, Jun Yang, Joao Gama, Juggapong Natwichai, and Yongxin Tong, editors, *Database Systems for Advanced Applications*, pages 36–51, Cham, 2019. Springer International Publishing.

Appendix A

First appendix

A.1 Formulas of metrics

$$TPR = Recall = \frac{TP}{TP+FN} \tag{A.1}$$

$$FPR = \frac{FP}{TN+FP} \tag{A.2}$$

$$Precision = \frac{TP}{TP+FP} \tag{A.3}$$

$$F-score = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*TP}{2*TP+FP+FN} \tag{A.4}$$

$$Metric_{macro-average} = 0.5*Metric_{sockpuppets} + 0.5*Metric_{control} \tag{A.5}$$

NB: For TPR and FPR positive class (1) is sockpuppets, negative class (0) is legitimate accounts, i.e. control group. For precision and f-score, since the macro-average of these metrics are reported, positive and negative classes are variable and depends on the group (sockpuppets or control) whose score is being calculated.

A.2 Extended results of RQ1

Textual features models

	SVM	RF	NB	KNN	ADA
TPR AVG	0.329	0.239	0.378	0.385	0.268
FPR AVG	0.235	0.068	0.294	0.208	0.132
Precision AVG	0.559	0.665	0.548	0.604	0.607
F-score AVG	0.524	0.528	0.528	0.569	0.523
Training AVG (in s)	71.520	61.368	0.895	0.065	37.531
Testing AVG (in s)	22.226	1.130	0.395	77.503	1.571
TPR SD	0.029	0.043	0.046	0.086	0.059
FPR SD	0.034	0.010	0.044	0.020	0.033
Precision SD	0.010	0.010	0.006	0.031	0.003
F-score SD	0.008	0.031	0.012	0.044	0.030
Training SD (in s)	35.491	17.540	0.033	0.003	1.429
Testing SD (in s)	12.420	0.317	0.020	0.996	0.160

Table A.1: Extended results of models based on textual features.

Non-textual features models

	SVM	RF	NB	KNN	ADA
TPR AVG	0.618	0.693	0.912	0.709	0.694
FPR AVG	0.324	0.300	0.781	0.388	0.278
Precision AVG	0.681	0.697	0.627	0.666	0.708
F-score AVG	0.626	0.696	0.506	0.657	0.708
Training AVG (in s)	0.678	0.456	0.003	0.005	0.150
Testing AVG (in s)	0.154	0.029	0.001	0.116	0.013
TPR SD	0.205	0.013	0.019	0.072	0.016
FPR SD	0.229	0.007	0.028	0.082	0.000
Precision SD	0.005	0.003	0.013	0.006	0.008
F-score SD	0.025	0.003	0.018	0.012	0.008
Training SD (in s)	0.048	0.120	0.000	0.000	0.068
Testing SD (in s)	0.003	0.007	0.000	0.017	0.005

Table A.2: Extended results of models based on non-textual features.

Combined features models					
	SVM	RF	NB	KNN	ADA
TPR AVG	0.709	0.776	0.910	0.654	0.749
FPR AVG	0.328	0.230	0.730	0.395	0.265
Precision AVG	0.691	0.773	0.653	0.629	0.742
F-score AVG	0.690	0.773	0.543	0.629	0.742
Training AVG (in s)	32.271	17.383	0.012	0.004	1.039
Testing AVG (in s)	0.261	0.282	0.005	0.996	0.026
TPR SD	0.009	0.008	0.011	0.009	0.010
FPR SD	0.032	0.009	0.010	0.021	0.016
Precision SD	0.012	0.003	0.007	0.009	0.008
F-score SD	0.013	0.003	0.006	0.009	0.008
Training SD (in s)	32.964	9.746	0.000	0.000	0.397
Testing SD (in s)	0.005	0.141	0.000	0.035	0.008

Table A.3: Extended results of models based on combined features.

A.3 Additional data for RQ2

Figure A.1: Most downloaded Hugging Face transformers trained on Wikipedia as of 14/02/2023.

Transformers

	RoBERTa	DistilBERT	BERT	XLNet
TPR AVG	0.844	0.766	0.744	0.632
FPR AVG	0.098	0.278	0.274	0.385
Precision AVG	0.893	0.727	0.725	0.646
F-score AVG	0.867	0.746	0.734	0.566
Training AVG (in s)	738.605	371.902	726.550	1321.469
Testing AVG (in s)	42.261	22.729	45.376	139.208
TPR SD	0.010	0.017	0.005	0.335
FPR SD	0.010	0.006	0.021	0.268
Precision SD	0.012	0.012	0.009	0.060
F-score SD	0.005	0.011	0.004	0.200
Training SD (in s)	1.092	1.920	1.733	2.118
Testing SD (in s)	0.288	0.122	0.308	0.638

Table A.4: Extended results of transformers (belong to textual group).