MASTER'S THESIS

Identifying Age and Gender and Analysing their Relation to Programming Behaviour of Scratch Users

Golsteijn, J.

Award date:
2022

IDENTIFYING AGE AND GENDER AND ANALYSING THEIR RELATION TO PROGRAMMING BEHAVIOUR OF SCRATCH USERS

by

Jelmer Golsteijn

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University, faculty of Science
Master Software Engineering
to be defended publicly on Thursday May 12th, 2022 at 13:00 PM.

Student number:		
Course code:	IM9906	
Course name:	Software Engineering Graduation Assignment	
Thesis committee:	mw. dr. ir. Fenia Aivaloglou (supervisor),	Open University, Leiden University
	dhr. dr. Stefano Bromuri (second supervisor)	Open University

Open Universiteit
de beste! www.ou.nl

ACKNOWLEDGEMENTS

I want to express my gratitude for the support I received from my mother and my sister during my graduation process. Without them, I could not have achieved the things I have. My acknowledgements also go to dr. ir. Fenia Aivaloglou, who has inspired me with her enthusiasm and expertise. She enabled me to improve my academic skills throughout my graduation project. Finally, I want to thank dr. Stefano Bromuri for sharing his knowledge on Artificial Intelligence. This allowed me to take the next step to more advanced topics.

ABSTRACT

Informal online programming communities form a first introduction to computer science for a lot of people around the world. One such community is Scratch, which is a popular block-based programming environment that enables its users to share projects they create with fellow Scratchers. Most educational research on Scratch focuses on analysing the programming behaviour of its users, which mostly consist of children. The age and gender of these children are important factors as understanding the capabilities and interests of children of different ages and genders makes it possible to further refine programming education practices and tools to their needs. This thesis presents a way of automatically eliciting age and gender information of Scratch users on a large scale using machine learning models. The proposed methods were used to enrich an existing dataset of Scratch users and projects with age and gender information. We then quantitatively analysed the programming behaviour of Scratch users in the enriched dataset.

In order to deploy our machine learning models, we first scraped user data using the Scratch API, such as user profile texts and social network data. From these profile texts, we identified and manually verified more than 6,000 users who disclose their age and gender in order to construct a training set for our machine learning models. We then validated the performance of several models on the training data. This resulted in the selection of a network-based Node2Vec model for gender identification, and a text-based Transformer model with selective classification for age identification. Cross-validation results revealed that both of these models achieve an F1-score of around 0.80 on the training set. We used these models to automatically elicit age and gender information for the rest of the dataset. This allowed us to quantitatively analyse block type and programming concept usage in relation to age and gender.

The use of our selected machine learning models resulted in gender information for 336.394 Scratch users, which is 82.64% of all users in the utilised dataset. Age information was elicited for 14.993 Scratch users, which is 3.68% of all users in the utilised dataset. Furthermore, our gender distribution was more similar to that of the entire Scratch population than our age distribution, which was skewed towards higher age groups. Our block type and programming concept analyses revealed some differences related to gender. Male Scratchers use 7 out of 11 block types and the programming concepts of conditionals, coordination, iteration, and variables in a larger percentage of their projects than female users. Looks, Control, and Events blocks are used more frequently in projects by female users. There were hardly any age-related differences regarding the usage of block types and programming concepts.

The proposed age and gender identification methods open up several directions for future work. These involve further exploration of the gender-related differences in programming behaviour that were observed in this study. This can be achieved by applying our machine learning methods to other datasets. More advanced analysis frameworks can also be used to deepen the understanding of gender- and age-related differences in programming behaviour.

Contents

1

INTRODUCTION

The presence of software has been ever increasing in our daily lives and society in general over the last fifty years. Its vital position in the world's infrastructure needs to be maintained by the people that have the proper computer science skills. However, as the amount of software increases, so does the need for computer scientists. It is therefore important to attract people to the field of computer science and make sure that they acquire the appropriate skills from the moment they first come in contact with computer science. This first introduction often occurs in formal computer science courses that are part of school curricula, or through informal online programming communities. One such community that focuses on newcomers to the field of computer science and programming is called Scratch[1]. The Scratch platform allows users to share projects that were created using a visual block-based programming language and connect with fellow users.

With over 80 million projects shared since 2007, Scratch has been a popular topic of investigation in the field of educational research [Aivaloglou and Hermans, 2016; Fields et al., 2017; Zeevaarders and Aivaloglou, 2021]. One of the research topics that is frequently studied is how Scratch users, which mostly consist of children, program and participate on the Scratch platform. Knowledge of how children engage in online programming communities and how they use introductory programming languages allows us to understand how to tailor these platforms to the needs of newcomers to the field of computer science and programming. The age and gender of these children are important factors and are also frequently studied [Fields et al., 2017; Funke and Geldreich, 2017; Graßl et al., 2021; Hermans and Aivaloglou, 2017; Wohl et al., 2015]. The reason for this is that understanding the capabilities and interests of children of different ages and genders makes it possible to further refine programming education practices and tools to their needs. Some studies on how children learn and use programming and computer science concepts make use of an experimental setup where specifically designed courses are taught to children and their performance and behaviour are analysed [Funke and Geldreich, 2017; Graßl et al., 2021; Hermans and Aivaloglou, 2017; Wohl et al., 2015]. This approach benefits from being able to measure many variables regarding a student's traits and behaviour, at the cost of only having a small sample size. On the other hand, there are approaches that use large datasets of scraped or publicly available data from online programming communities like Scratch and perform quantitative analyses to uncover patterns of how their users program

[1] https://scratch.mit.edu/

and participate [Aivaloglou and Hermans, 2016; Fields et al., 2017; Zeevaarders and Aivaloglou, 2021]. However, these datasets rarely contain more detailed information about the users such as age and gender, since they are not publicly exposed or available.

This thesis presents a study that involves 1) enriching an existing dataset of Scratch project repositories [Zeevaarders and Aivaloglou, 2021] with age and gender information of their authors, and 2) performing analyses on this dataset in order to explore how programming behaviour of Scratch users is related to their age and gender. By doing this, we aim to answer to following research questions:

RQ1 What age and gender information can we identify of Scratch users from the publicly available information on the Scratch platform using machine learning models?

RQ2 Which age- and gender-related differences can be detected in the block types Scratch users use in their projects?

RQ3 How frequently do Scratch users of different gender and age groups use the conditionals, coordination, iteration, and variables programming concepts in their projects?

In this thesis, we start by introducing the Scratch platform and its corresponding terminology in Section 2. After that, Section 3 describes other works in the fields of research that are related to our study. In Section 4, we discuss the methods that have been used to answer our research questions. Section 5 contains all the information pertaining the selection of machine learning models for our first research question. The elicited age and gender information and the results of our analyses are presented in Section 6. A discussion of the results and limitations of the study can be found in Section 7. Finally, Section 8 concludes the thesis.

2

THE SCRATCH PLATFORM

Scratch is a visual block-based programming language that is designed for children with no prior programming experience. Scratch is developed by the Lifelong Kindergarten group at the MIT Media Lab[1] and was released in 2007. A Scratch program consists of blocks of different shapes, sizes, and colors. These attributes, as well as the textual information inside the blocks, determine the functionality of the blocks and how they can be connected to form a script. Scratch users can assemble these scripts on a canvas called the code area by means of a drag and drop feature. Each code area belongs to a sprite and bounds the scope of the scripts within them. A sprite is an image that can be seen when the Scratch program is run and the scripts within its corresponding code area determine how the sprite will behave (e.g. moving, talking, changing size). Scratch follows the event-driven programming paradigm. This means that every script needs to be triggered by a certain event, which is defined by the top-most block of the script, also called a Hat block. The most common Hat block is the When Green Flag Clicked block, which is triggered by clicking a green flag that appears over the user interface. Figure 2.1 shows the interface for creating a Scratch program.

Scratch users can create projects either through the desktop client or on the official Scratch website. Additionally, the Scratch website provides a social platform on which its users can share their projects, view creations of others, and socially engage with fellow users in various ways. Each Scratch user has their own profile page, an example of which is shown in Figure 2.2. The top section of the profile page provides some personal details of the Scratch users, including their username, date joined, country of origin, an 'About me' and 'What I'm working on' description, a featured project, and an activity feed showing the user's actions on the Scratch platform up to one year ago. Furthermore, the profile page shows the projects the user has shared and favorited, as well as which studios the user follows. Studios are places in which projects from different users can be grouped. A studio usually has a theme that indicates what types of projects you may find in the studio. The bottom half of the profile page, which can be seen in Figure 2.2b, contains the studios the user curates, the usernames and profile pictures of other Scratch users the user follows and is followed by, and a comment section in which the user themselves and other users can leave comments. Scratch users can also place comments on studios and projects.

[1] https://www.media.mit.edu/

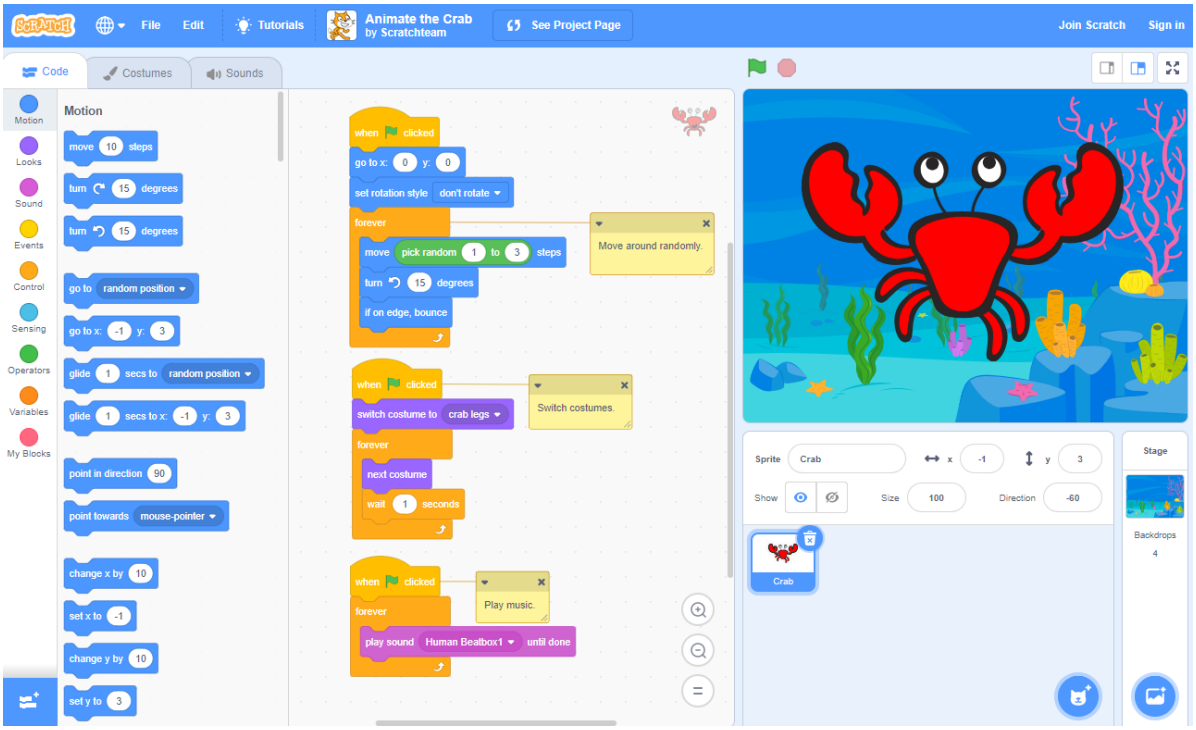
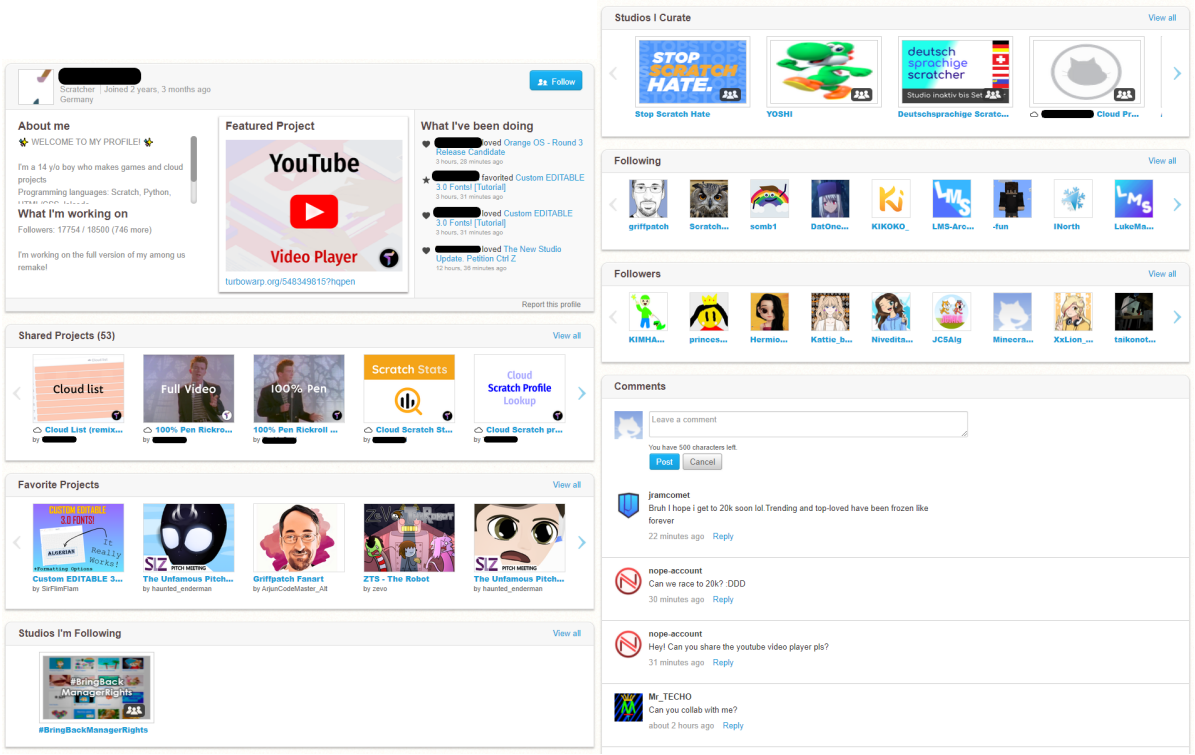


Figure 2.1: The editing interface of an example Scratch program made by the Scratch Team



(a) Top half

(b) Bottom half

Figure 2.2: A user profile page on the Scratch website

3

Related work

Our project is related to three research areas. First, the goal of our project is to study the role of age and gender in introductory programming. Second, the research method that is used involves a quantitative analysis of Scratch programs. Third, the method of eliciting age and gender information involves age and gender identification using machine learning. Related work on these topics is discussed in Sections 3.1, 3.2, and 3.3 respectively.

3.1. Age and gender effects in introductory programming

Understanding the role of age and gender when learning to program and coming into contact with computer science is an important research topic within the area of computer science educational research. The reason is that this understanding can help develop the proper tools, practices, and courses that facilitate an optimal introduction to the field of computer science that is tailored towards the needs of students of different age and gender. This research is especially relevant for gender, as the field of computer science is disproportionately occupied by males. This gender gap has been observed, among others, by Wang and Degol [2016], who describe that women were only awarded 18% of the computer science bachelor degrees in the USA in 2012. Examples of studies that have effectively closed the gender gap in specific scenarios exist. For instance, Rubio et al. [2015] analysed gender-related differences in a university level MATLAB introductory programming course. They observed the existence of a gender gap based on pre- and post-course questionnaires. In order to try and close this gender gap, the authors designed several learning modules based on physical computing and implemented them in the course. The evaluation of the newly designed course using questionnaires showed that the differences in perception and learning outcomes had disappeared. Although there is a need for more confirmation in other institutions, this study shows that the gender gap in introductory programming can be closed using alternative learning techniques.

The role of gender has also been studied in relation to the Scratch programming language. Funke and Geldreich [2017] investigated the relationship between gender and the characteristics of Scratch programs that were created by students during a three-day course. The course was designed for 9 and 10 year old students and was attended to by 58 children in total spread over four iterations. A quantitative and qualitative analysis of the students'

Scratch projects showed that boys tend to use more blocks on average and a greater variety of blocks. Boys also made more frequent use of Motion type blocks, which could indicate a preference towards game type projects. Girls on the other hand tend to use more sprites and use twice as many Look type blocks, which could indicate a preference towards story type projects. They observed a balanced gender distribution among animation type projects. Finally, the projects were categorized in one of five levels of understanding. Their analysis showed that while the programs on the lowest level were disproportionately made by girls and those of the highest level only made by boys, the gender was equally distributed across all levels in between.

The study design of Funke and Geldreich [2017] was later replicated by Graßl et al. [2021] in order to obtain a combined total of 319 Scratch programs. They studied gender differences and similarities by performing automated topic and program analyses of these Scratch programs. Their goal was to acquire a better understanding of gender-related differences in the use of programming languages like Scratch, which can serve towards creating better learning environments that can address the gender imbalance in computer science. A Latent Dirichlet Allocation model [Blei et al., 2003] was used to automatically extract topics from the textual data of Scratch programs. They studied topical differences in the programs of boys and girls, and found stereotypical differences in interests. For instance, boys took a liking in bats, ghouls, and soccer, while unicorns, dancing, and music appealed to girls. There were also topics like circus, parties, and animals that appealed to both genders. Similarly to the study by Funke and Geldreich [2017], the program analysis showed that girls prefer story-like projects, while boys prefer game-like projects. This difference was also reflected in the usage of code blocks. Girls primarily use simple control blocks like $forever$ and $wait$, whereas boys use conditional statement blocks more frequently. The authors suggest that this is due to game-like projects requiring more advanced control blocks to implement the game logic, whilst story-like projects usually only have a sequential flow. As for code smells, girls' projects contain twice the number of duplicate and empty sprites compared to boys' projects, whereas boys' projects contain more "Missing Initialization" and "Stuttering Movement" code smells. The authors state that this difference may be due to story-like projects containing more decorative sprites than game-like projects, whilst game-like projects contain more user interaction. Based on these findings, the authors suggest that more attention is required for the needs and interests of girls, so that they are equally challenged and motivated in teaching materials.

The studies by Fields et al. [2013] and Fields et al. [2017], which are also discussed in Section 3.2, explore the role of gender in relation to the Scratch programming language. The authors used a dataset containing the activities of 5,004 Scratch users to quantitatively analyse programming and participation patterns. The observed programming patterns range from only using a low amount of loops and almost no other advanced concepts, to the usage of large amounts of advanced concepts, especially Booleans. Fields et al. [2017] studied if belonging to a certain programming profile could be related to gender, length of membership, or participation. They found that gender has a significant impact on being in one of these programming profiles, as girls seem to appear in disproportionately high numbers in the lowest profile, and in low numbers in the most advanced profile. The opposite pattern is observed for boys. On the other hand, there seemed to be no gender differences in participation profiles [Fields et al., 2013]. This could indicate that social engagement might not always lead to programming engagement on these kinds of platforms. The study

by Gan et al. [2018], which is also discussed in Section 3.2, analysed the project sharing behaviour of male and female users in a dataset of over 1.1 million users. They found that while girls share less than boys initially, the reverse effect is observed among experienced users.

With respect to age, research tends to focus on what tools provide the optimal means of conveying computer science and programming language concepts to children. Wohl et al. [2015] performed a comparative study of teaching computer science to 5, 6, and 7 year old children using three different methods: Unplugged, Cubelets, and Scratch. The Unplugged method aims to teach children about computer science concepts in an unplugged environment, whereas the Cubelets method aims to teach these concept with the use of tangible objects. They conducted their study on three UK primary schools, hosting three sessions on each school with each a different ordering of methods used. The level of understanding of the three main concepts of algorithms, logical prediction, and debugging was measured quantitatively using marked paper models that the pupils had to create based on the work they had done, and qualitatively, using interviews that were taken after each session. The results showed that while the Unplugged method generated the highest levels of understanding of the three concepts, the Scratch method sparked more creativity and initiative among pupils. They also observed a downward trend in the enjoyment ratings that the pupils give as the sessions progressed.

Another example of a work that studies conveying programming concepts using Scratch is the one by Hermans and Aivaloglou [2017]. The authors designed and ran an introductory Scratch programming MOOC (Massive Open Online Course) in which they taught basic programming concepts as well as software engineering concepts to over 2,220 children. Of the students that reported their age and gender, 73% of the participants was between 7 and 11 years old and 31,66% was female. The resulting data was analyzed to see if there is a difference in scores between programming and software engineering concepts, if there are age-related differences in the participants' performance on these concepts, and if the participant profiles and first week activities can be used to predict course completion. They found that there was no significant difference between the mean grades of programming and software engineering questions. A comparison between the grades of 11-12 and 13-14 year old students showed that there was a significant difference on the Operator and Procedures concepts. A logistic model for predicting course completion shows that factors like being late and the mean grade in quizzes can influence the chance of successful completion. Furthermore, age and gender does not have a significant effect on completing the course.

Understanding how children of different ages approach and solve computer science problems is essential for the development of appropriate computer science curricula. Some studies have been dedicated to exploring means of measuring these kinds of computational thinking skills according to concrete criteria. For example, Seiter and Foreman [2013] proposed a framework for understanding and assessing computational thinking skills of children in the primary grades, which they called the Progression of Early Computational Thinking (PECT) Model. The model detects high-level programming design patterns and maps them to computational thinking concepts. These design patterns are distilled from "Evidence Variables", which are concretely measurable computational aspects of Scratch programs (e.g. Block usage). Each evidence variable, design pattern, and computational thinking concept is organised by a three point scale that denotes the proficiency of compu-

tational thinking displayed by the Scratch user. The authors performed a pilot-test study of the PECT Model to demonstrate its ability in codifying computational thinking and detecting computational thought differences among students of different ages. The study involved assessing the proficiency levels of 150 Scratch projects using the PECT Model, and analysing the results across Grades 1 till 6 (corresponding to the ages of 6 till 12). The results of the study show that design patterns which utilise advanced programming concepts were most often used by children of Grades 5 and 6, whereas the distribution of the use of simpler design pattern was balanced across Grades 1 till 6.

3.2. Quantitative Scratch analysis

Studies that investigate how children program are widely available in the literature of the field of educational research. The Scratch platform has served as an excellent means for some of these studies as Scratch is popular amongst children and has publicly available programs. This enabled researchers to perform quantitative analyses that aim to reveal patterns of how children code and participate on these platforms. One of the first of such analyses was performed by Fields et al. [2013], who quantitatively analysed participation patterns on the Scratch platform. They collaborated with the Scratch Team at MIT to gather data about 5,004 Scratch users during the first three months of 2012. This data includes logged activities and back-end data such as age and gender. Latent Class Analysis (LCA) [Muthén and Muthén, 2000] was used to identify groups of users that participate similarly. Six variables were used to measure participation: 1) Remixing, 2) Downloading projects, 3) Commenting, 4) Favorites, 5) Love-its, and 6) Friend requests. They found that there was no user that participated in any of the measured activities without creating at least one project themselves. Project creation therefore serves as a gatekeeper to all other activities. Consequentially, only the users that had created at least one project (44.5%) could be used for the analysis. Downloading projects also seemed to be a gatekeeper for the other social activities, which may indicate that users not only play projects and interact on the Scratch platform, but also investigate how the projects were made. Furthermore, the number of classes identified by the LCA analysis got progressively less over time, with the class indicating the lowest level of activity increasing in relative size. This could be an indication of gradual drop-off in activity over time. Finally, they found that there were only minimal gender differences in participation within the identified class profiles.

The authors continued to study the dataset of 5,004 Scratch users and published a new work [Fields et al., 2017] in which they quantitatively analysed programming patterns. They examined programming concept usage in relation to the level of participation, gender, and length of membership of Scratch users. Their study first investigated if there are any broad patterns of programming language usage that qualitatively distinguish the users' programs. Similarly to [Fields et al., 2013], the authors used LCA to identify four programming patterns that showed consistent size across three months. The authors then looked if differences in quality could be related to length of membership, gender, or participation. They observed that length of membership does not play a large role in the chance of belonging to one of these profiles. For a discussion on the role of gender, see Section 3.1. Lastly, they looked at how the identified programming patterns shifted month by month and observed that there was high movement between the three most advanced patterns, and low movement between the most simple pattern and the others.

Simlarly to Fields et al. [2013, 2017], Gan et al. [2018] also collaborated with the Scratch

Team at MIT to construct a dataset of Scratch users and projects. They were permitted to query a copy of the SQL database that runs the Scratch online community. This way, the authors collected data for every user who registered between July 1^{st} 2014 and January 31^{st} 2015. They then collected projects (excluding remixes) that were created on the Scratch web platform through January 31^{st} 2016. The final dataset contained over 1.1 million users and 5.6 million shared and unshared projects. The authors analysed project sharing behaviour of male and female users using bivariate descriptive statistics. This showed that inexperienced girls shared less than inexperienced boys, while this effect is flipped for experienced boys and girls. Using Bayesian regression analyses, the authors have shown that this effect can, for a large part, be explained by the differences in the way boys and girls participate. Gan et al. [2018] also studied the relation between gaining positive feedback and sharing behaviour. They found that inexperienced users are more likely to share projects when gaining prior positive feedback. As with gender, this effect reversed for experienced users.

While the analysis of the Scratch user datasets by Fields et al. [2013, 2017] and Gan et al. [2018] have provided valuable insights, they also remain the only one of their kind. Since then, there have been no similar collaborations where MIT provides a dataset for researchers to analyse. Moreover, Scratch has changed a lot since the most recent dataset was constructed in 2016. In the meantime, the number of monthly active project creators has grown from 162,471 in the first month of 2016, to 808,547 in the first month of 2022[1]. However, researchers in the field of educational research have found other means of analysing how children code using Scratch. One of such means is to scrape publicly shared Scratch programs from the Scratch website and construct a dataset that can be analysed quantitatively. Aivaloglou and Hermans [2016] were the first to perform a large-scale exploratory study of Scratch programs by scraping and analysing over 250,000 public Scratch projects. They studied the size and complexity of these programs, the coding abstractions and programming concepts used, and the occurrence of code smells. The results of the analysis showed that the majority of Scratch projects are small and simple, and contain scripts without decision points. Only 8% of the projects use procedures, while 77% make use of loops, although rarely including a conditional statement. Code smells do seem to occur in Scratch programs, with 28% containing dead code and 30% containing large scripts. They also discuss some language design implications like the need for having a separate workspace for storing unconnected blocks to reduce dead code.

An even larger scale exploratory study was later performed by Zeevaarders and Aivaloglou [2021], who studied the progression of programming concepts practiced by Scratch users by analysing their complete public project repositories. A dataset was constructed with over 112,000 authors and 1 million projects. This dataset was analyzed to determine how Scratch users progress in the use of elementary programming concepts and application of Computational Thinking (CT) skills. They also looked at which programming concepts were practiced by users before they left the Scratch platform. Their results show that there is an upward trend in the use all elementary programming concepts. The application of CT skills also seems to have a positive trend for the most advanced levels in the Dr. Scratch rubric [Moreno-León and Robles, 2015] that was used. However, about half of the users that left Scratch after creating at least nine projects did so without utilizing procedures, and a third left without using conditional loops.

[1] https://scratch.mit.edu/statistics/

A downside of using datasets of public Scratch programs is that they contain minimal information about the authors of the programs. The reason for this is that sensitive user information like age and gender is not publicly available on the Scratch platform. Given that this information is useful when analysing how children code using Scratch, another approach that involves designing and running Scratch programming courses is also used in this area of research. Examples of studies that make use of this approach include the ones by Wohl et al. [2015], Funke and Geldreich [2017], Hermans and Aivaloglou [2017], and Graßl et al. [2021]. A discussion of these works can be found in Section 3.1, as they all explore the role of age or gender in introductory programming.

3.3. Age and Gender Identification using Machine Learning

3.3.1. Machine Learning Methods

An important part of our research is the identification of age and gender of Scratch users in the utilised dataset. This can be viewed as an Author Profiling (AP) task, which is a field of research within the area of Natural Language Processing (NLP) that concerns the automatic identification of certain personal traits based on the content an author produces. In order to stimulate the progression of AP research, PAN[2] organizes a shared task each year. Researchers can work on these tasks, submit their work to PAN, and write a publication on the results. The results of all participants are published in an overview paper by PAN. These papers provide insight in the state of the art methods in the field of AP, which are almost exclusively machine learning approaches. In 2018, the shared task involved gender identification of Twitter users using textual information and images from Tweets [Rangel et al., 2018]. Twitter, much like Scratch, is a social platform with large amounts of user-generated data. Participants of the shared task were granted access to a multilingual corpus containing 12,600 Twitter users with 100 tweets and 10 images for each. The results indicated that traditional supervised machine learning approaches like Support Vector Machines and Logistic Regression remain competitive for textual information [Daneshvar and Inkpen, 2018], while new approaches that make use of Neural Networks are able to utilize imagery data much better [Takahashi et al., 2018].

The current state-of-the-art Neural Network based methods in NLP tasks originate from the Bidirectional Encoder Representations from Transformers (BERT) model. This language representation model was introduced by Devlin et al. [2019] and can be pre-trained on large unlabelled text corpora to learn deep bidirectional language representations. The training objective is based on a masked language model (MLM). This involves randomly masking input words, and predicting these words based on their context. The pre-trained model can then be fine-tuned for specific downstream tasks by adding an additional output layer.

Although these kinds of Neural Network based models perform well at various classification tasks, a downside of using them is their complexity. The increasing complexity of machine learning models has a negative effect on the interpretability of these models. This makes it more difficult to understand how a machine learning model makes its predictions and to trust these predictions. In order to solve this trust problem, recent studies have fo-

[2] https://pan.webis.de/

cused on exploring ways to make complex machine learning models interpretable. One of such methods has been proposed by Ribeiro et al. [2016] and is called Local Interpretable Model-agnostic Explanations (LIME). This explanation technique involves learning local interpretable models for any machine learning model. The term local refers to the prediction of individual data points instead of the prediction of all data points (global). Local models are learned by randomly sampling nearby data points and using these data points to obtain models that are locally faithful to the original model. LIME asserts the interpretability of these local models by requiring the input data to be interpretable. The technique is also model-agnostic, as it does not make assumptions about the original machine learning model and treats it as a black box. Another method of interpreting predictions of complex models is called Shapley Additive Explanations (SHAP), and was proposed by Lundberg and Lee [2017]. SHAP involves assigning an importance value to each feature used in the prediction of a machine learning model, which can help understand the contribution of each feature in a particular prediction. The importance values are obtained by measuring the difference between the predictions with or without a certain feature. However, in non-linear models or when the features are not independent, the order in which the features are given as input matters. In those cases, all possible orderings have to be considered and an average importance value will have to be calculated.

3.3.2. Data representation

Supervised machine learning methods base their predictions on training data. As the raw data is often too large and contains too much noise, only a subset of features is used that characterises the data better given the task at hand. The process of extracting, constructing, and selecting features from the data is quite labor intensive as it usually requires several iterations in order to obtain a good performance for the given task. Recent work in the field of machine learning has focused on automatically learning features from data instead of using pre-defined features. In the context of social media and online communities, this method is referred to as user embedding learning. Pan and Ding [2019] review this recent work in a literature study on social media-based user embedding and define it as "the function that maps raw user features in a high dimensional space to dense vectors in a low dimensional space." Another benefit apart from not having to use pre-defined features is that user embeddings capture general user characteristics and can thus be used for a variety of downstream user analysis tasks such as age and gender identification. Pan and Ding [2019] observe four types of user embeddings based on the type of user data they represent: text (e.g posts, comments), image (e.g. profile pictures, Instagram posts), network (e.g. friends, followers), and other (e.g. likes, favorites).

Of these types of user embeddings, text-based user embeddings have been widely studied. Pan and Ding [2019] describe the most commonly used methods for acquiring text-based user embeddings and make a distinction between unsupervised dimension reduction methods and self-supervised neural network-based prediction methods. Unsupervised dimension reduction methods include Latent Dirichlet Allocation (LDA), which tries to explain observations by unobserved latent groups. A benefit from this approach is that its results can be interpreted by humans. Other unsupervised dimension reduction methods include Singular Value Decompostion (SVD) and Principle Component Analysis (PCA), which decompose sparse user-word or word-word matrices into more compact matrices that contain the hidden relations and structures of the data. A large benefit from these un-

supervised methods is that they do not require labelled data and can thus be used to obtain user embeddings from large collections of raw data.

However, most text-based embedding methods make use of neural network-based methods, which are supervised machine learning methods and do require training examples. Therefore, these neural network-based methods often require an auxiliary training task for which a large amount of labelled data can easily be acquired. These methods are usually qualified as self-supervised machine learning, as the data for the auxiliary task can be automatically retrieved without the need for human annotation. The type of auxiliary task depends on the used method and the downstream analysis task. For example, Wu et al. [2020] have created an end-to-end neural network-based user embedding system, Author2Vec, in which authorship classification is used as an auxiliary training objective. They show that their system performs well in downstream user analysis tasks such as gender identification.

Another type of user embedding involves reducing social network topology to low dimensional embeddings that capture the social structures and relations of a user. Pan and Ding [2019] describe popular methods such as DeepWalk and Node2Vec that are based on performing truncated random walks in a social network graph in order to learn latent representations of nodes. As for image-based embeddings, most recent work has made use of VGGNet models. These pre-trained Convolutional Neural Network based (CNN) models were created by Simonyan and Zisserman [2015] and are available online. These models achieve state-of-the-art performance in image classification and the image embeddings can also be used for other classification tasks. Finally, user embeddings that are based on likes or similar types of data mostly make use of text-based user embedding methods. For example, Kosinski et al. [2013] show that SVD-reduced user/like matrices can be used in logistic regression models in order to identify user traits of Facebook users like age and gender very accurately.

When multiple types of user embeddings are used, fusion methods can be used to create a uniform embedding that represents a user. Apart from simple concatenation, Pan and Ding [2019] describe general fusion methods such as Canonical Correlation Analysis (CCA) and Deep Canonical Correlation Analysis (DCCA) which try to find a linear and non-linear transformation respectively so that two given feature vectors are maximally correlated. There are also several studies that make use of customized fusion methods that are specifically designed to combine certain types of user data. For example, Zhang et al. [2017] propose an algorithm called User Profile Preserving Social Network Embedding (UPPSNE) that makes use of both user profile data and network data to learn a vector representation of a network.

4

METHODS

In order to answer our research questions, we first enriched the Scratch dataset [Zeevaarders and Aivaloglou, 2021] with additional user data scraped using the official Scratch API. This data was required for the automatic identification of age and gender. Then we trained several machine learning models on manually labelled data and compared the results of these models in order to determine which models were best suitable for identifying the age and gender of Scratch users. These models were then used to elicit age and gender information of Scratch users in the utilised dataset. The resulting age and gender data was used for an analysis on block and programming concept usage. All files that were created, used, or generated for the purpose of this thesis are publicly available online[1], supporting reproducability to the fullest extent possible. The following sections describe the methods used in this process in further detail.

4.1. DATASET

In order to be able to study the learning curves of Scratch users, Zeevaarders and Aivaloglou [2021] scraped the public project repositories of 407,079 authors, containing a total of 7,109,821 projects. The number of blocks used in these project sum up to a total of more than 520 million. The dataset also contains Scratch user and project metadata, such as the country of origin of the user, and when a project was last modified. Their tools started scraping on the 1^{st} of September 2019, and finished on the 27^{th} of October 2019. All the scraped data was then parsed and stored in a relational database.

We have used the dataset by Zeevaarders and Aivaloglou [2021] for this study and extended the scraping effort to obtain even more data about the Scratch users and projects in the utilised dataset using the official Scratch API. Furthermore, we manually labelled age and gender information of more than 6,000 Scratch users in the utilised dataset. The following sections provide more detail on the kinds of data that the original dataset was enriched with.

4.1.1. SCRAPING & DATASET ENRICHMENT

Four types of user data were scraped using the official Scratch API. The first scraper started running on the 27^{th} of September 2021, and the last scraper finished running on the 21^{th}

[1] https://1drv.ms/u/s!AofN0KrcSoj_oJAQ5Ypr2Me5-XEt8g?e=8aMC8u

of October 2021. At least one API call was issued for all 407,079 Scratch users in the dataset per data type. Table 4.1 contains an overview of the results of our scraping effort. All four types of user data are discussed in more detail in the following paragraphs.

General user profile data

There was already quite a lot of general user profile data in the dataset, such as the user ID, username, the joined date, and the country of origin. However, we also wanted the information that users provide in the "About me" and "What I'm working on" sections of their user profile. These text areas are respectively referred to as the bio and status in the API and contain a maximum of 200 characters each. These texts were especially useful for the identification of age and gender of Scratch users, as the information relates directly to the users themselves. We encountered 65 failed API requests, which would occur if the account had been deleted.

User profile comments

Comments that users post on the Scratch website also provided a great source of textual data. Sadly, the Scratch API does not provide means to retrieve all comments a specific user has posted. We were, however, able obtain all comments that have been posted on a user profile. Therefore, we have scraped the comments that were posted on the user profiles of each Scratch user in the utilised dataset.

User profile pictures

The user profile pictures were also used for identifying a user's age and gender. These images have been stored in a file system, while the image file paths were stored in the database. This was to prevent the database size from becoming too large. After that, each profile picture was compared to the default profile picture. All images that were identical to the default profile picture were removed. A total of 258 profile pictures were unusable, as their image files were corrupted.

Project favorites & user follows

We have also scraped which projects a user has favorited and which other Scratch users a user follows. User follows were used in this study to map the social network on the Scratch platform. Project favorites were scraped for potential future work purposes.

Data type	Count
Status text	154,736
Bio text	177,013
Profile comment	35,559,553
Non-default profile picture	284,521
Following relation	15,835,193
Project favorite	30,008,212

Table 4.1: Overview of the amount of scraped data for each data type

4.1.2. Collecting labelled data

The identification of age and gender cannot be done without having access to labelled training data that our supervised machine learning models can use to learn how to map

user data to age and gender labels. This training data consists of pairs of Scratch users and their age or gender. The utilised dataset did not contain such data and the self-reported age and gender of Scratch users is not made publicly available by Scratch. Therefore, we depended on Scratch users that voluntarily disclose their age or gender in their profile bio or status for the assembly of our labelled training data. We have made use of pattern matching rules to find bio or status descriptions that contain commonly used phrases that indicate a user's age (e.g "I am 10 years old") and gender (e.g "I am a girl"). These profile descriptions were then manually checked for any false positives.

Three types of regular expressions have been used: one that extracts both age and gender, one that extracts just age, and one that extracts just gender (Table 4.2). Manual additions or adjustments were made if age or gender information was present in the text and was not captured by the regular expression. Special labels were attached to texts that contained interesting information other than age and gender. Details about the resulting dataset can be found in Section 5.1.

Expression type	**Regular expression**
AGE_GENDER	`\b(?:i am|i\'m|im) (?:a|an) ([0-9]+) (?:year old|y/o) (boy|male|man|guy|dude|lad|girl|female|woman|lady)\b`
GENDER	`\b(?:i am|i\'m|im) (?:a|an) (boy|male|man|guy|dude|lad|girl|female|woman|lady)\b`
AGE	`\b(?:i am|i\'m|im) ([0-9]+) (?:years old|y/o)\b`

Table 4.2: Utilised set of regular expressions for extracting age and gender information from Scratch profile texts

4.2. AGE AND GENDER IDENTIFICATION

In this phase of the research, we used the scraped data to identify the age and gender of Scratch users in the utilised dataset. Supervised machine learning models have been trained on various types of manually labelled user data. The best models were used to automatically obtain age and gender information of other Scratch users in the utilised dataset. The following sections dive further into the machine learning models and validation methods used.

4.2.1. TRAINING MACHINE LEARNING MODELS

For the identification of age and gender of Scratch users, we have used textual, visual, and network user data. Textual data was available in the form of profile status and bio texts, and comment texts, while visual data was available in the form of profile pictures. The following relations between users depict the social network they are part of and can thus be

treated as network data. For each of these data types, we have trained at least one supervised machine learning model. For gender, this can be seen as a binary classification task ('male' or 'female'). For age, this can be seen as a multi-class classification task, where age is represented as one of four age groups ('4-9', '10-11', '12-13', or '14+'). The following sections describe for each data type how the data was processed and which machine learning models were trained.

TEXTUAL USER DATA

For each user, we aggregated all status, bio, and comment texts belonging to that user into a single document. Three different machine learning models have been trained on the aggregated textual user data. The models range from simple and traditional (bag-of-words) to advanced, recent, and state-of-the-art (Transformer). These models are described in more detail in the following paragraphs.

Bag-of-words (tf-idf) The first model we have trained is a bag-of-words model with tf-idf transformation. A bag-of-words model represents natural language as a point in vector space. The dimensions of this vector space are determined by the size of the vocabulary of all users. Each row represents the textual data of one user. Each column represents a word from the vocabulary. The values in each cell determine the number of times that word occurs in the user text. Therefore, a bag-of-words model loses most syntactic information of a text [McTear et al., 2016].

Tf-idf transformation gives weights to each word in a user text based on how important the word is. This weight is calculated by multiplying the term frequency (tf) by the inverse document frequency (idf). Term frequency is the relative frequency of a word within a user text. Inverse document frequency is the inverse of the portion of user texts that contain a certain word. This means that words that occur infrequently in all user texts receive a low weighting, while words that occur frequently in a few user texts receive a high weighting [Salton and Buckley, 1988].

Before the user texts were fed into the bag-of-words model, they were first subjected to pre-processing. This reduced number of words in the texts and decreased the vocabulary size, whilst maintaining the semantics of the texts. The following filters were applied to the texts:

1. Lowercase all characters
2. Remove punctuation
3. Remove URLs
4. Remove username references (@username)
5. Remove stopwords
6. Lemmatize all words
7. Tokenize the texts

After bag-of-words vectorisation and tf-idf transformation, the user text representations were fed into a Support Vector Machine (SVM). An SVM is a supervised machine learning method and has the objective of seeking a hyperplane that separates the two classes of data-points with a maximum margin to the nearest data-point of both classes [Cortes and Vapnik, 1995]. SVMs have shown good performance in classification tasks up to this day [Rangel et al., 2018] and have also been used for gender classification using textual data of social media users [Benton et al., 2016; Schwartz et al., 2013].

Doc2Vec The second model we have trained is a Doc2Vec [Le and Mikolov, 2014] model. Doc2Vec is an extension of Word2Vec, which learns individual word embeddings based on the notion that semantically similar words appear in similar contexts. Two models can be distinguished based on the auxiliary task used: Continuous Bag of Words (CBOW) predicts a target word based on one or more context words, and Skip Gram (SG) predicts one or more context words based on a target word. Doc2Vec extends Word2Vec by also creating a low dimensional feature vector for each document, which in our case is the aggregation of all texts of a single user. This way, if two Scratch users use similar words, their document vectors should be close to each other in vector space. These document vectors were then fed into an SVM, similarly as with the bag-of-words model. The user texts are also subjected to the same pre-processing steps as described in the paragraph on the bag-of-words model.

Transformer The last model is a pre-trained roBERTa [Liu et al., 2019] Transformer model which has been fine-tuned for age and gender identification. Transformer models are one of the more recent developments in NLP and deviate from previous state-of-the-art models like Recurrent Neural Networks (RNNs) by building on attention mechanisms instead of sequential processing [Vaswani et al., 2017]. Because natural language is not processed sequentially by Transformers, parallel training becomes available in a greater degree than with RNNs. This allows Transformers to be trained on larger corpora than what used to be possible. These pre-trained models can be fine-tuned for specific downstream analysis tasks and have shown state-of-the-art performance. The pre-trained Transformer models have their own tokenizers, which is why we supplied the raw aggregated user texts instead of the pre-processed ones.

Figure 4.1 provides a schematic overview of the pre-training and fine-tuning process of a roBERTa model [Devlin et al., 2019; Liu et al., 2019]. RoBERTa's pre-training objective is to predict words that have been masked in the input sequence. The model deploys a dynamic masking strategy in which 15% of the input tokens are masked. Of these masked tokens, 80% are regularly masked, 10% are replaced by a random word, and 10% remain the original word. The resulting tokens are converted to input embeddings. Each input embedding is a concatenation of a token, segment, and position embedding. The roBERTa encoder stack transforms this input embedding to a bidirectional contextual representation. This means that the tokens were not processed in sequence (bidirectional) and two identical words may have different representations depending on their context (contextual). A softmax output layer converts these representations into probability vectors. Each value in a vector depicts the probability that a certain vocabulary token was originally at that place in the input sequence.

Fine-tuning a roBERTa model for a specific downstream analysis task, like gender prediction, largely uses the same methods. Because of the different objective, masking the

input sequence is no longer required. Furthermore, the [*CLS*] token, which indicates the beginning of an input sequence, is used as an output for the given classes of the analysis task.

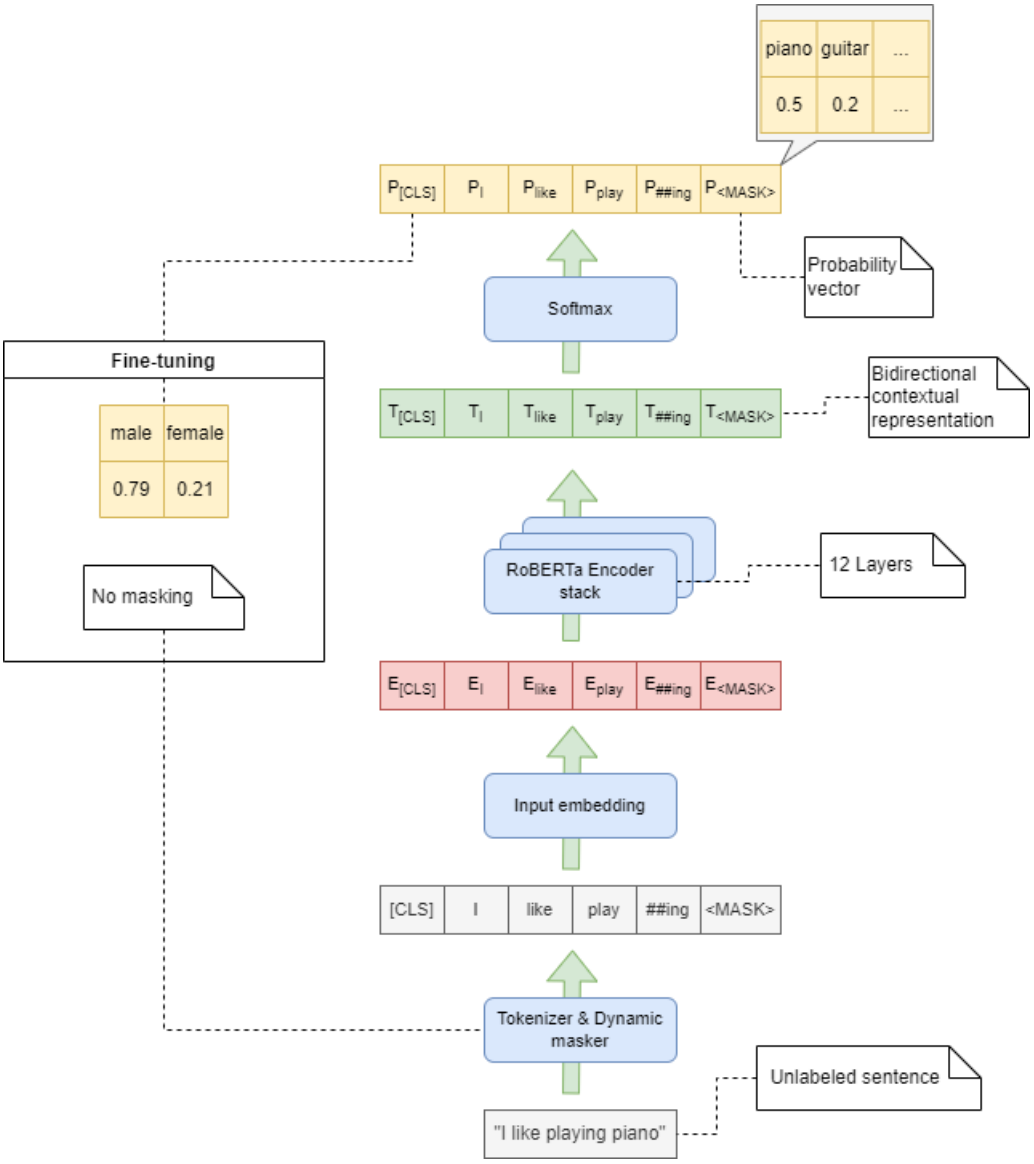


Figure 4.1: Schematic overview of the workings of pre-training and fine-tuning roBERTa

VISUAL USER DATA

We also used profile pictures as visual user data for the prediction of age and gender. We used the VGGNet16 model [Simonyan and Zisserman, 2015] to extract probability vectors from the profile pictures. Each vector has 1,000 probability values. Each value belongs to a certain class (e.g. 'dog' or 'cat') and represents the probability that that class is displayed by the image. These vectors are fed into an SVM, similarly to the bag-of-words and Doc2Vec methods.

Network user data

Another type of user data that we used instead of textual or visual features is network data. We constructed a network graph given the data of which other Scratch users each user in our dataset is following. In this network, each node represents a user from our dataset, and each directed edge represents a following relation. The Node2Vec method [Grover and Leskovec, 2016] was used to extract user embeddings for each node in the graph. It does so by generating biased random walks through the graph. These random walks were than fed into a Word2Vec skip-gram model. Here, each walk is treated as a sentence, and each visited node is treated as a word. The node embeddings are generated by continuously predicting the context nodes for each node in the graph (skip-gram), which we then fed into an SVM for the identification of age and gender. The process of generating node embeddings is visualised in Figure 4.2 [Stamile et al., 2021].

There are two main sampling strategies for generating random walks described by Grover and Leskovec [2016]. There is depth-first sampling, in which the random walk is biased towards visiting nodes that are not close to the previously visited node. This promotes homophily similarity: highly interconnected nodes (communities) should be close to each other in embedding space. There is also breadth-first sampling, in which the random walk is biased towards visiting nodes that are close to the previously visited node. This promotes structural equivalence: nodes with similar structural roles in a network should be close to each other in embedding space.

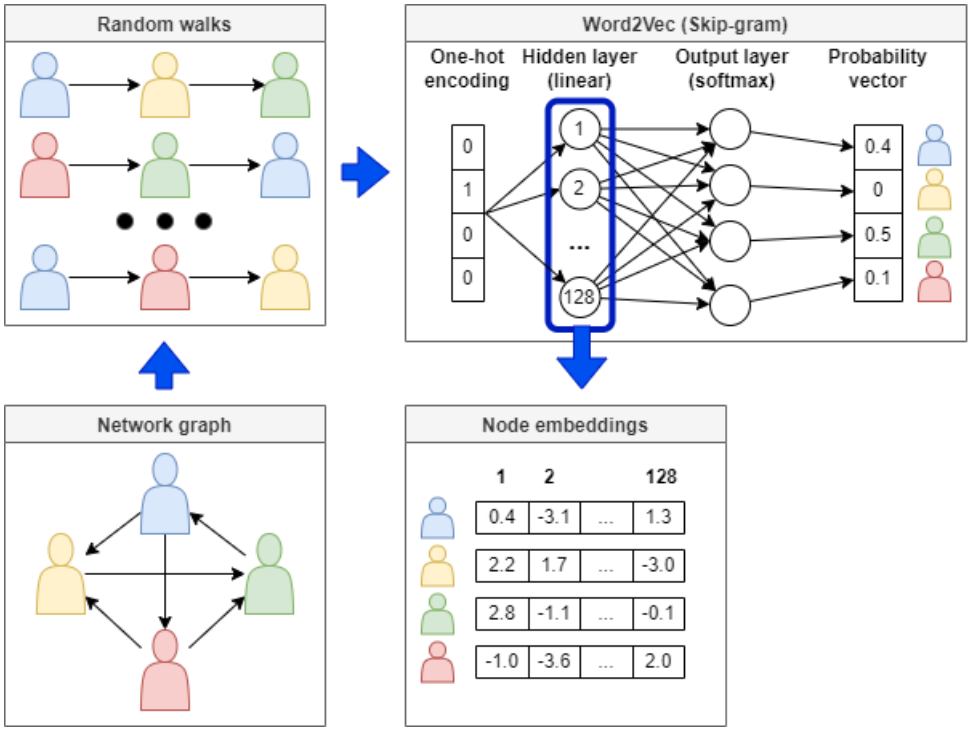


Figure 4.2: Schematic overview of the workings of generating node embeddings using Node2Vec with an example graph of four nodes, and a node embedding vector size of 128. The values in the probability vector that the Word2Vec model outputs depict, for each node, the probability that a randomly selected nearby node is that node.

4.2.2. Model validation

The performance of our models was measured using 5-fold cross-validation. This method involves splitting up the labelled data in 5 subsets, where each subset is used as a test set to evaluate the performance of the classifier once, while the other subsets are used for training the classifier. This should prevent the classifier from over-fitting on specific test cases. We have used the F1-score to measure the performance of a classifier, which is the harmonic mean between precision and recall. In order to identify which textual features our models considered important for classifying age and gender, we have used SHAP [Lundberg and Lee, 2017].

SHAP calculates importance values (Shapley values) of the features that are used by a prediction model. In our case, these features are words or tokens. It learns these values by leaving out features, and measuring its effect on the final prediction. A benefit of SHAP is that it allowed us to calculate the global importance of a feature (e.g. on a model-level), instead of just the local importance (e.g. on a prediction-level). We have implemented SHAP into our cross-validation process. Each iteration, after the text-based model has been trained on the training folds, we use SHAP to calculate the importance values for the features in the validation fold. The use of this method was required as the original raw features are lost in the process of transforming natural language to vector space. This means that the function that our textual models learned was no longer interpretable by humans. SHAP provided a way of explaining the predictions of our text-based models.

Our models which used textual data were also tested on texts of which disclosures of age and gender were temporarily removed. This allowed us to see how our models would perform when a direct disclosure of age and gender was not always present, which is the case in the unlabelled part of the dataset. Another approach that has been explored is the use of selective classification. This involves only identifying the age or gender of a user when the machine learning model reaches a certain probability threshold. The trade-off of this approach is that the model only classifies a portion of the dataset. A major benefit is an increase in performance.

These various validation metrics were then used to pick suitable models for both age and gender identification. These models were then used to classify the unlabelled part of the dataset. The resulting age and gender information allowed us to enrich the utilised dataset and answer our first research question.

4.3. Analysing block usage

The first step in answering the second research question was to collect block usage data. We queried the utilised database, that has been enriched with age and gender information, in order to retrieve block and project data for each user of a specific age group or gender. For the block usage analysis we have excluded projects that are remixes, as we could not determine which blocks were added by the user that remixed the project. This data was used to analyse the usage of different block types among the different age groups and genders. We defined the following block type taxonomy, which is almost identical to the one used in the Scratch web interface:

1. Motion
2. Looks

3. Sound
4. Events
5. Control
6. Sensing
7. Operators
8. Data
9. Procedure (combination of Procedure call, Procedure definition, and Procedure arguments block types)
10. Extensions (combination of Pen and Music block types)
11. Other (combination of MenuInput and Unknown block types)

We analysed the relative frequencies in which these block types are used within the projects of users of different age groups and genders. Furthermore, for each user, we identified whether they have used blocks of a certain type at least once in their projects or not.

The block usage of a user is represented as a ratio. For example, if a user has a usage rate of 0.4 for the block type 'Motion', then that means that 40% of their projects contain at least one 'Motion' type block. We used a ratio instead of absolute numbers in order to factor out the differences in total number of projects by different groups.

Significance tests were used to reveal any statistically significant differences between the block usage of the different age and gender groups. For gender, we used Wilcoxon rank-sum tests, which compare two independent groups of samples that are not normally distributed. For age, we used Kruskal-Wallis tests, which are an extension of the Wilcoxon rank-sum test for more than two independent groups. If a significant difference was found, a post hoc Dunn's test was used to reveal which specific age groups have significantly different means. The null hypothesis of these tests is that the distributions of all samples are equal, while the alternative hypothesis is that the distributions are not equal.

Due to the fact that we are dealing with large sample sizes, even very small differences can become significant. Especially since our alternative hypothesis does not specify a greater than or less than relation, the null hypothesis is easily rejected. Therefore, it was important to also look at the effect sizes. The effect size is defined as the magnitude of the difference between groups and is independent of sample sizes Sullivan and Feinn [2012]. General rules of thumb for interpreting the effect size can be seen in Table 4.3.

Effect size	η^2	r**-family**
Small	0.01 - 0.06	0.1 - 0.3
Moderate	0.06 - 0.14	0.3 - 0.5
Large	0.14+	0.5+

Table 4.3: Rules of thumb for interpreting effect sizes [Cohen, 1988]

4.4. ANALYSING PROGRAMMING CONCEPT USAGE

The first step in answering the third research question was to determine which programming concepts we wanted to analyse. Comparing programming concepts analysed by other works [Aivaloglou and Hermans, 2016; Fields et al., 2017; Graßl et al., 2021; Hermans and Aivaloglou, 2017; Zeevaarders and Aivaloglou, 2021], we concluded that the concepts of conditionals, coordination, iteration, and variables should at least be analysed, as these are present in almost all of the mentioned works and because they are considered the most challenging for novice Scratch users. This exact composition of programming concepts was used by Graßl et al. [2021], who also provided a mapping between these concepts and the blocks that use these concepts. We used this mapping with some slight modifications, which can be found in Table 4.4.

Programming concept	**Scratch Block type**	**Scratch Block**
Conditionals	Control	if
	Control	if else
	Control	wait until
	Control	repeat until
Coordination	Events	when i receive broadcast
	Events	broadcast
	Events	broadcast and wait
	Control	wait until
	Control	stop
Iteration	Control	forever
	Control	repeat
	Control	repeat until
Variables	Data	read
	Data	change by
	Data	set to
	Data	show
	Data	hide

Table 4.4: Mapping between programming concepts and corresponding Scratch blocks

For each user, we calculated the usage rate of these concepts, similarly to how we calculated the block type usage rate in Section 4.3. For example, if a user has a usage rate of 0.4 for the concept 'Iteration', then that means that in 40% of their projects the user has used the 'forever', 'repeat' or 'repeat until' blocks at least once. The same statistical tests were used as in Section 4.3. These tests allowed us to identify any differences or similarities in the use of programming concepts between users of different age groups and genders.

5

MACHINE LEARNING MODEL SELECTION

Before we selected which machine learning models to use for the automatic identification of age and gender of Scratch users, a training dataset was constructed. This training dataset contains age and gender information for more than 6,000 Scratch users. This dataset was used to validate the selected machine learning models. The results of this validation along with other criteria allowed us to select which machine learning models to use for eliciting age and gender information. The following sections describe this process in more detail.

5.1. TRAINING DATASET CONSTRUCTION

The construction of the training dataset used the dataset described in Section 4.1 as a starting point. The regular expressions described in Section 4.1.2 were used to filter out disclosures of age and gender among user texts. Before matching our regular expressions against all status and bio texts, non-English texts were filtered out. If more than one regular expression matched on the same body of text, only one match was stored. This was due to the fact that all bodies of text will be manually checked and there was no reason in checking the same body of text twice. Furthermore, only status and bio texts were considered. The reason comment texts were excluded is because it is more difficult to assess the truthfulness of comments as they have to be considered in the context of the comment chain they are part of and the profile they are posted on. Status and bio texts are standalone texts and were thus easier to evaluate. Also, texts that contain an implausible age (under 4 or over 80 years old) were stored separately.

Our set of regular expressions resulted in a total of 6,299 matches (Table 5.1). These texts were then all manually checked for any false positives. These false positives include instances where the context negates the phrases resembled by the regular expression (e.g. "call me Elle so ppl don't think I'm a boy" extracts "male"). Other instances include role-playing, where users pretend to be a (fictional) character, or humor/provocation, where users try to be funny or shock by providing false information (usually an age between 18-80). Table 5.1 also shows the number of false positives per regular expression.

Manual adjustments were also made if the extracted age or gender was incorrect, but the correct age or gender was present in the text. In case of AGE or GENDER type regular expressions, if the information that was not checked for was present in the text, it would be manually added as well. The benefit of this is that it allowed us to obtain age or gender information from phrases that are hard to capture in regular expression without generating

Regular expression	Matches	False positives	Adjustments
AGE_GENDER	637	9	4
GENDER	2,922	42	818
AGE	2,704	26	176
Implausible age	36	35	0
Total	**6,299**	**112**	**998**

Table 5.1: Number of matches, manually identified false positives, and manual adjustments or additions per regular expression

a lot of false positives. For example, "I am the older brother of ..." indicates that the user is a male. Table 5.1 also shows how many adjustments or additions were made per regular expression. Note the number of additions is high for the GENDER regular expression because it was used prior to the AGE one. This means that a large portion of the texts would also have been retrieved by the AGE regular expression.

Finally, we also labelled interesting information in these texts that may be useful within or outside the scope of this project. Table 5.2 shows which special labels were used, contains a short description of each label, and shows how many texts have been labelled per category.

Label	Description	Count
REAL_NAME	The user provides their real name in the text	649
SCHOOL	Contains more information about their school career (e.g. middle school, which grade, home-schooled)	478
FAMILY	Information about their family composition (e.g. siblings/cousins)	386
LOCATION	More details about their location then just their country (e.g. state/city)	250
DATE_OF_BIRTH	More information about the day, month OR year of their date of birth	188
RELIGION	Information about their religion (e.g. Christianity, but also atheism)	176
PERSONAL_DETAILS	Variety of personal details like autism, mental health, disabilities, personality type	86
PROG_EXPERIENCE	Information about experience in other programming languages	80
APPEARANCE	Information about the user's real life appearance (e.g. eye or hair color)	58
RELATIONSHIP	Information about the user's relationship status (single or in a relationship)	43
LGBTQ	The user explicitly states their sexuality/gender to be different than heterosexuality or male/female gender	40
Total		**2,434**

Table 5.2: Number of assigned special labels and their descriptions

After all results had been checked, we were able to calculate statistics about the training

data. Table 5.3 contains the gender distribution in the training data. Figure 5.1 contains the age distribution in the training data. It shows a similar distribution as the age distribution of all Scratch users[1]. Table 5.4 contains the number of users among the four age groups.

Gender	Number of users
Male	1678 (44.42%)
Female	2100 (55.58%)
Total	**3,778**

Table 5.3: Gender distribution in training data

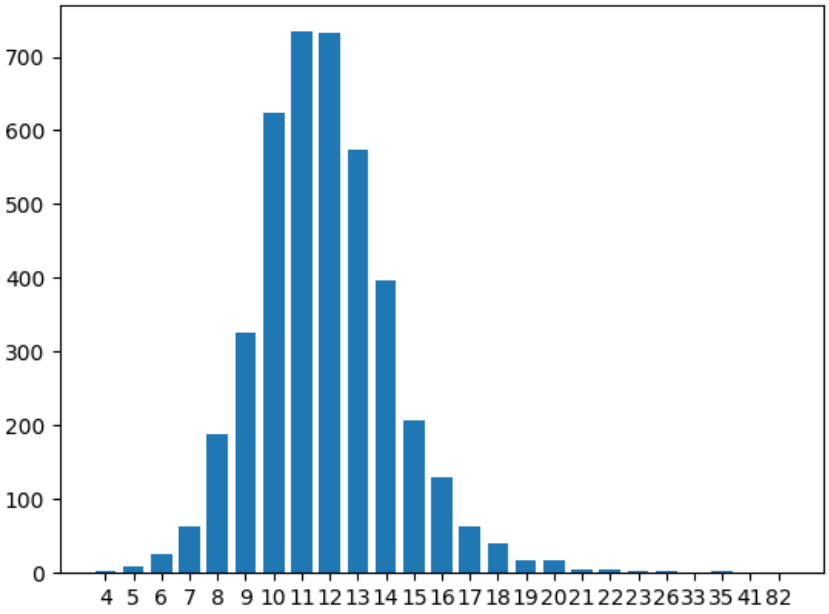


Figure 5.1: Age distribution in training data (total: 4170)

Age group	Number of users
4-9	616
10-11	1358
12-13	1306
14+	890

Table 5.4: Age group distribution in training data

[1] https://scratch.mit.edu/statistics/

5.2. MODEL VALIDATION RESULTS

Table 5.5 contains the 5-fold cross-validation F1-scores of the selected age and gender identification machine learning models. The statistical baseline always predicts the most common age group ('10-11') and gender ('female'). We see that the text-based models achieve the highest performance scores for both age and gender identification. Among the other models, only Node2Vec gender identification comes close to these performance levels. An overview of the implementation details and parameter values of these models can be found in Appendix B.

Data type	Model	Age	Gender
Textual	Bag-of-words	0.808	0.908
	Doc2Vec	0.839	0.930
	Transformer	0.988	0.990
Visual	VGGNet16	0.283	0.558
Network	Node2Vec	0.331	0.797
N/A	Statistical baseline	0.161	0.400

Table 5.5: 5-fold cross-validation F1-score results for age and gender identification

SHAP was used to get more insight in the contributions of certain words in the predictions of the our text-based models. All the Shapley values that have been calculated during cross-validation are combined and plotted in bar charts, which can be found in Appendix A. We used two ways to aggregate and display the Shapley values in a bar plot:

1. Mean: displays the features with the highest average importance values. This is useful for finding features that have a consistently high importance value across all predictions.

2. Sum: displays the features with the highest sum of importance values. This is useful for assessing the frequency in which features with high importance values occur.

The SHAP analysis results show that our Transformer model places high importance on the words 'boy' and 'girl' for gender identification and numbers corresponding to age groups for age identification. We also see domain knowledge of the English language, as other gender words (e.g. 'Emily' or 'brother') and written out numbers (e.g. 'twelve') are also part of the top features. For both our Doc2Vec and bag-of-words models, we see that the mean Shapley values contain strange features. This is likely due to the fact that the input sequences are long. This results in high frequency words having lower mean values, while important words that occur infrequently or even once retain a higher mean value. The sum Shapley values show similar results as with the Transformer model. However, unlike with the Transformer model, we also see words with high sum values that fall outside the gender domain (e.g. 'minecraft', 'mario', 'star', and 'war', but also 'love', 'cat', 'art').

Table 5.6 contains cross-validation F1-score results, but with removing age/gender disclosures from the validation folds. This way, the text-based models are trained on the same data as before, but evaluated against texts that do not contain age/gender disclosures. We see that for the Transformer model, performance metrics drop down to the level close to the statistical baseline. For the Doc2Vec and Bag-of-words models, we observe a drop in performance too, but not as extreme as with the Transformer model.

Model	Age	Gender
Bag-of-words	0.400	0.742
Doc2Vec	0.428	0.755
Transformer	0.385	0.557

Table 5.6: 5-fold cross-validation F1-score results, age and gender disclosures removed from validation folds

Table 5.7 shows the cross-validation results of the Transformer model when only allowing predictions with a probability higher than 0.985. We see that for the regular training data, this pushes the F1-score to be almost equal to 1, at the cost of only classifying a few percent less data. When we remove age and gender disclosures from the validation fold, you can see that the performance metrics increase by a lot, as they were almost at the level of the statistical baseline before setting a probability threshold. However, this does come at the cost of only being able to classify roughly 20% to 25% of the data.

	Age		**Gender**	
	F1-score	%-classified	F1-score	%-classified
Regular	0.997	98.002%	0.996	96.904%
Age/gender removed	0.800	20.077%	0.737	24.286%

Table 5.7: Selective classification cross-validation results (Transformer model, probability threshold: 0.985)

5.3. Training data bias

Based on F1-score results alone, our Transformer model would seem best suitable for the automatic identification of age and gender. However, F1-scores that close to one raise suspicion that the model is overfitting on the training data. Our plan is to use a classifier on the entire dataset so that we can identify the age and gender of these users. For this reason, we want to be able to argue that the performance that is achieved on the training data also holds for the entire dataset. The reality is that our training data is biased: it contains only texts of users that have voluntarily disclosed their age and gender on their Scratch platform. This means that this subset of users may be a sample that is not indicative for the entire dataset.

For all text-based models, our SHAP analysis results indicated (over)fitting on the features that directly indicate the age and gender of a user (e.g. numbers and gender words). The most likely explanation for this behaviour is that, due to the bias in the training data, all texts contain at least one disclosure of age or gender. The cross-validation results with removal of age and gender disclosures from the validation folds confirmed that the performance of our text-based models relies heavily on the presence of these disclosures. This made using text-based models problematic, as the user texts of users outside of the training data are not guaranteed to contain direct disclosures of age and gender.

5.4. Selected models

Two main criteria have been considered in the model selection process: performance and scale. Performance was measured using 5-fold cross-validation on the training set. Scale was measured using the number of users of whom a machine learning model could identify

the age and gender.

In Section 5.3, we discussed that the performance of text-based models can be unreliable due to the bias in the training data. However, selective classification results have shown to be able to mitigate a large part of the performance drop when identifying user texts in which we have removed age and gender disclosures. Therefore, we have selected our **fine-tuned roBERTa Transformer model with selective classification** for **age** identification of Scratch users, even though this method results in being able to classify a smaller portion of the dataset. For **gender** identification, we have selected our **Node2Vec model**. There were three reasons for choosing this model. First, the cross-validation results have shown similar performance levels as our selective classification Transformer model. Second, the Node2Vec model is not affected by the presence of age and gender disclosures in the user texts as it only uses network data. Third, we have network data for 336.394 users, while we have textual data for 183,292 users. This means that we were able to elicit more gender information using a network-based model than using a text-based model, as it operates on a larger scale.

6

RESULTS

6.1. RQ1: AGE AND GENDER INFORMATION

We used our Node2Vec model for the identification of gender of the Scratch users in our dataset. We selected the users from the unlabelled part of the dataset of which we have network data. After that, node embeddings were extracted for these users from the Node2Vec model we trained in the identification phase. An SVM was then trained on the node embedding and gender pairs from the training data. Finally, the SVM was used to predict the gender of the unlabelled users based their node embeddings. The results can be viewed in Table 6.1 and are visualised in Figure 6.1b.

	Female	**Male**	**Total**
Training	2,100 (55.58%)	1,678 (44.42%)	3,778
Identified	116,850 (34.74%)	219,544 (65.26%)	336,394
Total	118,950 (34.97%)	221,222 (65.03%)	340,172

Table 6.1: Gender identification results (number of users)

The first thing that stands out is the amount of users of whom we were able to identify their gender. The resulting data is nearly 90 times the size of the training data. Together with the training data, we have been able to identify the gender for 83.56% of users in the dataset. A second interesting aspect is the distribution of male and female users. Fields et al. [2017] report that the distribution of self-reported gender on the Scratch site was 33% female and 67% male as of December 2011, and 38% female, 58% male and 4% other/NA as of April 2015. We see a similar distribution for the identified part of the dataset. This is a good sign, especially considering that the training data was skewed towards female users. So even though our SVM was trained on data with a gender distribution that is not representative of the Scratch population, it was still able to identify a representative gender distribution among the unlabelled users.

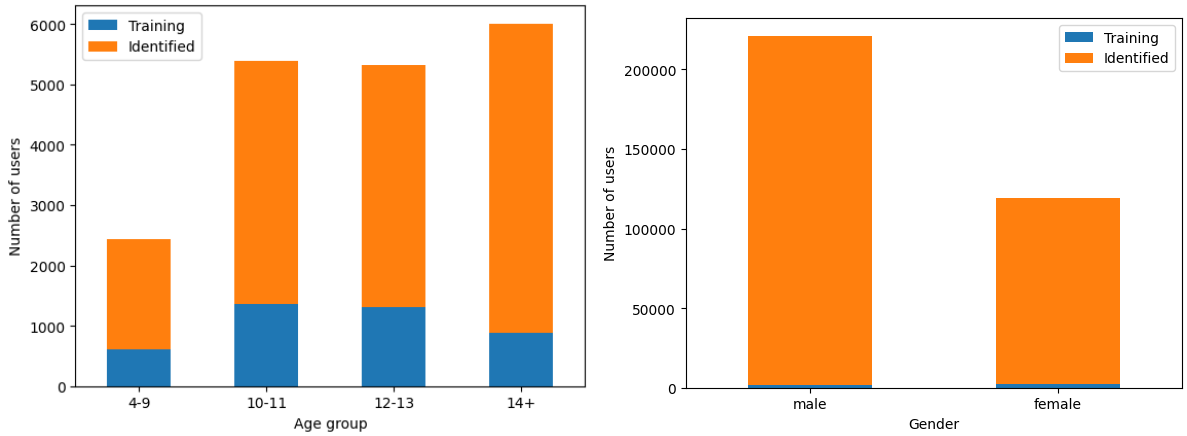
For the identification of age, we used our fine-tuned roBERTa Transformer model with selective classification. As our Transformer model makes use of textual data, we first selected all users of which we have (English) textual data. Our model was then fine-tuned on

all training data and used to predict the age groups of users of which we did not already have age information. Age groups of users were only identified when the probability of the prediction was higher than 0.985. This threshold resulted in an F1-score of 0.80 on the training data when age and gender disclosures were removed. This performance level is in line with our Node2Vec model for gender identification. The results can be viewed in Table 6.2 and are visualised in Figure 6.1a.

	4-9	**10-11**	**12-13**	**14+**	**Total**
Training	616 (14.77%)	1,358 (32.57%)	1,306 (31.32%)	890 (21.34%)	4,170
Identified	1826 (12.18%)	4031 (26.89%)	4015 (26.78%)	5121 (34.16%)	14,993
Total	2,442 (12.74%)	5,389 (28.12%)	5,321 (27.77%)	6,011 (31.37%)	19,163

Table 6.2: Age group identification results (number of users)

First thing to note is the total number of users of whom we identified their age group. Based on the results of the training data, we expected to be able to identify the age group for 20.077% of the users. From the 179,385 users of which we have English textual data, this would come down to 36,015 users. However, our model identified the age of 14,993 users, which is only 8.358% of the total number of users. Furthermore, we see that the distribution of age groups is skewed towards the 14+ age group. This is different from the training data, of which we had already shown to have a similar age distribution as the entire Scratch population.



(a) Age

(b) Gender

Figure 6.1: Age group and gender distributions for training data and identified users

Both the identified and labelled age and gender information, as well as the scraped user data, were imported in the utilised dataset. The resulting Entity Relationship Diagram can be viewed in Figure 6.2. The *authors* table now contains fields for the age group and gender. It also contains the *age_type* and *gender_type* fields. Their values can be either 'ANNOTATED' or 'PREDICTED', indicating whether the age or gender was manually annotated (part of the training data) or predicted by a machine learning model. Finally, the *age*

field contains a user's age as an integer value, which is only applicable for annotated age data. The enriched version of the dataset is available online[1].

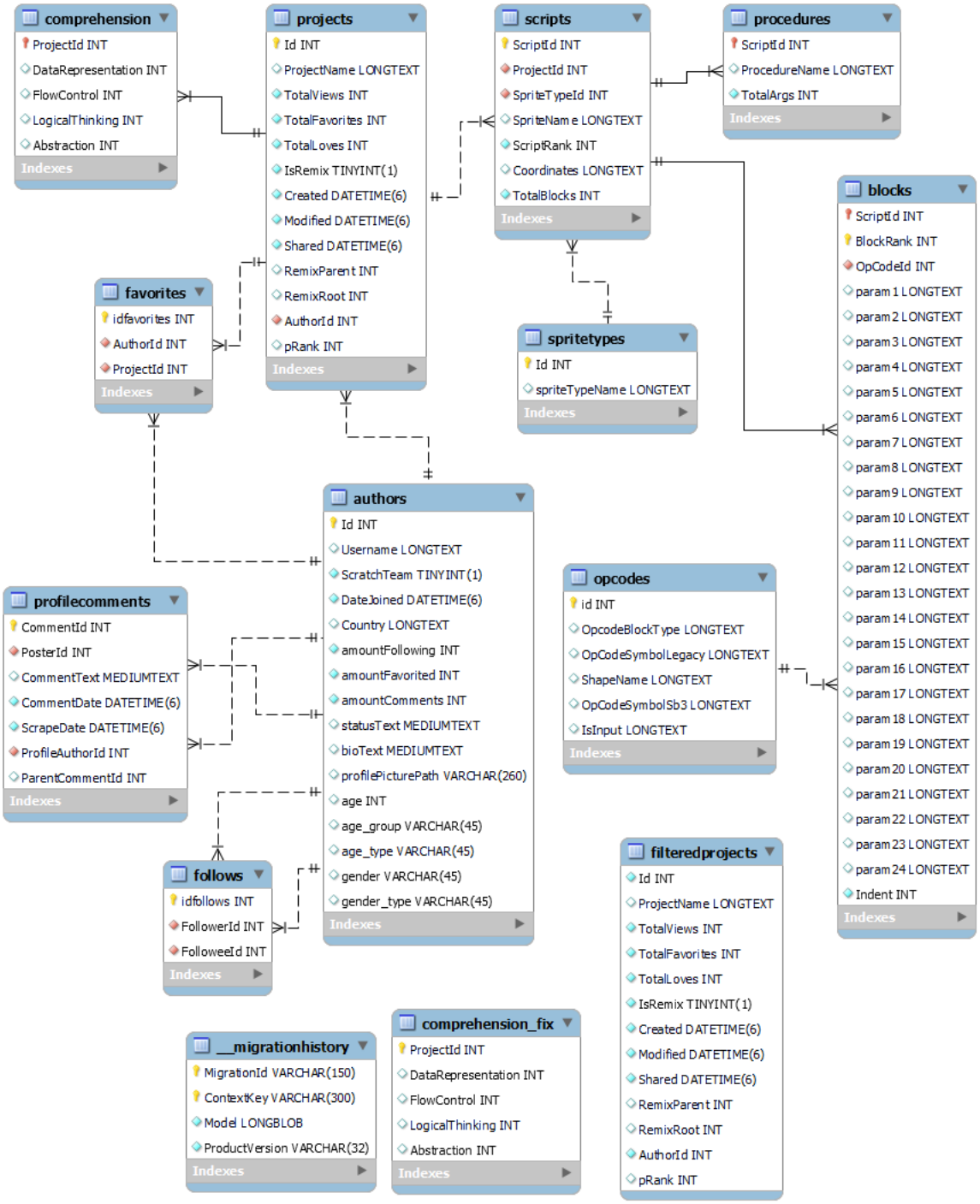


Figure 6.2: Entity Relationship Diagram of the enriched database by Zeevaarders and Aivaloglou [2021] after importing the age and gender data.

[1] https://1drv.ms/u/s!AofN0KrcSoj_oJAQ5Ypr2Me5-XEt8g?e=8aMC8u

6.2. RQ2: BLOCK TYPE USAGE

The analysis of block type usage was performed on the users for which age and gender was identified or manually labelled, and for the projects that were not remixes. Tables 6.3 and 6.4 show the amount of users, projects, and blocks that were considered in our analysis. Figure 6.3 visualises the distributions of these users, projects, and blocks for the age and gender groups. It shows that male users and users of the highest age group have relatively more projects and use relatively more blocks in these projects than the other groups.

Tables 6.5 and 6.6 show the top ten blocks used by the age and gender groups. For gender, we see a number two spot for the 'variable' (Data) block, which does not occur in the female top ten at all. We also see that 'set variable to' (Data) and 'equals' (Operator) blocks occur only in the male top ten, while the 'show' (Looks), 'go to x/y' (Motion), and 'repeat' (Control) blocks occur only in the female top ten. Furthermore, the difference between the rank 1 and rank 10 blocks is 6.46 percent point for female users, while this difference is only 2.21 percent point for male users.

	# of users		# of projects		# of blocks	
Remixes	Included	Excluded	Included	Excluded	Included	Excluded
4-9	1,943	1,933	61,616	53,973	7,139,715	4,586,960
10-11	4,295	4,270	109,757	96,148	11,783.738	7,792,305
12-13	4,254	4,226	110,902	97,569	11,994,608	9,199,378
14+	4,537	4,499	170,885	152,141	23,902,527	19,920,730
Total	15,029	14,928	453,160	399,831	54,820,588	41,499,373

Table 6.3: User, project, and block data for users of different age groups (remixes included/excluded)

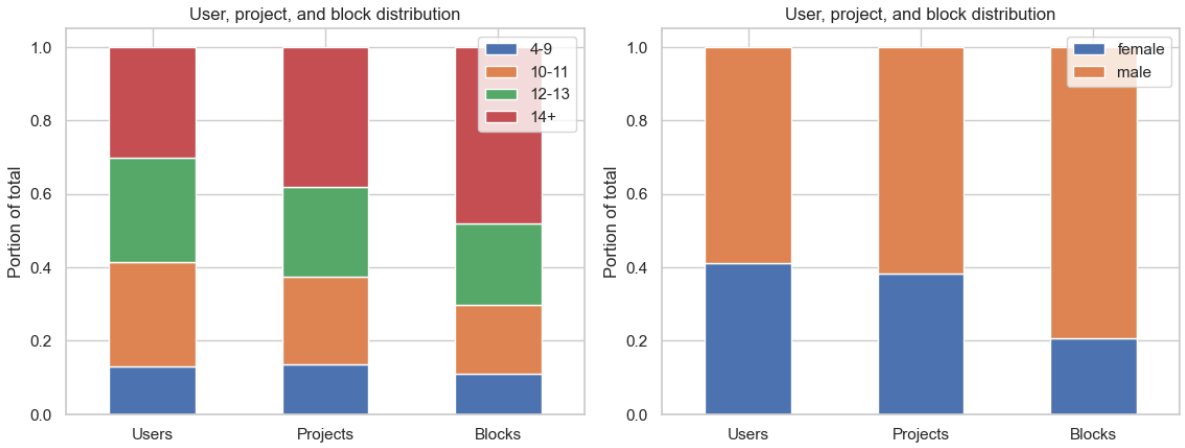
	# of users		# of projects		# of blocks	
Remixes	Included	Excluded	Included	Excluded	Included	Excluded
male	97,322	96,694	2,314,346	2,098,570	396,060,554	302,495,753
female	68,287	67,678	1,492,317	1,296,703	104,087,169	79,303,306
Total	165,609	164,372	3,806,663	3,395,273	500,147,723	381,799,059

Table 6.4: User, project, and block data for users of different genders (remixes included/excluded)

	4-9		10-11		12-13		14+	
Rank	**Block**	**%**	**Block**	**%**	**Block**	**%**	**Block**	**%**
1	data_variable	7.43%	data_variable	6.22%	control_wait	6.72%	control_wait	6.25%
2	control_wait	5.64%	control_wait	6.1%	looks_switchcostumeto	5.4%	looks_switchcostumeto	5.36%
3	event_whenflagclicked	4.72%	looks_switchcostumeto	4.91%	data_variable	5.23%	data_variable	5.31%
4	data_setvariableto	4.44%	event_whenflagclicked	4.89%	event_whenflagclicked	5.03%	looks_hide	4.28%
5	looks_switchcostumeto	4.39%	looks_hide	4.38%	looks_hide	4.52%	event_whenbroadcastreceived	4.26%
6	looks_hide	3.92%	event_whenbroadcastreceived	3.97%	event_whenbroadcastreceived	4.24%	event_whenflagclicked	4.25%
7	event_whenbroadcastreceived	3.88%	data_setvariableto	3.76%	control_if	3.79%	operator_equals	3.89%
8	control_if	3.7%	control_if	3.68%	data_setvariableto	3.59%	data_setvariableto	3.81%
9	operator_equals	3.68%	operator_equals	3.45%	operator_equals	3.35%	control_if	3.66%
10	control_forever	2.93%	control_forever	2.99%	looks_show	3.03%	looks_show	2.84%

Table 6.5: Top ten blocks per age group with percentage in comparison to total blocks used by each group

The relative frequencies in which block types are used within the projects of users of different age groups and genders are visualised in Figures 6.4 and 6.5. We see similar distributions of block frequencies across the different age groups with no clear trends as the age



(a) Age (b) Gender

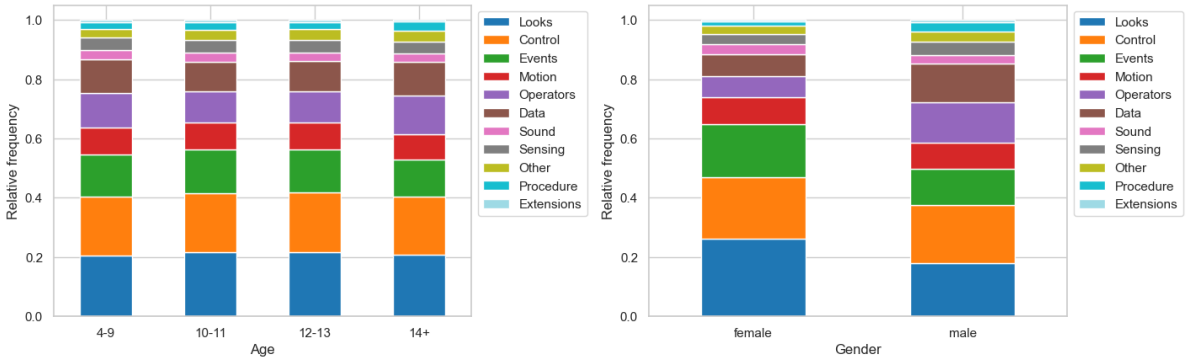
Figure 6.3: Distribution of users, projects, and blocks for the different age groups and genders

	Female		Male	
Rank	**Block**	**%**	**Block**	**%**
1	control_wait	9.09%	control_wait	5.34%
2	event_whenflagclicked	6.8%	data_variable	4.96%
3	looks_switchcostumeto	6.24%	event_whenflagclicked	4.47%
4	looks_hide	5.52%	control_if	4.29%
5	event_whenbroadcastreceived	4.69%	looks_switchcostumeto	4.28%
6	looks_show	3.69%	data_setvariableto	4.07%
7	control_forever	3.38%	looks_hide	4.01%
8	motion_gotoxy	3.06%	event_whenbroadcastreceived	3.96%
9	control_if	2.83%	operator_equals	3.78%
10	control_repeat	2.63%	control_forever	3.13%

Table 6.6: Top ten blocks per gender with percentage in comparison to total blocks used by each group

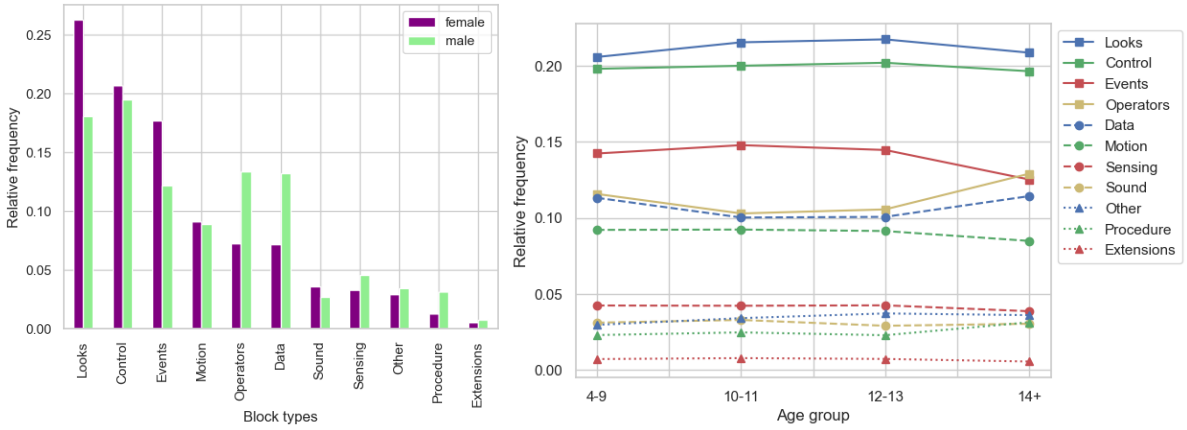
group gets higher. For gender, we do see notable differences in block type frequency distributions. Looks, Control, and Events blocks make up a total of 64.7% of the blocks used in projects created by female users, while this percentage is only 49.8% for male users. On the other hand, the mean relative frequency of Data and Operator blocks used by male users is nearly two times that by female users.

The usage rate of each block type for all users that have been analysed is visualised in Figure 6.6. Tables 6.7 and 6.8 summarise the observed effects sizes as well as the corresponding p-values of the differences in usage rate between the different age groups and genders. In Table 6.7 we see that only one effect size reaches the threshold of a small effect, namely for the 'Other' block type. This means that even though significant differences were found for almost all of the other block types, the magnitudes of these differences are so very small that they are essentially meaningless. The results of the Dunn's tests show that for the 'Other' block type, the usage rate is significantly higher for the '14+' age group with respect to the other groups, and the usage rate of the '12-13' age group of significantly higher than the '10-11' age group. This effect can also be seen in Figure 6.7, in which the mean usage



(a) Age (14,928 users, 399,831 projects) (b) Gender (164,372 users, 3,395,273 projects)

Figure 6.4: Block type relative frequencies for all age groups and genders

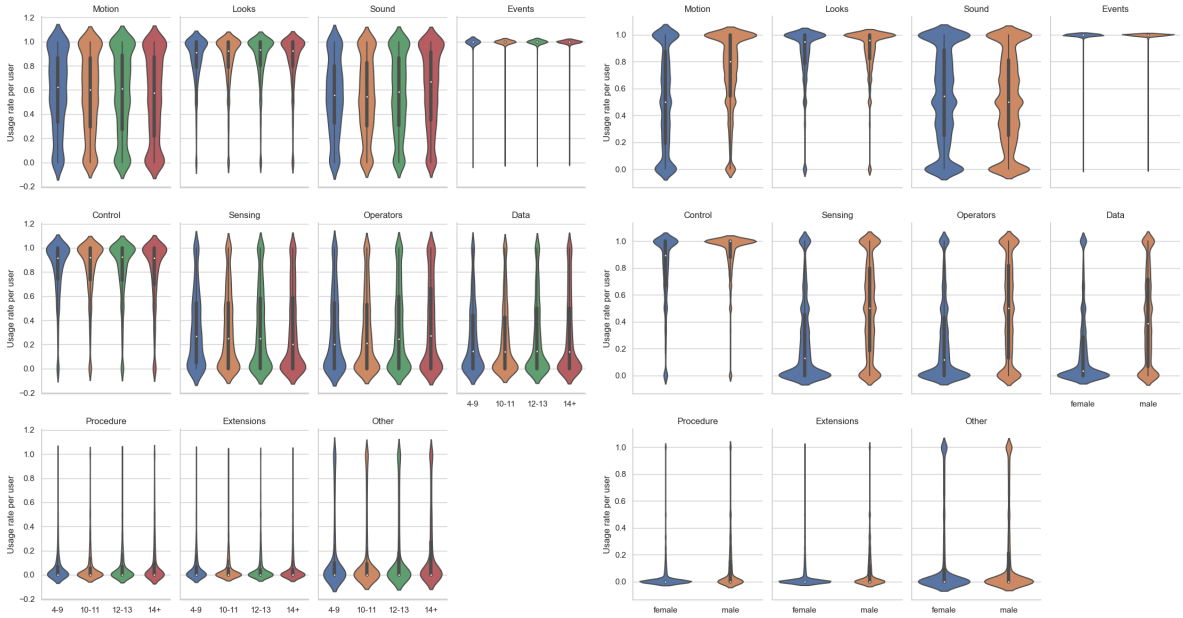


(a) Gender (164,372 users, 3,395,273 projects) (b) Age (14,928 users, 399,831 projects)

Figure 6.5: Block type relative frequencies for all age groups and genders

rates are plotted.

The p-values and effect sizes of the differences between users of different gender are shown in Table 6.8. First thing to note is that a significant difference was found for all block types. We see that for the block types 'Other', 'Sound', 'Looks', and 'Events', the effect sizes are too small to be considered a small effect. We can therefore deem these differences meaningless. However, for the block types 'Procedure', 'Extensions', 'Motion', and 'Control' we do see a small effect size. For the block types 'Data', 'Operators', and 'Sensing' we even see a moderate effect size. Figure 6.7 compares the mean usage rates of these block types by male and female users and reveals that male users have used the block types of which we saw a small or moderate difference in more of their projects.



(a) Age (14,928 users, 399,831 projects)

(b) Gender (164,372 users, 3,395,273 projects)

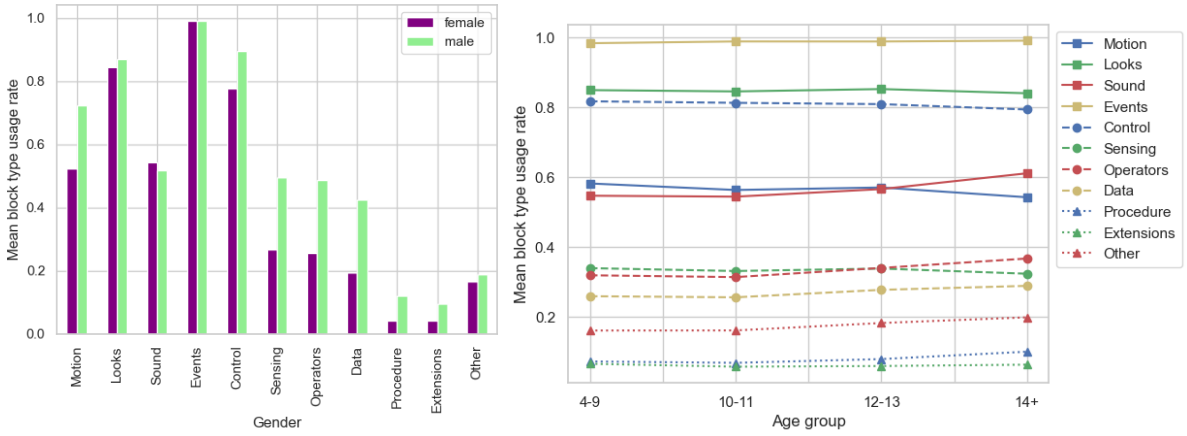
Figure 6.6: Block type usage rate distributions for all age groups and genders

Block type	**p-value**	η^2
Other	9.27e-41*	0.0125
Sound	2.47e-24*	0.00737
Procedure	3.13e-12*	0.00359
Operators	0.0000000077*	0.00252
Extensions	0.0000387*	0.00135
Events	0.000107*	0.00120
Motion	0.000212*	0.00111
Sensing	0.000638*	0.000953
Looks	0.00585*	0.000637
Control	0.0452*	0.000338
Data	0.102	0.000215

Table 6.7: Effect sizes of differences in block type usage rates between age groups. $\alpha = 0.05$. red: small effect, yellow: moderate effect, green: large effect

Block type	**p-value**	**r** $(z/\sqrt{N})$
Data	2.2e-16*	0.340
Sensing	2.2e-16*	0.324
Operators	2.2e-16*	0.311
Motion	2.2e-16*	0.267
Procedure	2.2e-16*	0.240
Control	2.2e-16*	0.228
Extensions	2.2e-16*	0.211
Other	2.2e-16*	0.0702
Sound	2.2e-16*	0.0404
Looks	2.2e-16*	0.0339
Events	2.814e-10*	0.0156

Table 6.8: Effect sizes of differences in block type usage rates between genders. $\alpha = 0.05$. red: small effect, yellow: moderate effect, green: large effect

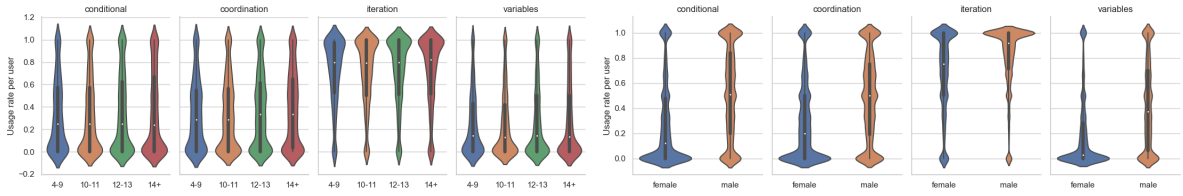


(a) Gender (164,372 users, 3,395,273 projects) (b) Age (14,928 users, 399,831 projects)

Figure 6.7: Mean block type usage rates for all age groups and genders

6.3. RQ3: PROGRAMMING CONCEPT USAGE

Similar to our block type analysis, the analysis of programming concept usage was performed on the users for which age and gender was identified or manually labelled, and for the projects that were not remixes. Figure 6.8 visualises the usage rate of each programming concept for all users that have been analysed. The detailed results of the significance tests comparing the usage rates are available online[2]. Tables 6.9 and 6.10 summarise the p-values and effect sizes of the programming concept usage differences between users of different age groups and genders. For age, we see that a significant difference was found for the usage of coordination and iteration concepts. However, the effect sizes of both of these differences are too small to be considered a small effect. For gender, we see significant differences for all programming concepts. The effect sizes show a small effect for the Coordination and Iteration concepts, and a moderate effect for the Conditionals and Variables concepts. Figure 6.9 compares the mean programming concept usage of male and female users. It shows that male users use all of the programming concepts in more of their projects than female users.



(a) Age (14,928 users, 399,831 projects)

(b) Gender (164,372 users, 3,395,273 projects)

Figure 6.8: Programming concept usage by Scratch users of different age groups and genders

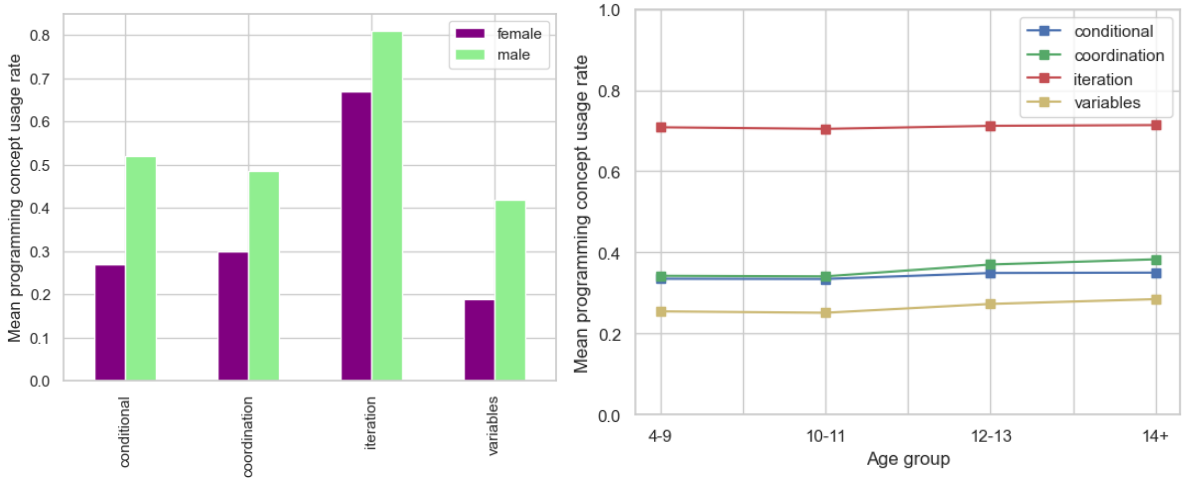
Programming concept	**p-value**	η^2
Coordination	0.00000000423*	0.00261
Iteration	0.0343*	0.000378
Variables	0.0768	0.000258
Conditionals	0.632	-0.0000857

Table 6.9: Effect sizes of differences in programming concept usage rates between age groups. $\alpha = 0.05$. red: small effect, yellow: moderate effect, green: large effect

Programming concept	**p-value**	**r** $(z/\sqrt{N})$
Conditionals	2.2e-16*	0.342
Variables	2.2e-16*	0.340
Coordination	2.2e-16*	0.265
Iteration	2.2e-16*	0.222

Table 6.10: Effect sizes of differences in programming concept usage rates between genders. $\alpha = 0.05$. red: small effect, yellow: moderate effect, green: large effect

[2] https://1drv.ms/u/s!AofN0KrcSoj_oJAQ5Ypr2Me5-XEt8g?e=8aMC8u



(a) Gender (164,372 users, 3,395,273 projects) (b) Age (14,928 users, 399,831 projects)

Figure 6.9: Mean programming concept usage rates for all age groups and genders

7

Discussion

The elicited age and gender information of Scratch users in the utilised dataset shows that the use of machine learning methods was especially effective for the identification of gender. The use of social network data resulted in performance levels that neared those of text-based methods. The major benefit was the abundance of network data that was present in the dataset. This is due to the way the dataset by Zeevaarders and Aivaloglou [2021] was scraped; They recursively scraped user data of Scratch users and their followers.

We were not able to elicit nearly as much age information. This was partly due to textual data being not as common as network data, and partly due to the selective classification method that was used. These findings are important to consider when applying these methods in future work. For existing datasets, these methods require the presence of either network or textual data depending on the information one wants to obtain. For new datasets, it may influence what data should be scraped. Furthermore, the quantity of elicited information may be an important factor for choosing which methods to use. For example, our block type and programming concept analyses considered almost 15.000 users of which we obtained age information. In hindsight, these quantities could have been obtained by manual annotation as well. For gender, however, our analyses considered over 160.000 users. In this case, manual annotation would not be an option if we wanted to obtain similar quantities.

The block type and programming concept usage analyses results have shown that there are hardly any differences between the age groups. To the best of our knowledge, there are no other works that have studied the relationship between age and block type and programming concept usage to compare to. The work by Hermans and Aivaloglou [2017] did observe a statistically significant difference between the grades of 11-12 and 13-14 year old students on the Operator and Procedure concepts. However, this concerns proficiency of these concepts, not their usage.

We did observe many differences in block type and programming concept usage between male and female users. For all of these observed differences, we found that male users have higher usage rates of the given block types (Data, Sensing, Operators, Motion, Procedure, Control, and Extensions) and programming concepts (Conditionals, Variables, Coordination, and Iteration) than female users. Within the projects created by female users, Looks, Control, and Events blocks occur more frequently than in projects created by male users. These results illustrate gender differences like those observed in some of the

other works in the field. For instance, Funke and Geldreich [2017] also observed increased usage of Motion type blocks by male users. Graßl et al. [2021] found that, in addition to Motion blocks, Sensing blocks are used more frequently by boys. The authors also found an increased usage of the conditionals and loops concepts by boys. Finally, Fields et al. [2017] observed that the groups of Scratch users that display mastery of the most advanced programming concepts and block type are disproportionately occupied by male users. All of these findings are in line with what we observed. However, our sample size indicates that the gender differences are present throughout the Scratch online community on a more widespread scale than what was previously observed.

7.1. Future work

Future work can be dedicated to deepening the understanding of the gender differences we observed. This can be done by applying our gender identification model to other existing datasets or by scraping new datasets with more recent projects and users. The use of more advanced analysis frameworks can help understand the causes of the observed gender differences. For instance, Funke and Geldreich [2017] and Graßl et al. [2021] found that differences in block type and programming concept usage may be caused by different project type preferences. Constructing a framework of concrete criteria that determine the type of a project can help validating these findings on a larger scale. Other aspects can be analysed as well. For instance, the chronological order of projects within user repositories, which was analysed by Zeevaarders and Aivaloglou [2021], can be considered in relation to gender and block type or programming concept usage. Finally, the project favorite information that was scraped in this study can be studied in relation to gender and block type or programming concept usage as well.

7.2. Limitations of the study

The age and gender information was obtained by machine learning models that obtained an F1-score of around 0.80 on the training data. Given that similar performance levels hold for the identification of the unlabelled part of the dataset, this means that the age and gender were incorrectly predicted for around 20% of the users. This forms a limitation to the analysis results and should be considered when interpreting these results. Furthermore, even though the age and gender information that was used in the training data was extracted from user texts, it does not guarantee that the users were truthful when disclosing their age or gender. However, the fact that this information was disclosed both voluntarily and publicly does make its truthfulness more likely than when this information was disclosed mandatorily and privately. Lastly, age information was identified and labelled based on textual information provided in the user profile texts. However, we do not know when these profile texts were last updated. This means that we were unable to determine the current age of a Scratch user (e.g. approximate a date of birth) as we could only identify the age at the time the profile text was written.

Another limitation to note is that, for our age analysis, we analysed only around 400,000 (0.41%) out of the total of 97.7 million projects that have been shared on the Scratch platform since their launch[1]. For gender, we analysed around 3.4 million (3.5%) projects. This means that, while the samples can be considered large compared to some other works, they

[1] https://scratch.mit.edu/statistics/

are still a small sample of the entire population. Furthermore, the scraping method used by Zeevaarders and Aivaloglou [2021] may have collected only a local sample of users, as their dataset was constructed by recursively scraping users and their followers. This means that all users in the dataset are part of a certain network. Another limitation involves the absence of recent users and projects, as the scraping effort by Zeevaarders and Aivaloglou [2021] was stopped at the end of October 2019. Therefore, the dataset does not contain the Covid-19 pandemic induced boom of activity. Finally, we only have access to shared projects. This means that unshared projects are not part of the dataset and that our analyses do not provide a complete picture of the programming behaviour of the analysed users. These limitations should be considered when evaluating the generalisabilty of our results.

7.3. Ethics regarding gender

The use of gender in our analyses obliged us to consider the ethical issues that concern it, as we transform a complex human characteristic into a binary variable. Therefore, we aimed to adhere to the ethical frameworks and guidelines proposed in the work by Larson [2017] as much as possible. An important guideline is to avoid using gender unless necessary. The necessity of using gender in our study is based on one of the research objectives: to deepen the understanding of gender differences which have been observed across all STEM disciplines. The use of gender as a variable is in line with other works in this field of research that have similar research objectives [Fields et al., 2013, 2017; Funke and Geldreich, 2017; Gan et al., 2018; Graßl et al., 2021; Hermans and Aivaloglou, 2017; Seiter and Foreman, 2013; Wohl et al., 2015].

Further guidelines involve making explicit the gender theory and category assignment used. We viewed gender as an identity that can be freely expressed by a person. We did not check or enquire whether the gender identity matches one's biological gender. The gender labels in our training set were acquired using self-identified gender in profile texts. The gender disclosures posted in these profile texts were voluntarily disclosed, publicly available, and free-form. These are important aspects as they promote truthfulness and respect a person's autonomy, which is also one of the guidelines. Common gender words in the English language were used to map these disclosures to a binary notion of gender. This mapping was made explicit in the form of regular expressions, which supports the transparency and accountability of our study.

As for the gender labels that were predicted using our Node2Vec model, we tried to promote transparency and accountability by providing an elaborate description of the model and disclosing all parameters used. With our text-based machine learning models, we were able to further embrace transparency and accountability by providing prediction explanations using SHAP. However, our network-based Node2Vec model did not make use of textual information and therefore largely remained a black box. An ethical issue that we could not resolve is that we do not have explicit consent from the analysed users to take part in the study. This is, however, an ethical problem that all studies that use large collections of public data face. We justify the ethical issues we could not resolve by calling on the principles of beneficence and justice discussed in the Belmont Report [for the Protection of Human Subjects of Biomedical and Research, 1978], as a better understanding of the relationship between gender and programming behaviour equally benefits all those involved in computer science in the form of more appropriate educational tools and practices.

8

Conclusion

In this thesis, we presented an automated way of eliciting age and gender information of Scratch users on a large scale using machine learning models. We then demonstrated how this information can be used by performing a quantitative study on block type and programming concept usage in relation to age and gender. First, additional user data such as posted texts and social network information were scraped using the Scratch API. From these texts, we extracted those users which disclose their own age and gender in order to form a training dataset for our machine learning models. The performance of a selection of models was validated on the training data, which resulted in a network-based Node2Vec model to be selected for gender identification, and a text-based Transformer model with selective classification to be selected for age identification. These models were used to automatically elicit age and gender information for the rest of the dataset. This newly acquired data allowed us to quantitatively analyse block type and programming concept usage in relation to age and gender. The answers to our three research questions are summarised in the following paragraphs.

RQ1: What age and gender information can we identify of Scratch users from the publicly available information on the Scratch platform using machine learning models? Our Node2Vec model was able to automatically elicit gender information for 336.394 Scratch users, which is 82.64% of all users in the utilised dataset. 116.850 (34.74%) of these users were identified as female, and 219.544 (65.26%) were identified as male. This gender distribution is in line the entire Scratch population as reported in other works. This is especially remarkable as the training data on which the Node2Vec model was trained had an unbalanced gender distribution (55.58% female and 44.42% male). Our fine-tuned roBERTa model with selective classification was able to automatically elicit age information for 14.993 Scratch users, which is 3.68% of all users in the utilised dataset. This was less than half of what was expected based on the results of the training data. Furthermore, the age distribution was skewed towards the '14+' age group, while the training data had an age distribution that was similar to the entire Scratch population.

RQ2: Which age- and gender-related differences can be detected in the block types Scratch users use in their projects? For each user with identified or labelled age or gender information, we measured how many of their projects contained at least one of each of the

defined block types. Moderate gender-related differences were found using statistical tests in the usage rate of Data, Sensing, and Operators blocks, with male users utilizing them in a larger percentage of their projects. Small gender-related differences were found in the usage rate of Motion, Procedure, Control, and Extensions block types, all with male users using them in a larger percentage of their projects. The usage rate of Other, Sound, Looks, and Events blocks turned out to be similar between male and female users. For age, only a small statistically significant difference was found in the usage rate of Other type blocks. The mean usage rates showed that the higher the age group, the higher the usage rate of Other type blocks. For all other block types, we observed similar usage rates between the different age groups.

RQ3: How frequently do Scratch users of different gender and age groups use the conditionals, coordination, iteration, and variables programming concepts in their projects? We measured the usage rates of certain blocks that indicate the use of the conditionals, coordination, iteration, and variables programming concepts similarly to RQ2. We found moderate gender-related differences in the use of conditionals and variables, and small gender-related differences in the use of coordination and iteration concepts. For all of the observed differences, the mean usage rate of male users was higher. For both genders, iteration blocks were most common as they were used at least once in around 67% of the projects made by female users, and around 80% of the projects made by male users. On the other hand, blocks indicating the use of the variables programming concept were only used at least once in around 19% of the projects made by female users, and in around 41% of the projects made by male users. There were no age-related differences found in the use of programming concepts. For all age groups, we observed a usage rate of around 70% for iteration blocks, around 37% for conditional and coordination blocks, and around 25% for variables blocks.

BIBLIOGRAPHY

Efthimia Aivaloglou and Felienne Hermans. How kids code and how we know. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, August 2016. doi: 10.1145/2960310.2960325. 1, 2, 9, 22

Adrian Benton, Raman Arora, and Mark Dredze. Learning multiview embeddings of twitter users. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/p16-2003. 17

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003. 6

Jacob Cohen. *Statistical power analysis for the behavioral sciences*. L. Erlbaum Associates, Hillsdale, N.J, 1988. ISBN 9780805802832. 21

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. doi: 10.1023/a:1022627411411. 17

Saman Daneshvar and Diana Inkpen. Gender identification in twitter using n-grams and lsa. In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*, 2018. 10

CSIRO's Data61. Stellargraph machine learning library. https://github.com/stellargraph/stellargraph, 2018. xvi

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423. 10, 17

Deborah A. Fields, Yasmin B. Kafai, and Michael T. Giang. Understanding collaborative practices in the scratch online community: Patterns of participation among youth designers. *Computer-Supported Collaborative Learning Conference, CSCL*, 1:200–207, January 2013. 6, 8, 9, 41

Deborah A. Fields, Yasmin B. Kafai, and Michael T. Giang. Youth computational participation in the wild. *ACM Transactions on Computing Education*, 17(3):1–22, August 2017. doi: 10.1145/3123815. 1, 2, 6, 8, 9, 22, 29, 40, 41

United States. National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. *The Belmont report: ethical principles and guidelines for the protection of human subjects of research*, volume 2. Department of Health, Education, and Welfare, National Commission for the . . . , 1978. 41

Alexandra Funke and Katharina Geldreich. Gender differences in scratch programs of primary school children. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM, November 2017. doi: 10.1145/3137065.3137067. 1, 5, 6, 10, 40, 41

Emilia F. Gan, Benjamin Mako Hill, and Sayamindu Dasgupta. Gender, feedback, and learners' decisions to share their creative computing projects. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–23, November 2018. doi: 10.1145/3274323. 7, 8, 9, 41

Isabella Graßl, Katharina Geldreich, and Gordon Fraser. Data-driven analysis of gender differences and similarities in scratch programs. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '21, August 2021. doi: 10.1145/3481312.3481345. 1, 6, 10, 22, 40, 41

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD : proceedings. International Conference on Knowledge Discovery & Data Mining*, volume 2016, pages 855–864, July 2016. doi: 10.1145/2939672.2939754. 19, xvi

Felienne Hermans and Efthimia Aivaloglou. Teaching software engineering principles to k-12 students: A MOOC on scratch. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE, May 2017. doi: 10.1109/icse-seet.2017.13. 1, 7, 10, 22, 39, 41

M. Kosinski, D. Stillwell, and T. Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110 (15):5802–5805, March 2013. doi: 10.1073/pnas.1218772110. 12

Brian Larson. Gender as a variable in natural-language processing: Ethical considerations. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 1–11, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-1601. URL https://aclanthology.org/W17-1601. 41

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China, 22–24 June 2014. PMLR. 17

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692. 17

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. 11, 20

Michael Frederick McTear, Zoraida Callejas, and David Griol. *The conversational interface*, volume 6. Springer, 2016. 16

Jesús Moreno-León and Gregorio Robles. Dr. scratch: A web tool to automatically evaluate scratch projects. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, page 132–133, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337533. doi: 10.1145/2818314.2818338. 9

B. Muthén and L. K. Muthén. Integrating person-centered and variable-centered analyses: growth mixture modeling with latent trajectory classes. *Alcoholism, clinical and experimental research*, 24:882–891, June 2000. ISSN 0145-6008. 8

Shimei Pan and Tao Ding. Social media-based user embedding: A literature review. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, June 2019. 11, 12

Francisco Rangel, Paolo Rosso, Manuel Montes-y Gómez, Martin Potthast, and Benno Stein. Overview of the 6th author profiling task at pan 2018: multimodal gender identification in twitter. *Working Notes Papers of the CLEF*, pages 1–38, 2018. 10, 17

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778. 11

Miguel Angel Rubio, Rocio Romero-Zaliz, Carolina Mañoso, and Angel P. de Madrid. Closing the gender gap in an introductory programming course. *Computers & Education*, 82: 409–420, March 2015. doi: 10.1016/j.compedu.2014.12.003. 5

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513–523, 1988. ISSN 0306-4573. doi: https://doi.org/10.1016/0306-4573(88)90021-0. URL https://www.sciencedirect.com/science/article/pii/0306457388900210. 16

H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLoS ONE*, 8(9):e73791, September 2013. doi: 10.1371/journal.pone.0073791. 17

Linda Seiter and Brendan Foreman. Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research - ICER '13*. ACM Press, 2013. doi: 10.1145/2493394.2493403. 7, 41

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, San Diego, CA, USA, Conference Track Proceedings*, ICLR, May 2015. 12, 18

Claudio Stamile, Aldo Marzullo, and Enrico Deusebio. *Graph Machine Learning*. Packt Publishing, June 2021. ISBN 1800204493. URL https://www.ebook.de/de/product/41926160/claudio_stamile_aldo_marzullo_enrico_deusebio_graph_machine_learning.html. 19

Gail M. Sullivan and Richard Feinn. Using effect size—or why the p value is not enough. *Journal of Graduate Medical Education*, 4(3):279–282, September 2012. doi: 10.4300/jgme-d-12-00156.1. 21

Takumi Takahashi, Takuji Tahara, Koki Nagatani, Yasuhide Miura, Tomoki Taniguchi, and Tomoko Ohkuma. Text and image synergy with feature cross technique for gender identification. *Working Notes Papers of the CLEF*, 2018. 10

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964. 17

Ming-Te Wang and Jessica L. Degol. Gender gap in science, technology, engineering, and mathematics (STEM): Current knowledge, implications for practice, policy, and future directions. *Educational Psychology Review*, 29(1):119–140, January 2016. doi: 10.1007/s10648-015-9355-x. 5

Benjamin Wohl, Barry Porter, and Sarah Clinch. Teaching computer science to 5-7 year-olds. In *Proceedings of the Workshop in Primary and Secondary Computing Education*. ACM, November 2015. doi: 10.1145/2818314.2818340. 1, 7, 10, 41

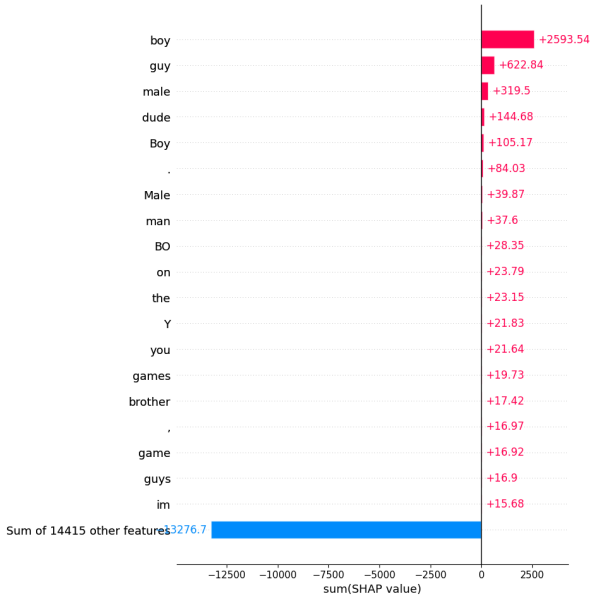
Xiaodong Wu, Weizhe Lin, Zhilin Wang, and Elena Rastorgueva. Author2vec: A framework for generating user embedding. *CoRR*, abs/2003.11627, 2020. URL https://arxiv.org/abs/2003.11627. 12

Ad Zeevaarders and Efthimia Aivaloglou. Exploring the programming concepts practiced by scratch users: an analysis of project repositories. In *2021 IEEE Global Engineering Education Conference (EDUCON)*, pages 1287–1295, 2021. doi: 10.1109/EDUCON46332.2021.9453973. 1, 2, 9, 13, 22, 31, 39, 40, 41

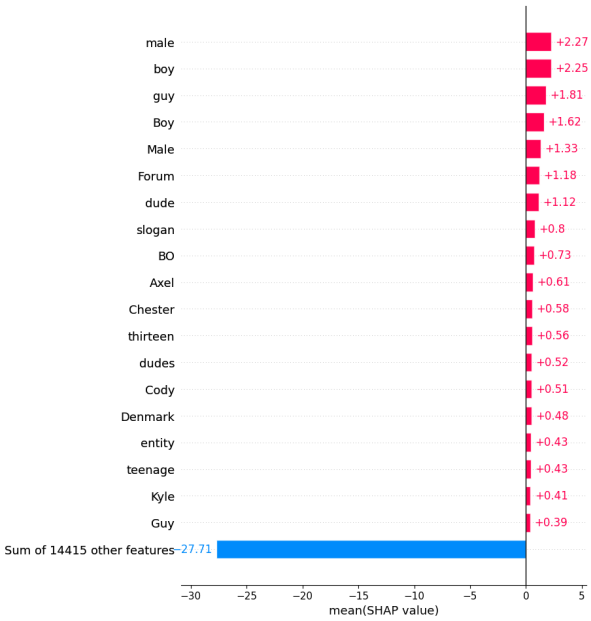
Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. User profile preserving social network embedding. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3378–3384. AAAI Press, 2017. ISBN 9780999241103. 12

APPENDIX A

SHAPLEY VALUES TRANSFORMER ROBERTA MODEL

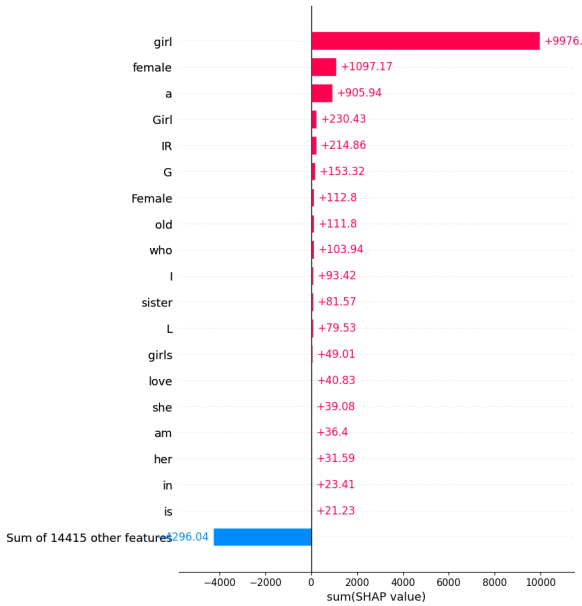


(a) Sum

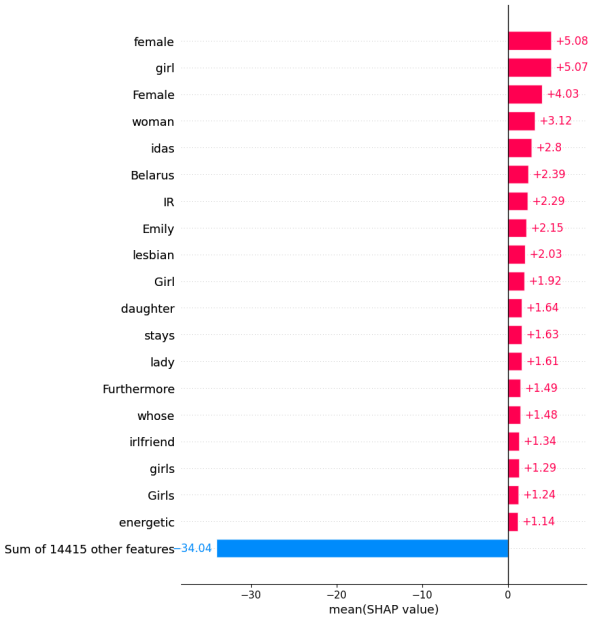


(b) Mean

Figure 1: roBERTa Shapley values (Gender: male)

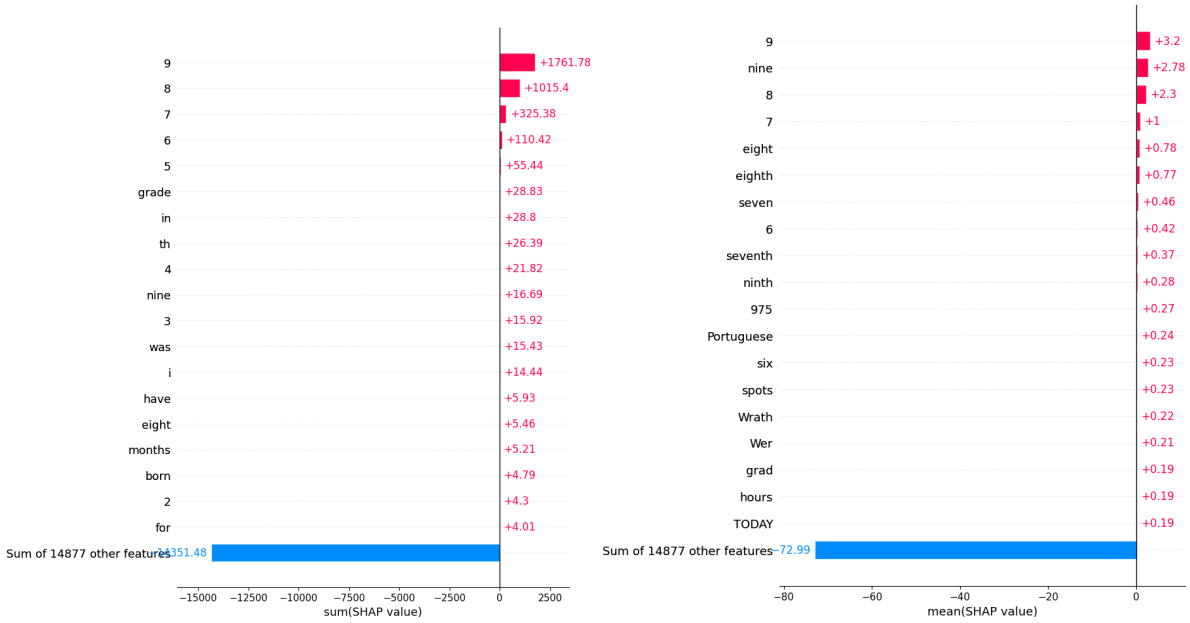


(a) Sum



(b) Mean

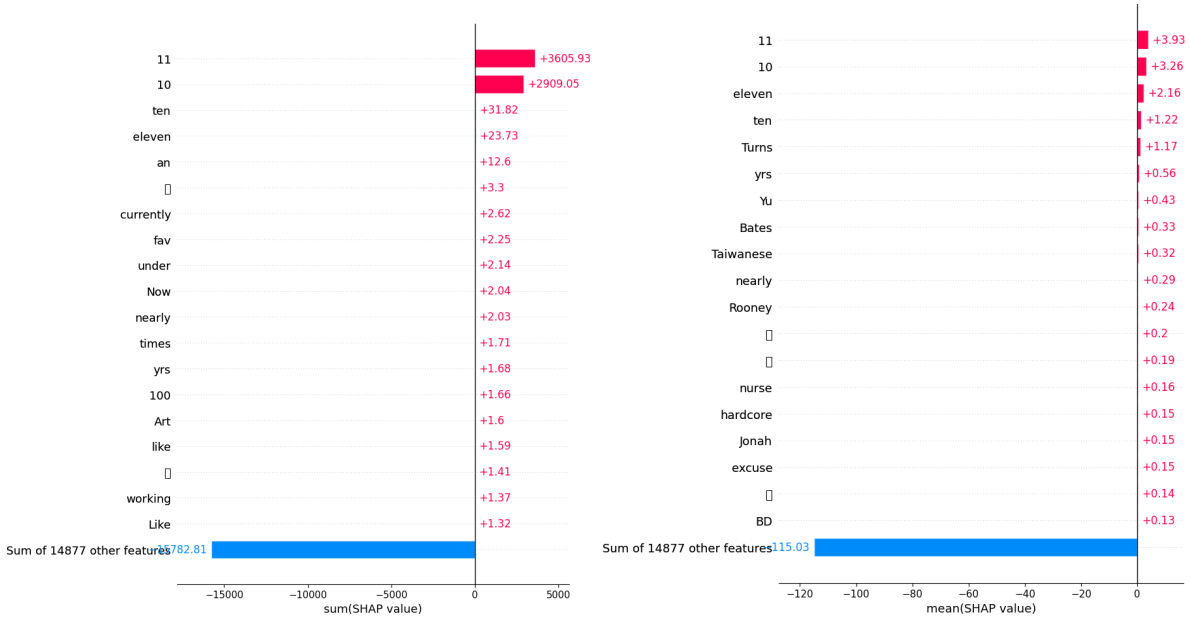
Figure 2: roBERTa Shapley values (Gender: female)



(a) Sum

(b) Mean

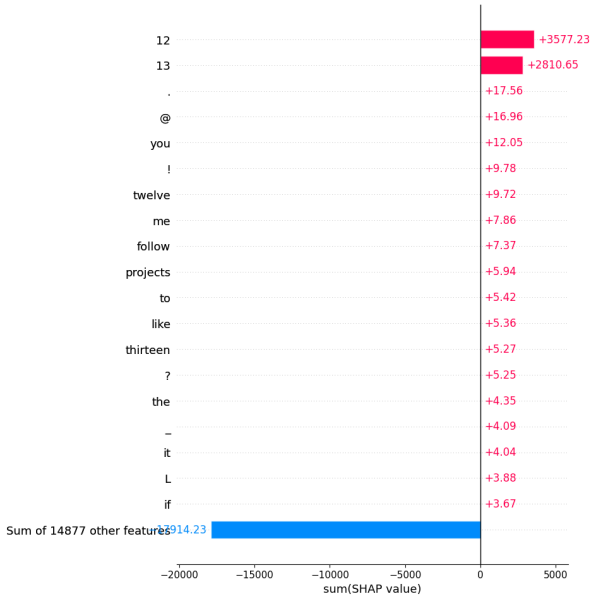
Figure 3: roBERTa Shapley values (Age: 4-9)



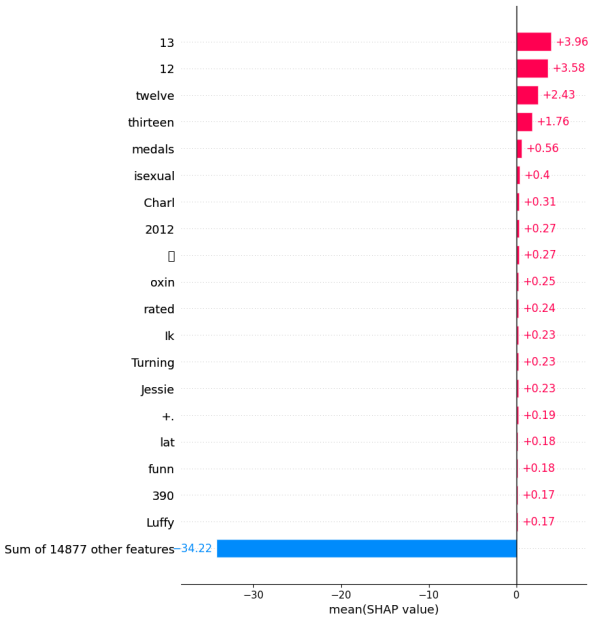
(a) Sum

(b) Mean

Figure 4: roBERTa Shapley values (Age: 10-11)

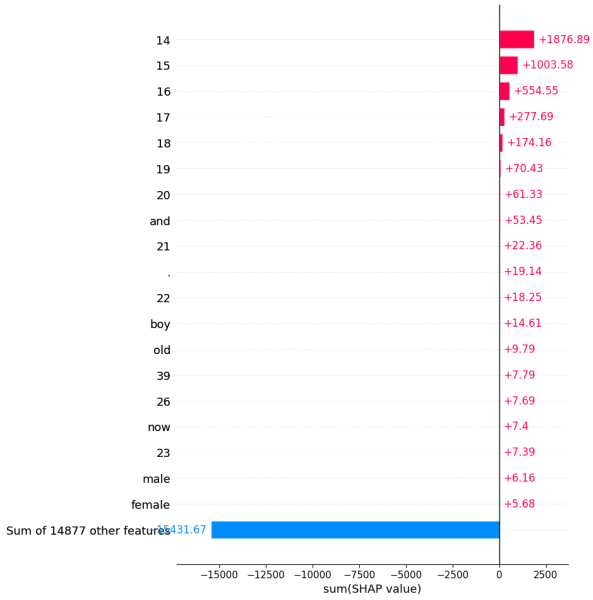


(a) Sum

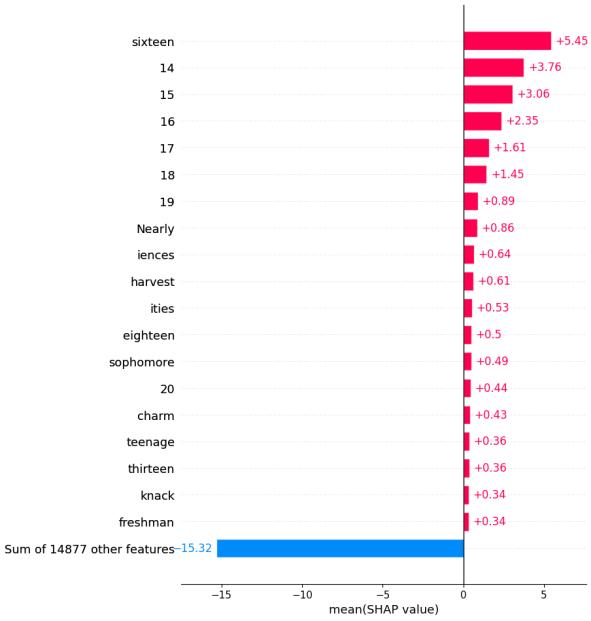


(b) Mean

Figure 5: roBERTa Shapley values (Age: 12-13)



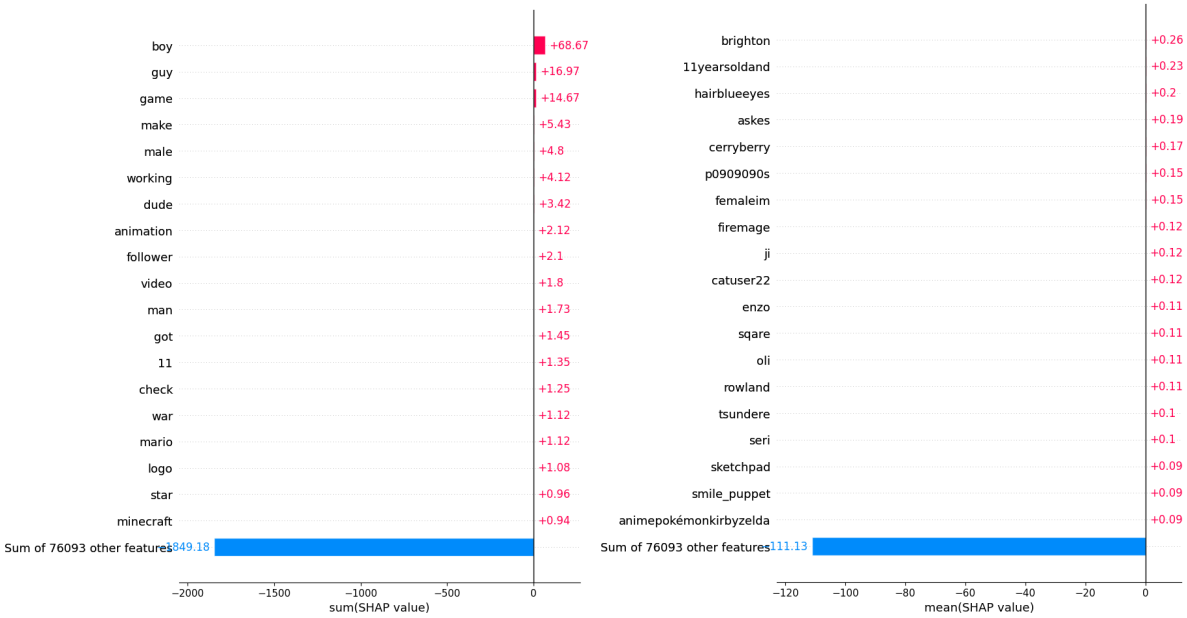
(a) Sum



(b) Mean

Figure 6: roBERTa Shapley values (Age: 14+)

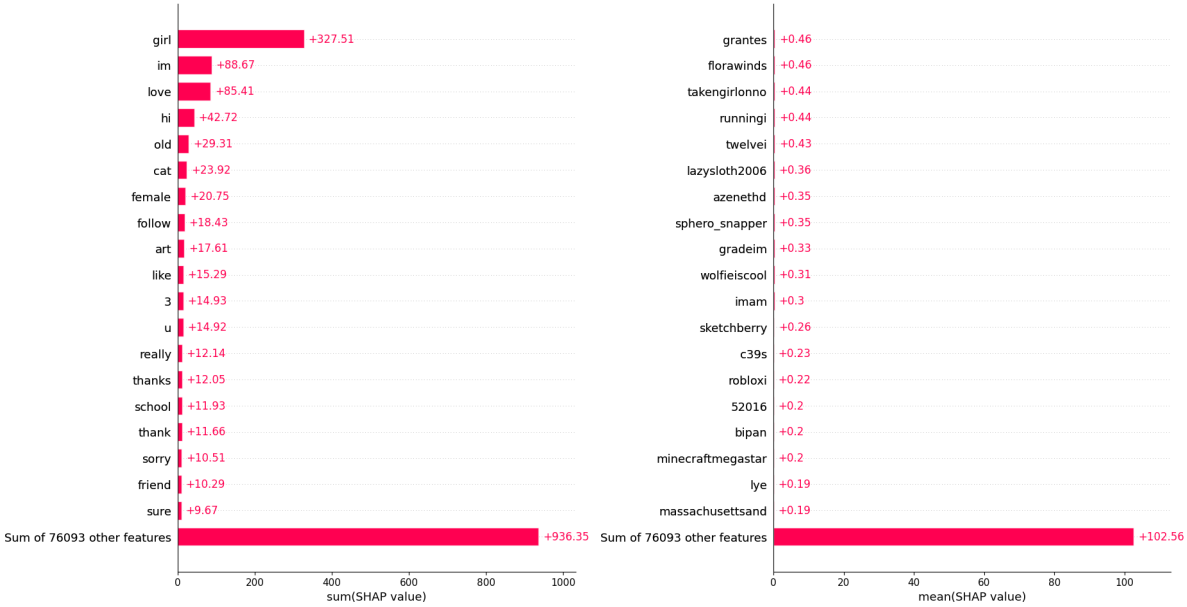
SHAPLEY VALUES DOC2VEC MODEL



(a) Sum

(b) Mean

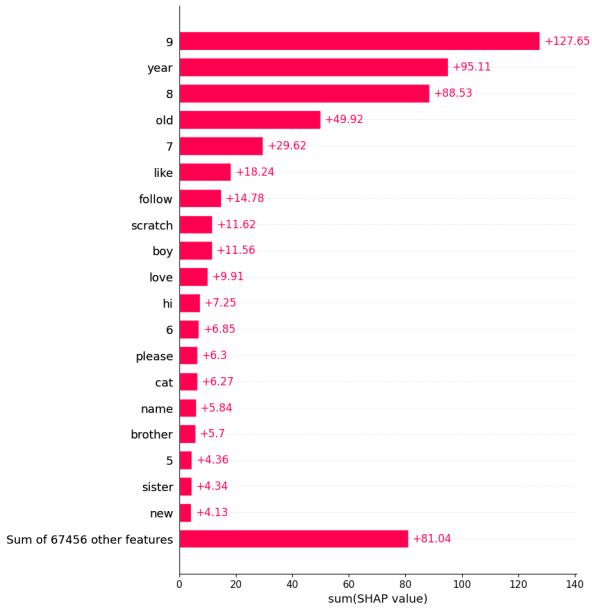
Figure 7: Doc2Vec Shapley values (Gender: male)



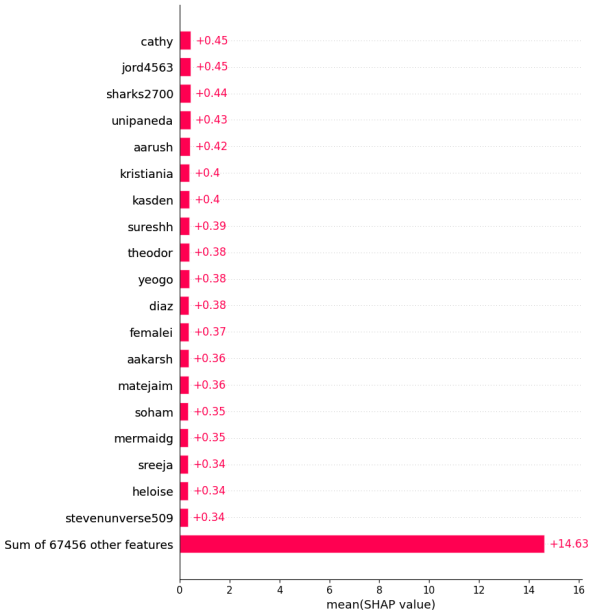
(a) Sum

(b) Mean

Figure 8: Doc2Vec Shapley values (Gender: female)

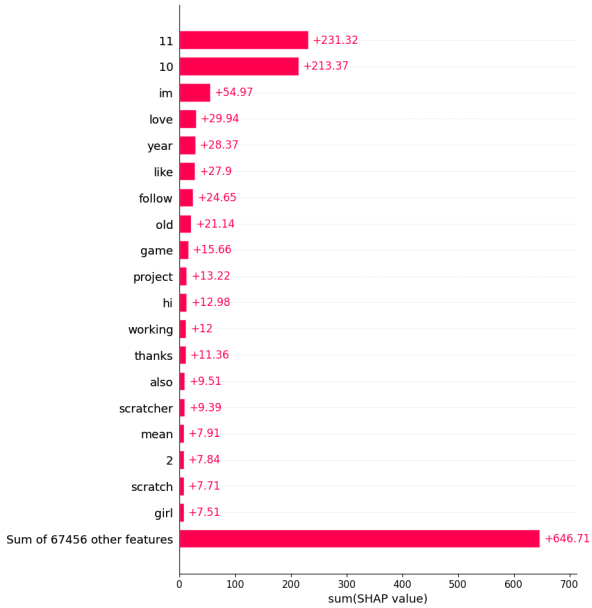


(a) Sum

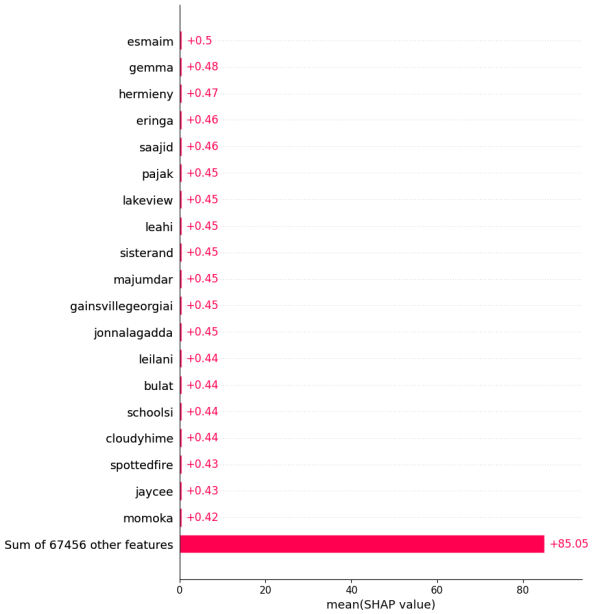


(b) Mean

Figure 9: Doc2Vec Shapley values (Age: 4-9)

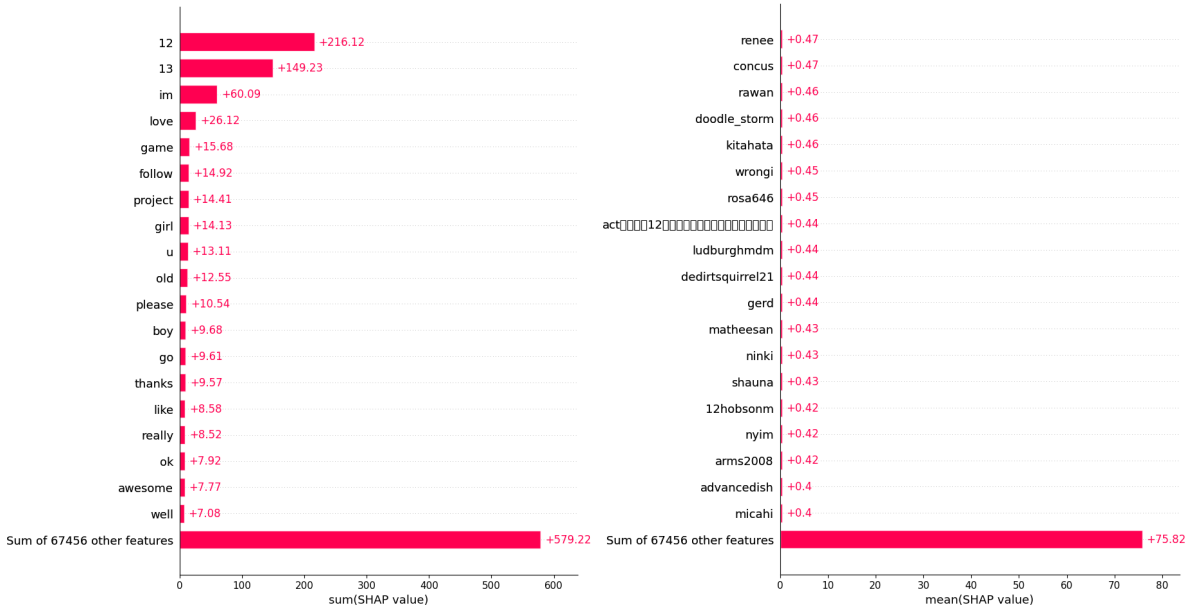


(a) Sum



(b) Mean

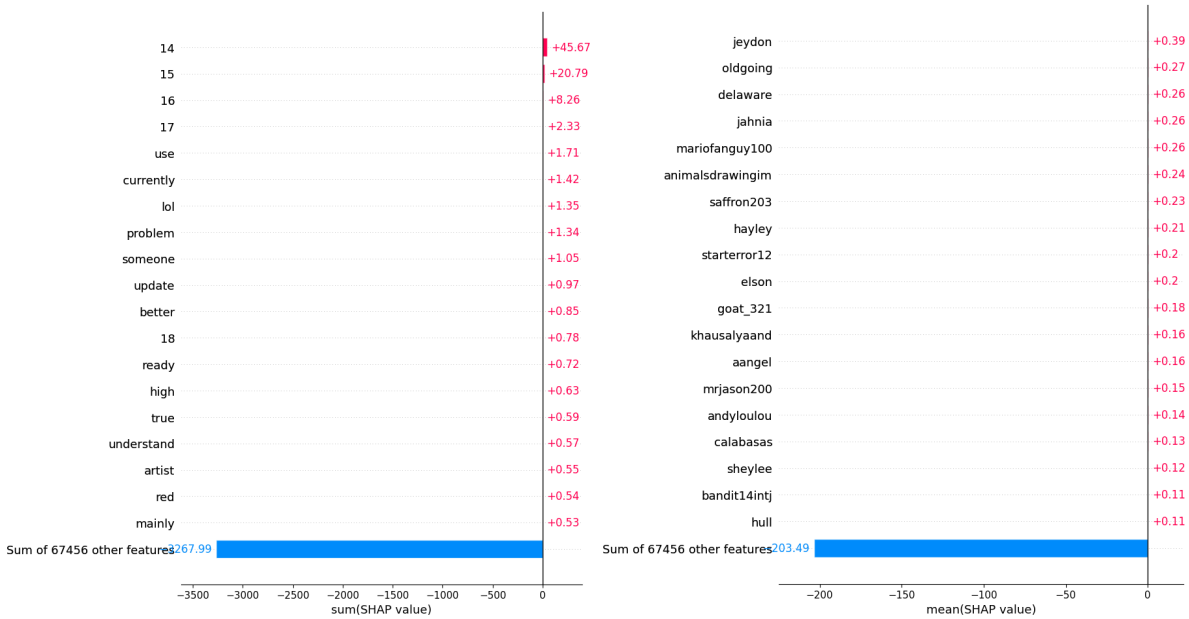
Figure 10: Doc2Vec Shapley values (Age: 10-11)



(a) Sum

(b) Mean

Figure 11: Doc2Vec Shapley values (Age: 12-13)

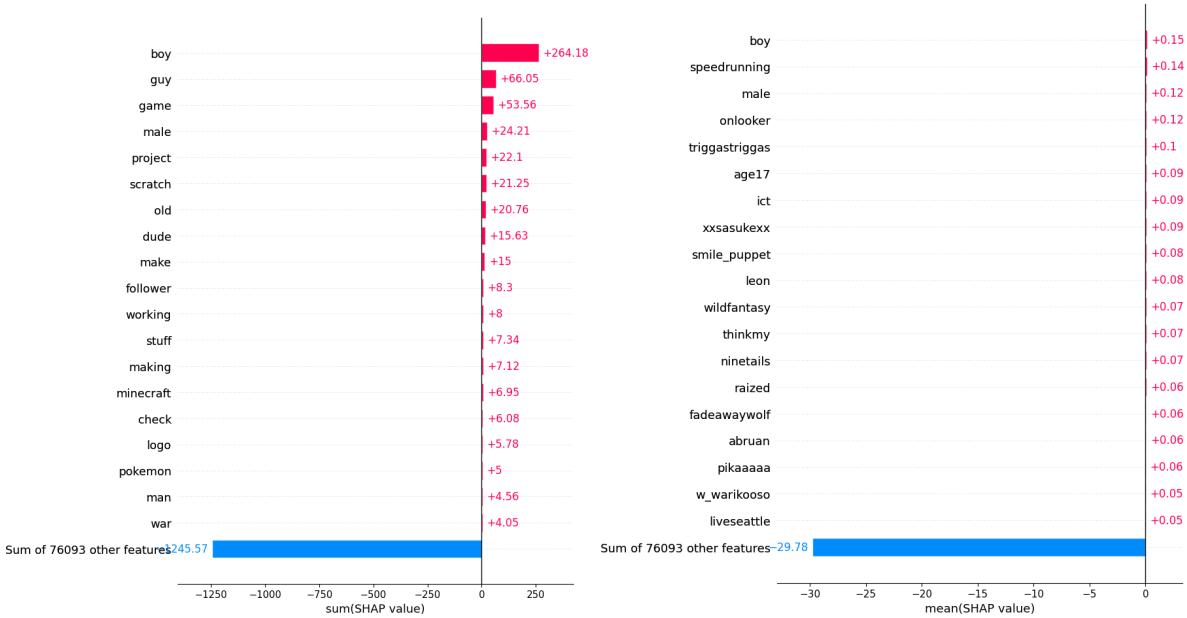


(a) Sum

(b) Mean

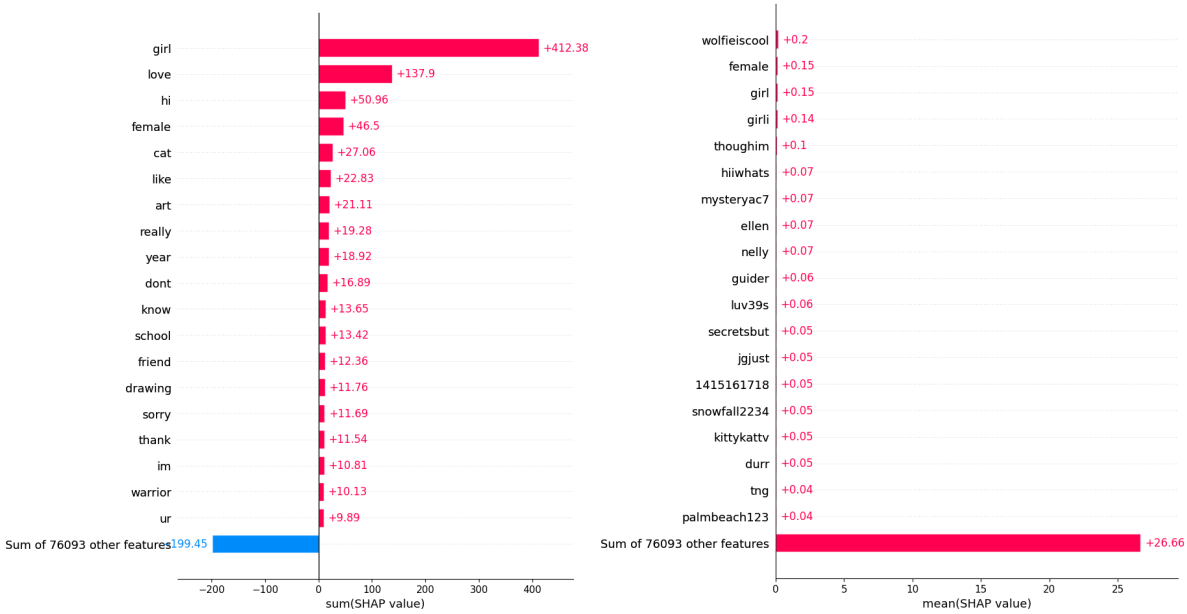
Figure 12: Doc2Vec Shapley values (Age: 14+)

SHAPLEY VALUES BAG-OF-WORDS MODEL



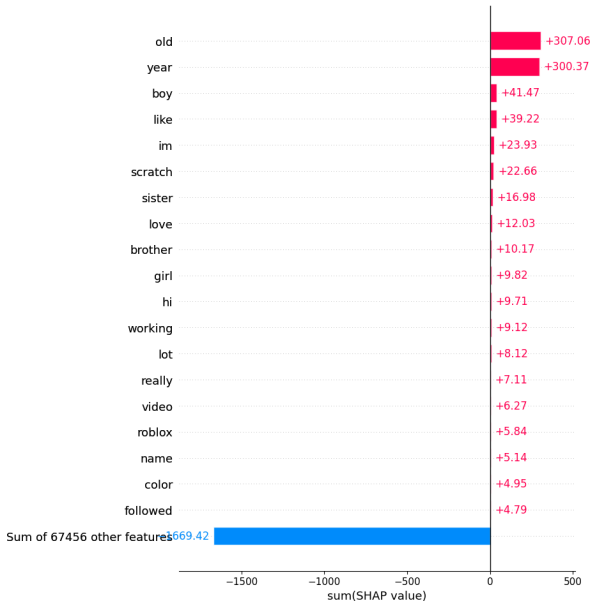
(a) Sum (b) Mean

Figure 13: Bag-of-words Shapley values (Gender: male)

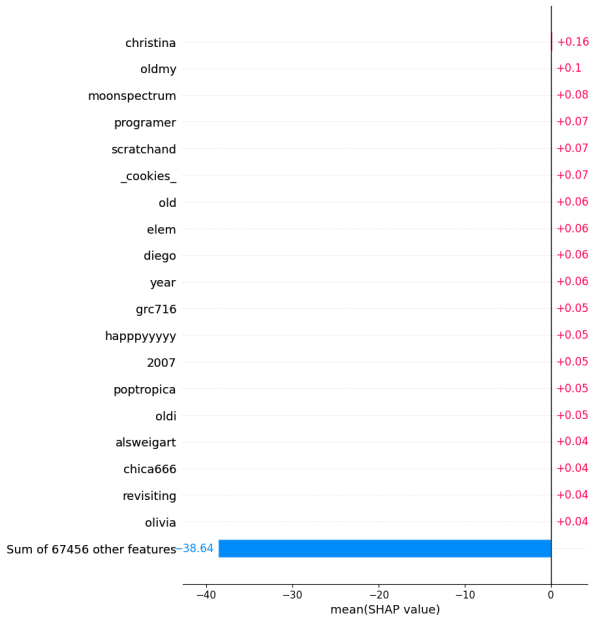


(a) Sum (b) Mean

Figure 14: Bag-of-words Shapley values (Gender: female)

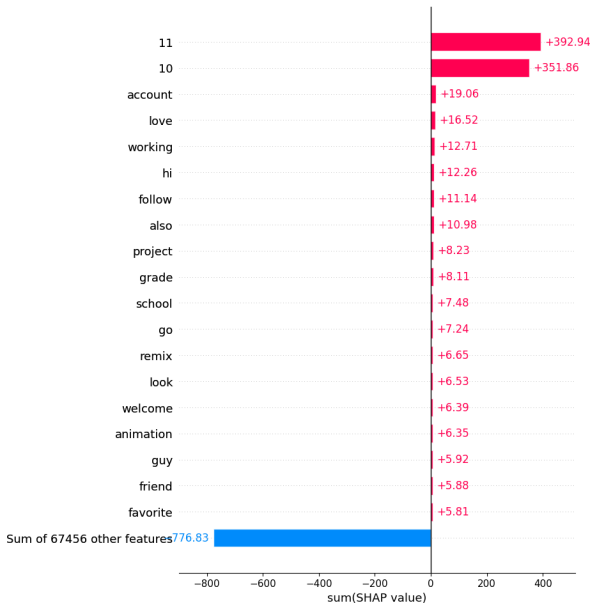


(a) Sum

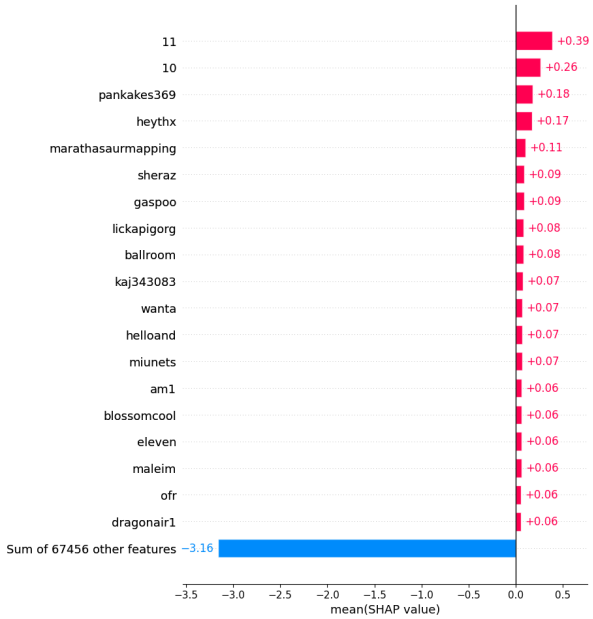


(b) Mean

Figure 15: Bag-of-words Shapley values (Age: 4-9)

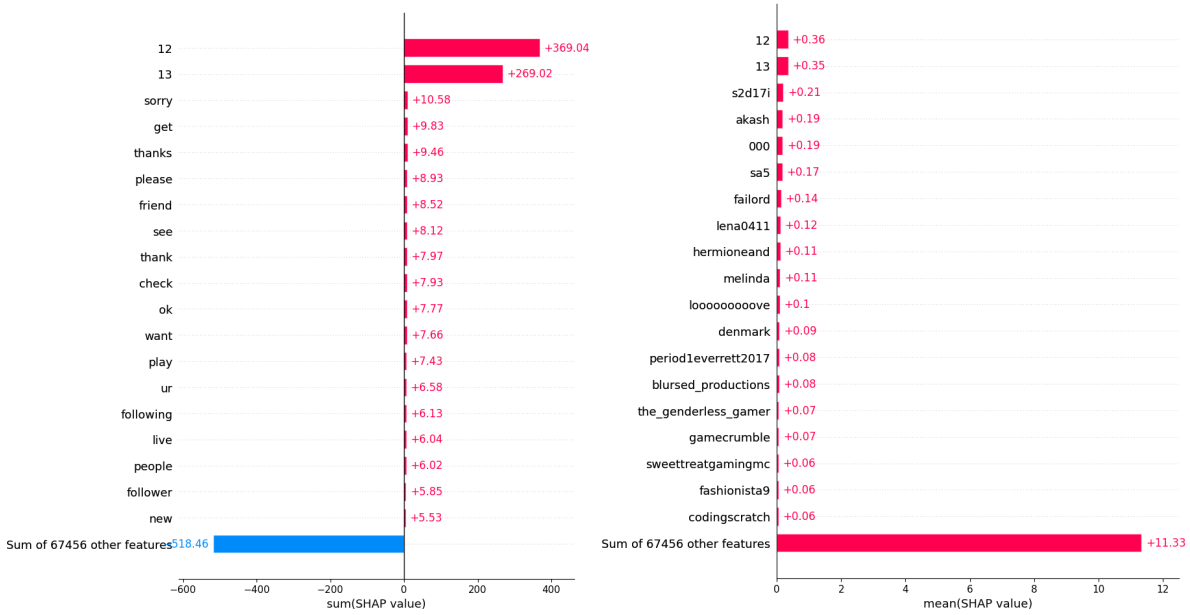


(a) Sum



(b) Mean

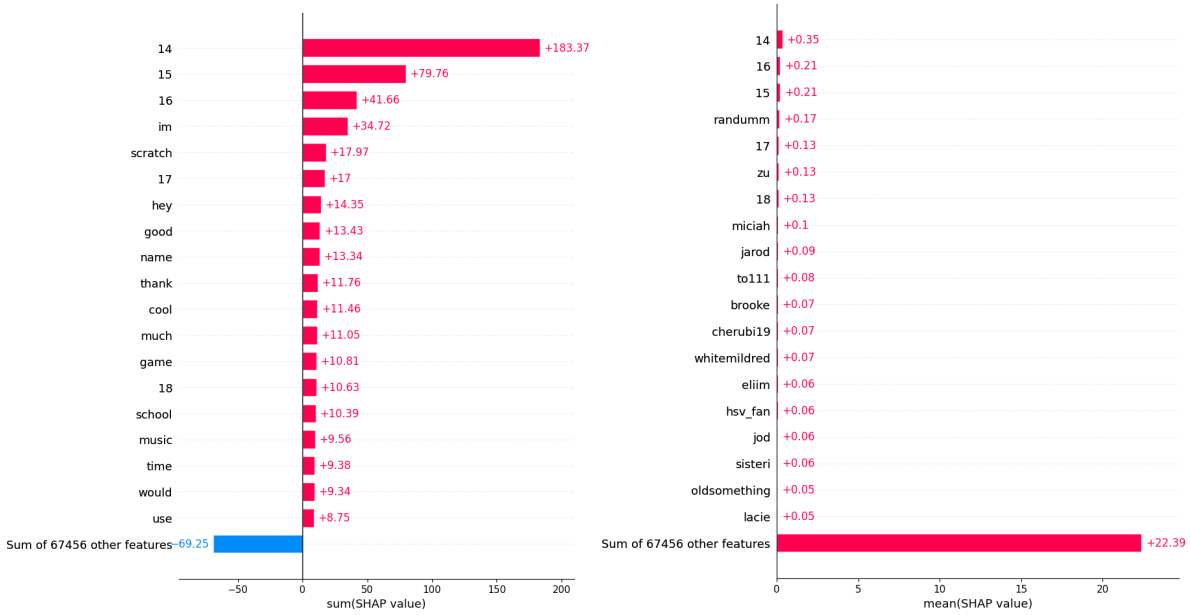
Figure 16: Bag-of-words Shapley values (Age: 10-11)



(a) Sum

(b) Mean

Figure 17: Bag-of-words Shapley values (Age: 12-13)



(a) Sum

(b) Mean

Figure 18: Bag-of-words Shapley values (Age: 14+)

APPENDIX B

Parameter	Value
Classifier	SVM
Optimization	Stochastic gradient descent (SGD)
Loss function	Modified huber

Table 1: Bag-of-words model parameters. Please refer to the default parameter values stated in Sklearn's CountVectorizer[1], Sklearn's TfidfTransformer[2] or Sklearn's SGDClassifier[3] documentations for any parameters not included in this table.

Parameter	Value
Classifier	SVM
Training algorithm	Distributed bag-of-words (DBOW)
Vector size	400
Training epochs	100

Table 2: Doc2Vec model parameters. Please refer to the default parameter values stated in Sklearn's SVC[4], Gensim's Doc2Vec[5] or Gensim's Word2Vec[6] documentations for any parameters not included in this table.

Parameter	Value
Pre-trained model	RoBERTa (base)[7]
Training batch size	16
Evaluation batch size	16
Maximum sequence length	128
Training epochs	1

Table 3: Transformer model parameters. Please refer to the default parameter values stated in SimpleTransformers' model configuration[8] documentation for any parameters not included in this table.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[5] https://radimrehurek.com/gensim/models/doc2vec.html#gensim.models.doc2vec.Doc2Vec
[6] https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec
[7] https://huggingface.co/roberta-base
[8] https://simpletransformers.ai/docs/usage/#configuring-a-simple-transformers-model
[9] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[10] https://keras.io/api/applications/vgg/#vgg16-function

Parameter	Value
Classifier	SVM
Image dimensions	224x224

Table 4: VVGNet16 model parameters. Please refer to the default parameter values stated in Sklearn's SVC[9] and Keras' VGG16[10] documentations for any parameters not included in this table.

Parameter	Value
Classifier	SVM
Vector size	128
Window	10
Minimal total node frequency	0
Training algorithm	Skip-gram
Training epochs	1
Random walk length	80
Walks per node	10
Return parameter p	1
In-out parameter q	0.5[11]

Table 5: Node2Vec model parameters. Please refer to the default parameter values stated in Sklearn's SVC[12], StellarGraph's [Data61, 2018] StellarDiGraph[13] and BiasedRandomWalk[14], and Gensim's Word2Vec[15] documentations for any parameters not included in this table.

[11] Promotes homophily similarity [Grover and Leskovec, 2016]

[12] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[13] https://stellargraph.readthedocs.io/en/v1.0.0rc1/api.html#stellargraph.StellarDiGraph

[14] https://stellargraph.readthedocs.io/en/v1.0.0rc1/api.html#stellargraph.data.BiasedRandomWalk

[15] https://radimrehurek.com/gensim/models/word2vec.html#gensim.models.word2vec.Word2Vec