Magic Pen: Automatic Pen-Mode Switching for Document Annotation

by

Kevin A. Desousa

A thesis submitted to the
School of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Faculty of Science
University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada
August, 2022

Thesis Examination Information

Submitted by: Kevin A. Desousa

Master of Science in Computer Science

Thesis Title: Magic Pen: Automatic Pen-Mode Switching for Document Annotation

An oral defense of this thesis took place on August 9, 2022 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee: Dr. Patrick Hung

Research Supervisor: Dr. Christopher Collins

Examining Committee Member: Dr. Jeremy Bradbury

Thesis Examiner: Dr. Miguel Vargas Martin

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Traditional digital pen interfaces use menu buttons to change the pen mode, which results in time and cognitive load spent on the round-trip and potential mode errors from tapping commonly small mode selection buttons. I present the Magic Pen, a technique that automatically switches between digital pen modes (draw, highlight, and underline). The Magic Pen system is driven by a Long Short-Term Memory (LSTM) model trained on pen data collected from two studies and uses Transfer Learning (TL) to tune itself towards how a user specifically annotates iteratively. To build a pen-mode classifier, I have created a dataset from 27 participants carrying out various annotation tasks and collected various features of how a digital pen is held and used on screen. My approach offers a unique alternative to standard mode-switching methods by allowing users to mark up a document without explicitly changing the mode. This system allowed them to stay focused while working. If the Magic Pen chooses the incorrect mode, mitigation techniques incorporate a flick gesture or a tap on the screen to quickly correct or remove a stroke. A web-based annotation environment was developed for Magic Pen that allows for rapid prototyping of annotation systems with support for user-supplied PDF documents. I originally evaluated an earlier iteration of Magic Pen in a mixed-methods study with 18 participants and further evaluated an improved approach with 5 participants. Magic Pen was found to be equally preferred overall compared to a more conventional menu approach, and the use of TL allowed for greater model predictability and stability.

Keywords: digital pen interfaces; mode switching; machine learning; transfer learning; error mitigation

Author's Declaration

I, Kevin A. Desousa, hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

The research work in this thesis that was performed in compliance with the regulations of Research Ethics Board/Animal Care Committee under REB Certificate number file number #15755.

_____ Kevin A. Desousa

Statement of Contributions

I hereby certify that I am the sole author of this thesis and that no part of this thesis has yet been published or submitted for publication as of August 2022. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Christopher Collins, for giving me this support and guidance throughout my academic hurdles. The door to Dr. Collins' office was always open, with him eager to help in times of need. I would not have gotten as far within the realm of academia if it was not for Dr. Collins' encouragement and many opportunities to explore research, even when I was only a lab coordinator.

Secondly, I would like to thank all my friends at the Visualization for Information Analysis Lab (VIALAB) for their advice, guidance, and help. I especially thank Mariana Akemi Shimabukuro, who, on countless occasions, helped direct me in terms of my research and overall life plans. Her wisdom encouraged me through many tough decisions and is invaluable to me. She, and the rest of my lab peers, help proved to be crucial to the completion of my degree.

I cannot forget to thank my family and friends for their continued support throughout the thesis development process. This accomplishment was made possible with their help.

Finally, I would like to express my profound gratitude toward my fiancée, Emily, for providing me unfailing support and continuous encouragement throughout the years of my work. Despite not being familiar with the field, she continuously and selflessly offered to aid me and my research, even if that meant listening to my practice talks and reading over my thesis a dozen times. Without her, I could not have completed my research.

In addition, the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

Thank you.

Contents

List of Tables

List of Figures

1 Introduction

Within the field of Human-Computer Interaction, there is a long history of developing and designing technologies with the primary goal of offering digital alternatives to traditional tools used in the real world. These digital replacements are introduced for a variety of reasons including removing conventional constraints, while also offering a feature-rich environment to aid users in their desired tasks. An example of such technology is the introduction of the digital stylus (or pen).

Throughout history, some form of pencils and pens have been used for annotating various documents. This history has allowed styluses to become synonymous with annotation, leading them to become one of the most widely used and recognized tools. As a result of their notoriety, there has been a push for styluses to be used in a digital form, so that annotations can be made digitally.

The main advantage for digital styluses is that the learning curve is kept fairly low by default due to their instant recognition amongst people. Another benefit is that their input is direct, especially when compared to a typical computer mouse. This directness allows users to perform tasks which require precision, such as drawing, with ease. Modern digital styluses such as the Apple Pencil and Surface Pen extend the list of features even further, with many offering barrel buttons, pressure sensors, and even accelerometers, which can be used to augment the standard methods of stylus interaction. However, while new devices contain these novel elements, existing software that utilizes the digital stylus as an input has not been updated to take advantage of them fully.

To be a successful replacement to existing methods of annotation, like pen and paper, digital stylus interfaces must offer a variety of features that go above and beyond what could be done traditionally [39]. This increase in usefulness is required due to drawbacks associated with digital pens, such as the fact that they are slow to start due to needing to turn on the device they are associated with. Despite its disadvantages, the digital pen can be a powerful tool. For example, a single stylus can be a pen, pencil, paintbrush, highlighter, or even an eraser, all at the press of a button, allowing for infinite possible modes. However, the introduction of these new features brings about another challenge: the need for a method to switch between the available features (i.e. modes). To overcome this more specific challenge, most stylus interfaces

leverage design elements taken from traditional desktops, such as the toolbar. While the use of the toolbar was initially effective, due to it being synonymous with mode-selecting, studies have shown it to be particularly slow when it comes to switching between digital stylus modes. In addition, the toolbar also required more user movement, causing strain and increasing the user's cognitive load [8].

There is an opportunity to build a system which can suggest or change the user's active stylus mode without requiring them to use a rudimentary menu, which has shown to be inefficient in stylus environments. Ideally, this form of mode switching would be done through an implicit approach, which is a form of interaction that does not require special gestures or behaviours to activate particular functionality; instead, it relies on using natural behaviours to determine appropriate actions [50]. While the purpose of moving a pen is for writing, the information obtained will also be used for mode switching.

Providing an easier method to change modes in an annotation environment has been a challenge in the past because: (1) Users forget mode-switching gestures (or forget to use them at the proper time) in practice; (2) The amount of stylus-modes possible is nearly endless, so a system should be able to support multiple modes (ideally without relying on a menu); (3) If the system produces an incorrect mode-change, it can cause greater stress and cognitive load than the toolbar that it's meant to replace. This work argues that to build an effective stylus mode-changing system, one must be able to learn from the user's annotation styles so that mode-specific cues can be inferred and react accordingly.

1.1 Motivation

Within the field of digital stylus annotation, a commonly-faced challenge is managing the mode problem — or the issue of offering and switching between multiple functions (modes) on a single stylus. While overloading of the stylus function is required to support many modes on a single stylus, research has shown that even minimal constraints can distract a user from their task [28]. Alternative methods for switching a user's active stylus mode have been well investigated in the past, with systems utilizing different factors of digital annotation. For example, some earlier works used on-stylus buttons to switch modes [25, 33], while others used on-screen gestures instead [18, 26, 27, 62]. Although these previous works were compelling, they had two main downfalls: their use of delimiters to distinguish mode-changing actions from regular annotation and that users had to both remember the gestures and use the gestures before annotating.

Creating a system that allows for digital annotation is challenging, especially when allowing for several functions on a single stylus. Managing the mode problem is nuanced, where a

user's actions on-screen can drastically affect the performance and preference of explicit mode-switching techniques, leading most users to default to the tried-and-true toolbar method. However, despite the drawbacks of previous works, there is potential for an implicit approach that can automatically switch the user's active stylus mode, which would remove the need for gestures or other delimiters.

The use of inferencing in the context of detecting expected user behaviour is not a novel concept. In the past, machine-learning systems have been used to detect items such as gestures and speech patterns [20, 54]. These same types of systems could be used to infer the user's intended mode based on their specific annotation style. However, these potential detection benefits have not yet been widely realized within the context of an annotation environment. Prior research has explored the use of a user's prior interactions to determine what function the user intends to use next, but does so without leveraging the features of the digital stylus itself; severely limiting the scope of what modes can be offered.

My goal is to take the concept of automatic influencing and extend it to determine the user's preferred stylus mode based on how they hold and use the pen (see Figure 1.1). We propose mitigation methods that can be used to help a user recover in the event of incorrect mode switches.

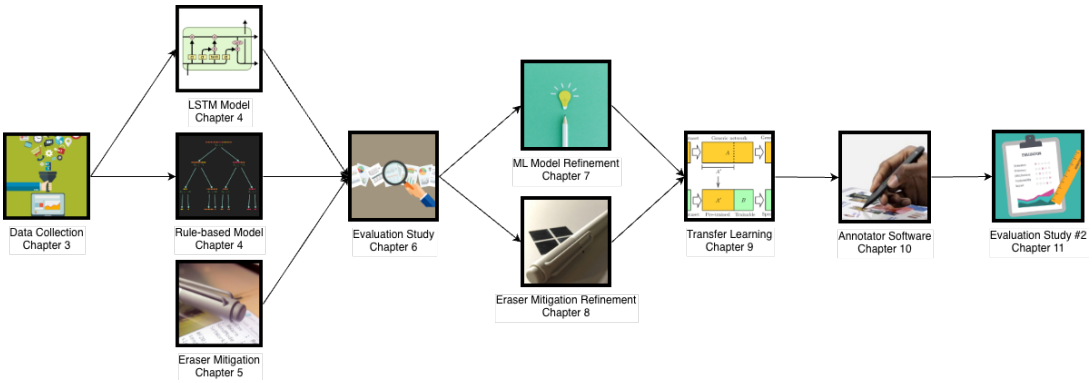


Figure 1.1: A progression diagram detailing the various methods conducted and their process.

1.2 Contributions

After observing a need for a stylus mode-switching system that does not require gestures or menus, I propose a recommendation model, called Magic Pen. This model would detect the user's annotation style, analyze it, and provide a real-time stylus mode switch, which can be used to change modes implicitly. The main contributions of this work are as follows:

- A machine learning model to accurately predict a user's intended stylus mode based on how they are holding and using their stylus. Two implementations are discussed: one using a neural network, and the other using decision trees.
- A annotation environment which combines the standard annotation environment with the stroke-mode switching system to implicitly change stylus modes.
- A transfer-learning system which is designed to learn from the user with as minimal user interaction as possible.
- A system to correct user strokes when they are either switched to, or made in, the wrong stroke mode. Two different techniques are introduced: The first allows for strokes' modes to be changed and the second allows for quick stroke removals.
- Two evaluation studies that explore user reactions to automatic annotation environments and how they handle incorrect stroke modes.
- A series of performance considerations for web-based automatic annotation environments.

2 Background

There is a lot of untapped potential for using input methods beyond a traditional mouse and keyboard to interact with a computer system [30]. The usability and usefulness of pen interfaces are crucial in the appeal of lightweight, hand-held computers, as their small size prohibits the inclusion of typical input methods [14]. This section discusses past approaches to the design of automated or rapid mode-switching techniques for digital pens.

2.1 Studies of Annotations

A considerable amount of previous work investigated ink annotations in order to derive implications for digital interfaces. The most relevant studies on annotations are Marshall's seminal study [28] on annotations of physical textbooks and Ovsiannikov et al.'s [39] the main types of annotations people perform and their uses. The results of these studies indicate that annotations are useful to remember, think, clarify, and share portions of text they found particularly meaningful; a critical activity identified as *active reading*. Ovsiannikov et al. note that, in order for electronic annotations to be more appealing than their paper counterparts, they must be superior in the capabilities they offer, such as the ability to provide an infinite number of pen tools. Most common annotations are to mark up portions of text with a marker, followed by highlighting keywords and writing in the margins. Therefore, these these types of annotations are focused in this work.

Marshall's study investigated marks made on physical, university-level textbooks and derived implications for designing digital annotation tools [28]. Their findings noted that a significant amount of the annotations were informal, with the strokes taking many different forms and appearing at seemingly arbitrary locations within the text (e.g. in the margins, in between lines, over text, within figures), calling for the ability to mark up content directly. They also noted the idiosyncratic nature of annotations and advocated for fluid tools to support varied personal practices. Lastly, the inkings made were extremely individual in form. Users added colour and symbols to aid their understanding while being limited to the overhead (i.e. cognitive load) that they felt was worthwhile. This informality and individuality within strokes suggest that for a digital annotation interface to be effective, it must support unconstrained sketches, allowing the

user to markup a document as they see fit. The authors mention that to support active reading to its full extent, interfaces must make an active effort to reduce the amount of overhead or cognitive load they induce on the users; otherwise, they may distract them. My work focuses on providing multiple types of pens to offer rich annotation capabilities to readers (e.g., different strokes for highlighting, underlining or writing notes over content) while limiting disruptions caused by pen-mode switching by relying on automatic methods.

The work done by Ovsiannikov et al. [39] extends the research field of investigating physical annotations that Marshall first began [28] to create guidelines for a better digital annotation system. The authors conducted an empirical study of annotations on paper to better understand what an exemplary annotation system should be. Their results revealed that the most common annotation style was marking portions of text with a marker, highlighting keywords and writing margins. When investigating particular annotations that participants made, ink and highlighter markings were found to help the user find portions of text quickly. Simultaneously, writing in the margins kept thoughts and ideas closely related to the nearest text's content. Overall, the authors note that for pen interfaces to be preferred, they must offer additional features and capabilities. In particular, this finding forms a balancing act with Marshall's work, where new features and capabilities must be introduced while reducing the cognitive load required.

2.2 Pen Interaction Investigations

Within the field of digital pen research, there has been prominent discussion about the types of interactions that should be supported and how natural they would feel on a digital pen. A considerable amount of the research done to find these interaction types has been oriented towards determining if digital stylus interaction differs from its analogue counterpart. In work done by Annet et al. [3], this difference in the interaction was explored by having participants annotate a document with an analogue pen and a digital pen. They found that several factors directly affected a user's experience with a stylus, such as stylus accuracy, latency, and unintended touch. Secondary factors, such as stroke aesthetics and beautification, were also discussed. While significant for using and adopting digital annotation devices, these factors did not affect the traditional pen similarly. Future works continued to investigate these factors to determine the boundaries for each [2, 4, 5, 6, 7]. They conclude that, for pen interfaces to be effective and introduce the least cognitive load, the impact of items like stroke latency must be reduced.

To discover if the digital stylus' affordances affected its impact on users, Riche et al. [44] conducted various diary studies. Their findings determined that, while they are visually similar

and perform similar actions (annotating), digital pens are not equal to standard pens, with varying affordances and activities. Their conclusion also outlined a list of digital pen-specific affordances, which is vital to designing pen interfaces as they allow designers to design their interactions specifically around digital pen capabilities. In addition, the affordances are required to innovate and improve on interaction techniques, which as Ovsiannikov et al. noted [39], is vital for the adoption of digital pens. This list of affordances would then be used in various other works, such as ActiveInk by Romat et al. [45] and SpaceInk by Romat et al. [46], to enable the digital pen to support features that were never thought of before. ActiveInk extends the field of digital annotation by allowing for gestures to interact with data elements, such as filtering a dataset geographically by circling individual states to drawing a line on a chart to remove a subset of data. SpaceInk introduces several techniques to allow for in-content annotations by dynamically reflowing documents. Both works demonstrate how the unique pen affordances discovered could be used to allow for greater interoperation with both text and data.

2.3 The Mode Problem and Optimization Efforts for Pen Interfaces

In sketching software, special consideration must be given towards managing the *mode problem* [56]. This problem mainly stems from the fact that there is an excess of commands available compared to what a digital pen can support at any given moment (called a "mode-error"). An example of a mode error in practice would be attempting to draw on screen when the selection mode is active. Thus, most software relies on some deliberate action to select amongst the overloaded functions of the pen. In the past, actions such as pressing a button [25], double-tapping the stylus [53], and explicit stroke gestures [18, 43] have been used to signify to the computer that a user wishes to change the pen's operation. However, the issue with these actions is that a user must remember to perform them before switching pen modes. If not, users find themselves in a situation where they create meaningless strokes without any indication of what they did wrong [47], possibly causing frustration. To recover from this state, users must divert their attention from their task to remove any unneeded strokes and enter their intended mode to repeat the mistaken action.

As a result of the *mode problem* with current pen interfaces, there have been efforts to optimize the mode-switching behaviour in traditional desktop interfaces, including the use of external actions to invoke a particular mode. For example, DENIM [25] and Flatland [33] rely on pen barrel buttons that users must press or hold to switch action options. Mode switches using these approaches can be permanent or last only while the button is depressed ("quasi modes"). However, as noted by Saund and Lank [47], this approach suffers from two critical

drawbacks. Physically pressing or holding a button while making a gesture can be awkward. It also requires the users to remember to perform this initial action before switching pen modes, which may not reduce cognitive load compared to menus.

Another approach that has been experimented with is the use of features offered by digital pens, such as pressure or tilt, to change pen modes. Xin et al. [61] investigated the natural use of pen tip pressure, tilt, and azimuth (PTA) as a means to find correlations between the different modalities that could be used to switch pen modes implicitly. Their findings conclude that the use of pressure is the most likely to cause incorrect mode-switches because of the wide range of pressure (61% of total pressure sensitivity range) used when writing. In comparison, azimuth is the least likely due to it only using 24% of its full range. The authors also note that PTA profiles can directly benefit adaptive mode switching, which is the main focus of this work.

Li et al. [24] extend the idea of using pressure to switch modes by investigating the performance of pressure-based mode switching, along with four other techniques. After a performance evaluation, they noted that the pressure technique, while promising, did not perform as well as the other techniques practically. However, they also noted that each participant had different pressure readings while using the method — the method's performance could be significantly improved if they accommodated that personalized pressure space.

One approach, which diverges from the traditional idea of utilizing pen features to switch modes, used the physical characteristics of the stylus itself to switch modes. Vogel and Casiez [60] investigated using a stylus in the shape of a Conté crayon to change interaction modes with a single hand. The idea was based on the traditional use of Conté crayons, in which different surfaces such as fine-tip drawing and flat shading, can be used for various artistic effects. With their system, several different pen modes were attached to the physical edges of the stylus which allowed modes to be changed quickly. While effective, the technique cannot support large amounts of different stylus modes due to the number of surfaces available. In addition, careful consideration must be taken when selecting surfaces for the modes, so as not to confuse users; both of which are limitations of this technique.

Other approaches to optimize mode-switching include displaying a list of pen modes (highlight, underline, etc.) after a detected pen stroke. Recording and monitoring the position of the pen with respect to on-screen elements [26, 27] enables the detection of specific gestures (delimiters) in a pen-stroke [18]. Systems such as Scriboli by Hinckley et al. [18] and Zeleznik and Miller [62] rely on such gestures (e.g. drawing a pigtail) to change mode. Research shows that these methods are a viable alternative to the standard menu because they are fast, unambiguous, and expressive. However, they require users to memorize gestures and interweave it with regular annotation.

In contrast to these previous interaction techniques, several methods attempted to infer pen modes which would preserve the user from manually switching modes. The Inferred-Mode protocol developed by Saund and Lank [47], uses a decision tree to determine which mode the user intends (draw or select), based on stroke shape and context (location) or displaying a menu as a fallback. A key drawback of this method is the detection of the intended mode *a posteriori*, which prevents the system to switch pen modes while strokes are laid out on the screen. Changing modes modifies the appearance of strokes after they are made, resulting in potentially visually disruptive experiences. In later evaluations [34], it was also found that participants struggled to make mental models of the mode-switching capability, which limited its capability among users.

In this thesis, these works on inferred modes are revisited and extended through deep learning for a low latency approach for mode inference between drawing modes. Rather than a button for ambiguous modes, gesture-based fallback methods are provided to cycle recent strokes between modes or remove them entirely. Analysis of a stroke's mode is done tens of times a second while a stroke is being drawn, which allows for low latency mode detection that can be used to update the visual representation of a stroke shortly after drawing begins.

3 Data Collection

The goal of my Magic Pen technique is low latency automatic pen-mode inferencing during document annotation. To guide the research, I designed and ran a data collection study on how people hold and use digital styluses while annotating with different pen modes. The study collected pen-data features such as pressure and tilt, with the analysis investigating property variances across modes.

3.1 Study Design

To reduce the amount of variance between strokes made between users, I limited the study to right-handed participants only. This restriction was done because left-handed people were found in pilot testing to make strokes in a different manner than right-handed individuals, potentially affecting the likelihood of discovering consistent mode-signifying patterns. In total, nine participants were recruited from the Ontario Tech University student population. Participants were screened to see if they had experience with digital annotation and unimpeded use of their hand and wrist. Each session lasted one hour and participants were compensated $40 for their time.

The study was conducted in a private, quiet laboratory with only one investigator present. A Microsoft Surface Pro 6 and a Surface Pen were provided to the participants to perform the tasks. External variables, like device orientation, were controlled by laying the tablet flat and taping it to the table 30cm away from the participant. Although this restriction incurred a loss of external validity, and participants noted that the orientation felt unnatural at first, I wanted to collect consistent pen pressure and orientation data for a proof-of-concept. Later development could be generalized to varying tablet setups if shown to be promising in the restricted case.

Participants were instructed to perform a series of text-annotation tasks while using the data collection interface which recorded each interaction the user would have with the digital stylus. Participants were ensured that they were not going to be assessed based on their performance or the legibility of their handwriting.

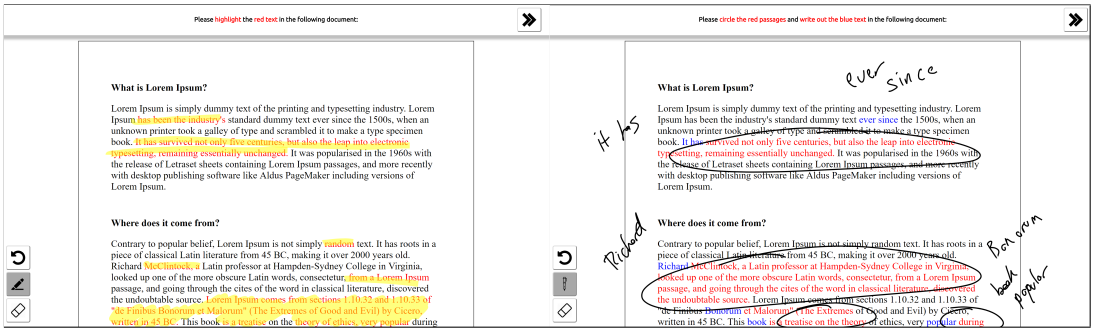


Figure 3.1: Example task screens from the Stroke Recorder System: (left) highlight (right) draw.

Different pen modes were used to conduct four annotation tasks on randomly selected passages of 10–30 characters. Each of the tasks was repeated twice in counterbalanced order for a total of 4 tasks x 2 trials = 8 task instances:

T1 **Highlighting** red passages from a document

T2 **Underlining** red passages from a document by drawing a line at the base of text

T3 **Striking out** red passages from a document by drawing a line through the middle of text

T4 **Drawing** circles, brackets, stars, asterisks, and arrows near the red passages while **writing out** blue text passages (Figure 3.1 right)

Each task screen allowed the participant to switch between the task's tool and the eraser, along with an undo button also provided. The annotation tasks were chosen to closely replicate the typical actions undertaken in document annotation scenarios, such as editing a manuscript, studying, or grading papers. The draw mode, also referred to as 'inking', needed special consideration, as there are a variety of different marks one can write within that mode. I attempted to cover these possibilities, by having participants draw various symbols and write-out text. Thus, the drawing task consisted of several sub-tasks (e.g. drawing circles, brackets, starts, asterisks, and arrows) which were presented on sequential screens.

After the annotation tasks were completed, participants were given concluding open-ended tasks in which they could use all the pen tools, in any order, to markup a provided document. The only restriction was to use each of the provided pen modes at least once. This phase was used to collect data on annotation behaviour in a more natural environment so that the interleaving of the different pen modes could be investigated.

3.1.1 Stroke-Recorder System

To allow for the capturing of information about how people hold and use digital pens when they annotate, a system to log stylus information was needed. When creating this system, an emphasis was placed on three key factors:

Low Latency: User engagement with pen interfaces is directly tied to interface (Figure 3.1 left) performance.

Flexibility: Support a variety of different pen tools and hardware platforms for remote deployment.

Data Completeness: Collecting all information regarding the pen and stroke may inform the classification.

After a comparison of these requirements to available platforms, I chose to create a web-based interface because of its ease of use and accessibility. While data collection was done in the lab on a single device type, I envisioned this tool as the basis of a future system for automatic mode switching. Thus, an added benefit of this decision was that the software would run on any pen-enabled device which followed the W3 pen standard [10].

Using the Javascript Pointer-Event API [35], the stroke recorder system collected all available features relating to the digital pen, and the status of the software (`time`, `pen-mode`, `text elements under stroke`). Some API-supported features were dropped because they were stroke invariant (e.g. `device operating system`) or not supported by the hardware/browser combination used (e.g. `twist`). These features were collected at an irregular rate due to Javascript's event-based nature, with most samples being around 10–15ms apart from each other.

3.2 Study Results

Prior to selection, study participants were asked about their experience with digital pens, to determine if expertise had a significant effect on performance. Seven out of nine had at least two or more years of experience with pen-based devices ranging from the Samsung Note devices to more advanced systems such as the Apple iPad Pro. The participants' prior experience seemed to affect how they interacted with the surface device. Those with less experience generally took longer to get adjusted to using a digital pen, whereas the more experienced were able to adapt themselves to the setup quickly.

After collecting data for 11,338 strokes using the stroke recorder system, the pen stroke's characteristics were then each individually considered for inclusion in automatic mode classification approaches. Before any information could be interpreted, it was necessary to pre-

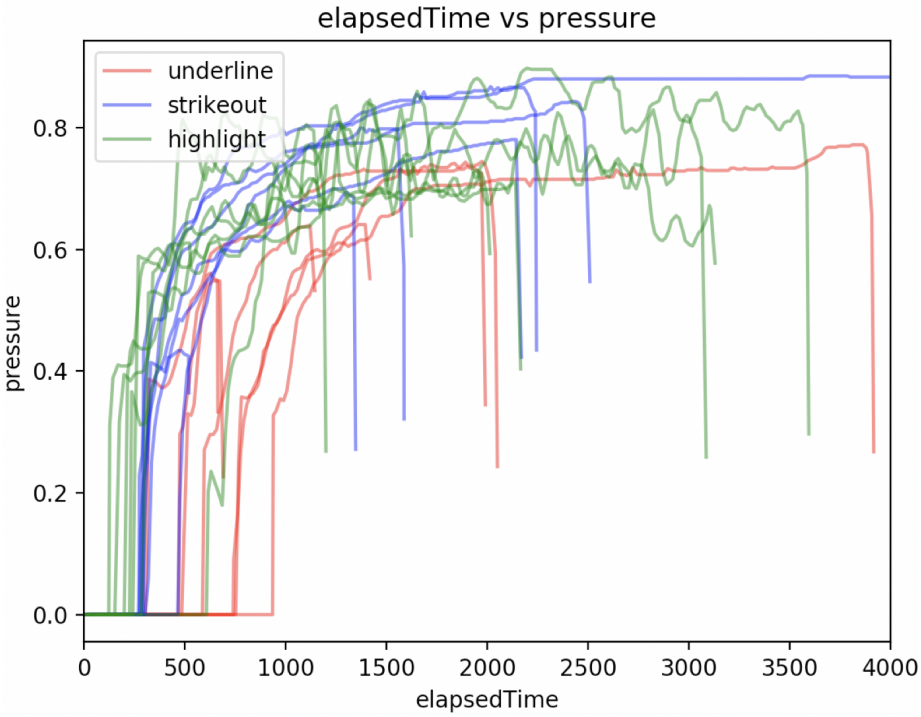


Figure 3.2: The difference in pressure between the pen modes. All modes follow a logarithmic trend at the beginning, with variances towards the end.

process the raw data into collections of ten stroke samples called 'stroke windows'. To transform and prepare the data, individual strokes were grouped to form stroke windows with particular features transformed into deltas, such as X-and Y-position, to prevent the stroke recognition model from learning absolute aspects of the tablet used within the study.

Charts visualizing the distribution of strokes within each pen mode were created in Python using the PyPlot library for Python [19]. The strokes were plotted with each line on the line graph depicting an individual stroke, and its colour representing the mode that the stroke was associated with. Upon creating the plots, it was discovered that there was a series of trends between pen modes. In particular, it was seen that for the various pen modes (aside from the drawing mode), there were patterns as to how a user would stroke in features such as pen pressure and tilt. Some of the trends can be seen in Figure 3.2. Each mode followed a logarithmic trend in terms of pressure, but the highlight mode, for example, was inconsistent as the stroke continued. This behaviour is explainable, since, throughout the studies conducted, it was noted that participants seemed to interact similarly between the two modes. The lack of differentia-

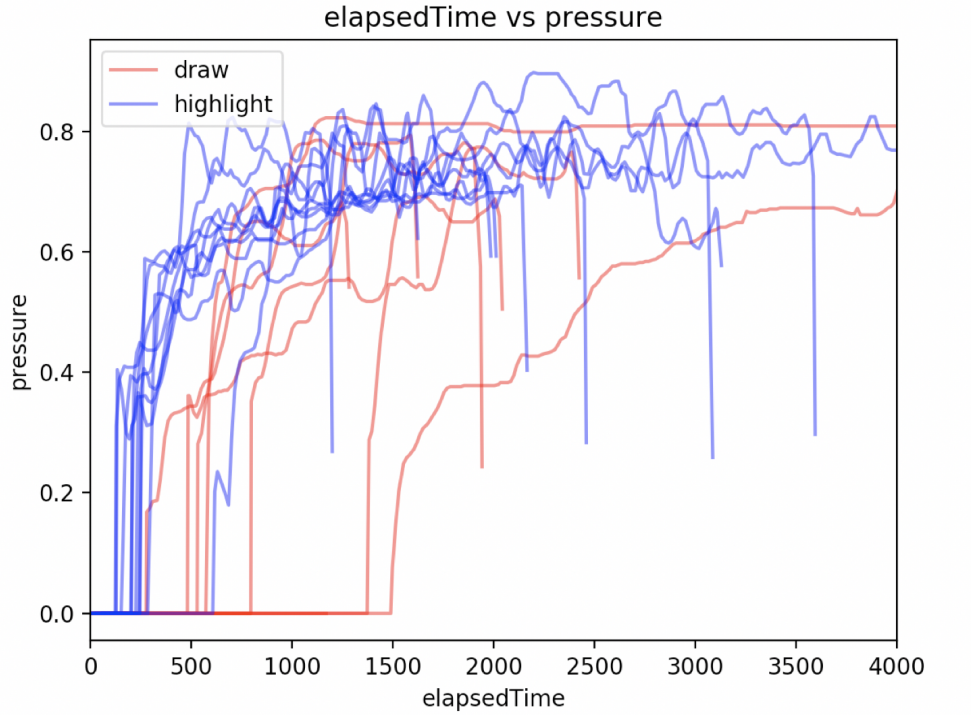


Figure 3.3: Pen pressure for the draw and highlight modes. The pressure profile for the highlight mode has a set form to it, while draw is much more inconsistent.

tion between the two modes proved to be problematic, as it became difficult to distinguish the two patterns based on pen data alone.

In addition to being distinct, all of the strokes were observed to be unique in comparison to draw. An example of this differentiation can be seen in Figure 3.3, with regards to pen pressure, where the highlight pen mode appears to be consistent and repetitive in form, while the draw mode does not seem to exhibit this steady structure.

Originally I hypothesized that the annotation leading up to a stroke would be significant in determining a resultant pen mode. This lead-up was thought to be significant because of the hypothesis that a user would hold a digital pen above the screen in a distinct way for each mode; providing enough information to allow for the pen mode to be interpreted before the pen even touches the screen. My hypothesis came from the original idea, in which a user would hold and use a pen in different ways depending on what pen mode they intended to use. In practice, however, the hover information seemed to introduce a great deal of noise, which led to its removal.

3.2.1 Final Feature Set

In addition to charting the data, I ran a series of experiments to determine if the Machine Learning (ML) system's performance improved with each attribute's inclusion, also known as feature selection. The experiments were run by building up a model for each combination of the features until it was determined that a feature set that performed optimally was achieved. During these experiments, the classification of strike out versus highlight was highly problematic. Strike out was eventually dropped from all further work.

I selected a subset of the collected and derived features, which were both promising through my analysis and informative to my ML classifier in experiments. In testing, it was found that not all features should be included as they introduced a significant amount of noise and could introduce latency in the real-time application of Magic Pen. Thus, the following features were selected as inputs to the ML model described in the next section:

`Delta X, Delta Y:` Change in the pen's position on-screen since the last recorded sample.

`Pen Pressure:` Pressure applied to the screen using the pen. Values range from 0.0 to 1.0.

`Delta Pen Pressure:` Change in pressure applied to the screen since the last sample.

`Nearest Objects:` Binary value encoding whether the pen is directly over text at any given moment.

`Closest Bottom:` Distance to the closest bottom of a text object's bounding box. Distances are encoded as a percent of the object's height.

The `Closest Bottom` attribute was a feature that was calculated post-hoc of the data collection. The feature was needed as the ML model tended to classify horizontal strokes at a great distance above text as underline. The attribute was calculated by taking the distance from the bottom of the closest text object's bounding box and then dividing that distance by the height of the text object to account for different font sizes in the model.

From what I learned in the findings, and with the data collected, I designed a system that transforms the raw pen information into a stroke-mode suggestion, described in the next chapter. This suggestion can then be used to change pen modes automatically.

4 Mode Switching Model

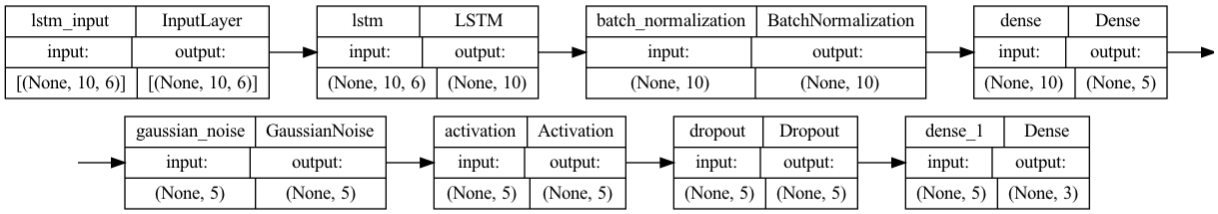


Figure 4.1: The network used in the ML technique to determine pen modes.

After collecting data and determining which features were most likely to be relevant for detecting pen modes, a stroke recognition model was created to identify the distinct modes. The main goal of this system, called Magic Pen, was to be able to switch between different pen modes effectively while avoiding incorrect pen-mode switches. This objective of a productive mode switching system was insisted upon because of the target to reduce the amount of cognitive load required within annotation interfaces. If the Magic Pen system produced an incorrect suggestion, the cognitive load needed to correct it would likely be greater than that of using a standard toolbar — defeating the system's intended purpose.

Another objective for Magic Pen was to produce a system that could reliably predict a user's intended pen mode at any point throughout a stroke with minimal latency. An early concern with the concept of automatic pen-mode switching was how the user would react to a stroke's mode changing, potentially as they draw, and how it could affect their annotations. This concern was found during an observation of the pilot runs of the prior data collection study where participants were found to immediately remove a stroke if it was in the incorrect mode, even if the mode would eventually be corrected. If Magic Pen temporarily switches to an incorrect mode, would the user immediately stop annotating, and how does it affect their efficiency and cognitive load? To avoid this possibility, an emphasis was placed on earlier classifications so that the "final" pen mode could be locked in as early as possible.

4.1 Machine Learning Approach

In order to switch between different pen modes while also keeping in mind the previously mentioned goals, a neural network was chosen. Long Short-Term Memory (LSTM) networks were initially chosen as a promising model for recognition. A LSTM was chosen because the data in use was temporal, and LSTMs have had extensive use in the past for temporal data in applications such as stroke and speech recognition [16, 55]. A training script was created in Python using the Tensorflow (TF) framework to generate the model [1]. The model's architecture can be seen in Figure 4.1.

In general, LSTM layers can learn more effectively with a higher number of hidden units, but this comes at the cost of requiring a significant amount of training data to generalize. Special consideration was taken with the number of hidden units for the LSTM layer to allow the model to learn complex trends while not overfitting. A series of optimization tests were conducted to determine the optimal number of hidden units and hyper-parameters by iteratively running the model with different combinations of varying values (a brute-force approach). A single LSTM layer with ten hidden units was the best compromise in terms of generalization and overfitting. Class weights were also calculated to even out the imbalanced training data so that the model did not overfit towards one classification specifically. Two sets of Dense layers then followed. Both Batch Normalization and Gaussian Noise regularization methods were used between the two Dense layers to avoid overfitting, as they have been used successfully in previous research [20, 54]. A single Dropout layer was also used following the Activation layer for similar reasons. Values of 0.1 and 0.15 were found to produce the best results hyper-parameters for both the Gaussian Noise and Dropout layers.

Regarding training the LSTM-based model, a batch size of 64 and a maximum of 500 epochs are used. This maximum allowed the model the greatest opportunity to learn from the dataset. Traditionally, with a large number of epochs, it is common to see extreme levels of overfitting. This phenomenon is partly due to the model gaining greater complexity geared towards the training dataset as it learns. The `EarlyStopping` TF callback [1] was used to avoid this overfitting while still allowing the model to grow. `EarlyStopping` functions by measuring the model's performance on its training data regarding the validation data. If the performance on the validation dataset stops improving after a threshold (called `patience`), the training stops. A patience value of 20 was found to be optimal in testing.

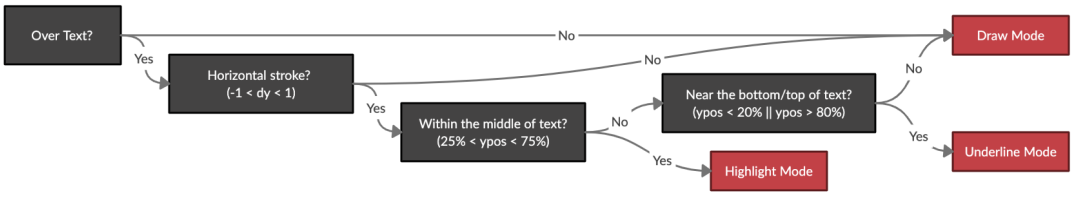


Figure 4.2: The decision tree used in the Rule-based technique to determine pen modes.

4.2 Rule-based Approach

Once the initial approach to switch pen modes was created, it became apparent that an alternative method which didn't rely on ML was feasible. The features that were significant to the model were observed, and it was found that the pen's position on the screen in relation to the other elements was strongly predictive of the pen mode. For example, the classification is likely to become highlight when the pen is near the middle of a word. Saund and Lank [47] made a similar discovery in creating the "Inferred-Mode" system — signifying that objects on screen might be a strong-enough feature to switch modes. With this observation in mind, I developed an alternative Rule-based approach and compared it to the ML method. The Rule technique has the drawback of not leveraging features such as pressure or motion dynamics to differentiate stroke types that occur outside the presence of text elements (e.g., in the margins).

A manually constructed decision tree was chosen to implement the Rule-based mode-switching system. The decision tree rules were selected by investigating the initial study's resultant data and observing the relative location over text for each pen mode. With this information, the primary rule was decided to be whether the pen was over text or not (see Figure 4.2). The decision stems from the fact that the highlight and underline modes were done over text nearly all of the time. Which, while often the case in real annotation work, is also an artifact of the data collection in which highlight and underline strokes were only asked to be made on text elements.

Following the initial rule, the rest of the checks see how far the pen is from the text's bottom. The distance is taken from the `Closest Bottom` feature, which is a percentage calculated by dividing the length to the bottom of the closest text object's bounding box by its height. The middle of the text element ($25\% < ypos < 75\%$) was chosen to trigger the highlight mode, while the top/bottom ($ypos <= 25\% || ypos >= 75\%$) was set to activate the underline mode. The top and bottom were chosen to trigger the underline mode as it was common for participants to make underline strokes which technically were on top of the text elements from the line below.

Similar to how the ML approach uses stroke windows to classify strokes, the decision tree also uses windows to make a classification. However, the difference between the two systems is that the latter only uses the most recent entry in the window, while the ML approach requires all of the entries. The use of windows for classification was done to keep the system architecture the same for both methods, allowing for comparisons between the two methods with the same input data.

4.2.1 Mode-Switching Pipeline

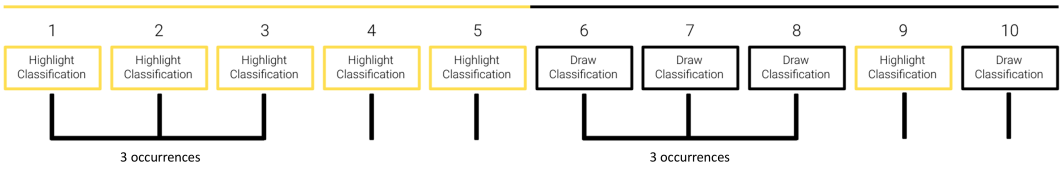


Figure 4.3: A visual representation of the mode selection technique. For a mode switch to be successful, there must be 3 classifications of the same mode in a row.

As an input, the model receives time samples (a *stroke window*), each window containing 10 samples 10ms apart. The model requires the use of all 10 entries in the stroke window to classify a stroke effectively. This window size equates to the model requiring 100ms minimum (10 samples X 10ms/sample = 100ms) to make a detection. The window size was determined empirically, to provide a good balance between accuracy and latency, vital to determine a stroke's mode as it is drawn. In addition, larger window sizes can increase the amount of noise in the classification, which should be avoided for performance reasons. The final model was trained on 6,007 strokes (110,754 windows, equally divided amongst modes) from 9 participants selected at random.

The pipeline to receive a classification from the model is as follows: First, record interaction data from the digital pen (pen sample), then transform the current and previously recorded sample into a stroke event by calculating specific stroke features. Once a stroke event is obtained, it is stored in a bucket (stroke window) until there are enough samples (10). This stroke window is sent to the model and is subsequently classified as a specific pen mode.

To transform a series of stroke classifications into a specific pen mode, a mode-selection technique is required. Special consideration was given when developing this selection technique as it needed to change pen modes only when necessary. This constraint was placed because it was hypothesized that rapid, sudden changes to a stroke's mode could be jarring for users and induce a higher cognitive load. In addition, the technique needed to filter out incorrect pen modes so that the stroke's mode was not changed incorrectly (although uncommon).

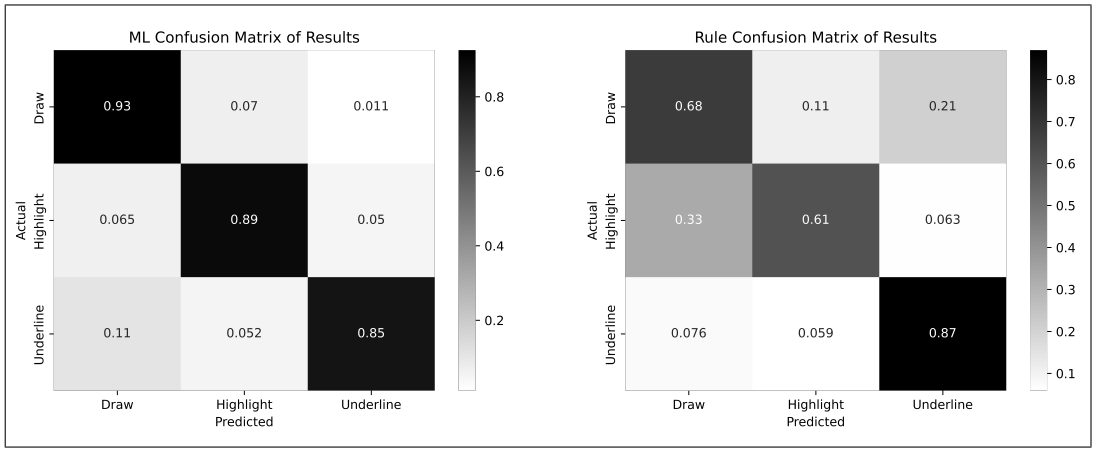


Figure 4.4: Confusion matrices of both techniques. ML generally performs better than the Rule-based counterpart, as shown by the higher values along the diagonal.

With these constraints in mind, a classification ranking technique was devised that chooses a final stroke's mode based on the most-recent classifications (see Figure 4.3). The algorithm ensures that the past X classifications must have the same classification before the stroke can have its mode changed. If there are less than X classifications, all recorded classifications are used. A value of 3 was chosen for X in testing as it allowed for the stroke's mode to be changed fairly easily while not allowing instability.

4.3 Performance

To determine whether the mode-changing model was effective at learning the user's different stroking behaviours, and which technique was more performant, an evaluation with the already-collected data was conducted. This performance was assessed by performing K-Fold Cross Validation with 10 folds, resulting in a set of 11,338 strokes randomly sampled from the 9 participants. After training on 293,997 stroke windows, the ML network achieved an overall accuracy of 88.55%, while the Rule-based approach achieved a 72.67% accuracy (see Figure 4.4).

The main benefit to using a neural network for the stroke classification task is that it can detect highlight and draw strokes consistently, despite some users making strokes that varied significantly from the mode's intended location. For example, in Figure 4.5 (top, on next page), some strokes can be seen at the top of the words (in the case of highlight) or even drawing over text. The model is seemingly able to leverage the additional features other than pen position to be able to classify these unambiguous strokes. In terms of errors, the model seemed to make the most errors within the underline mode, as the network commonly misclassified it with

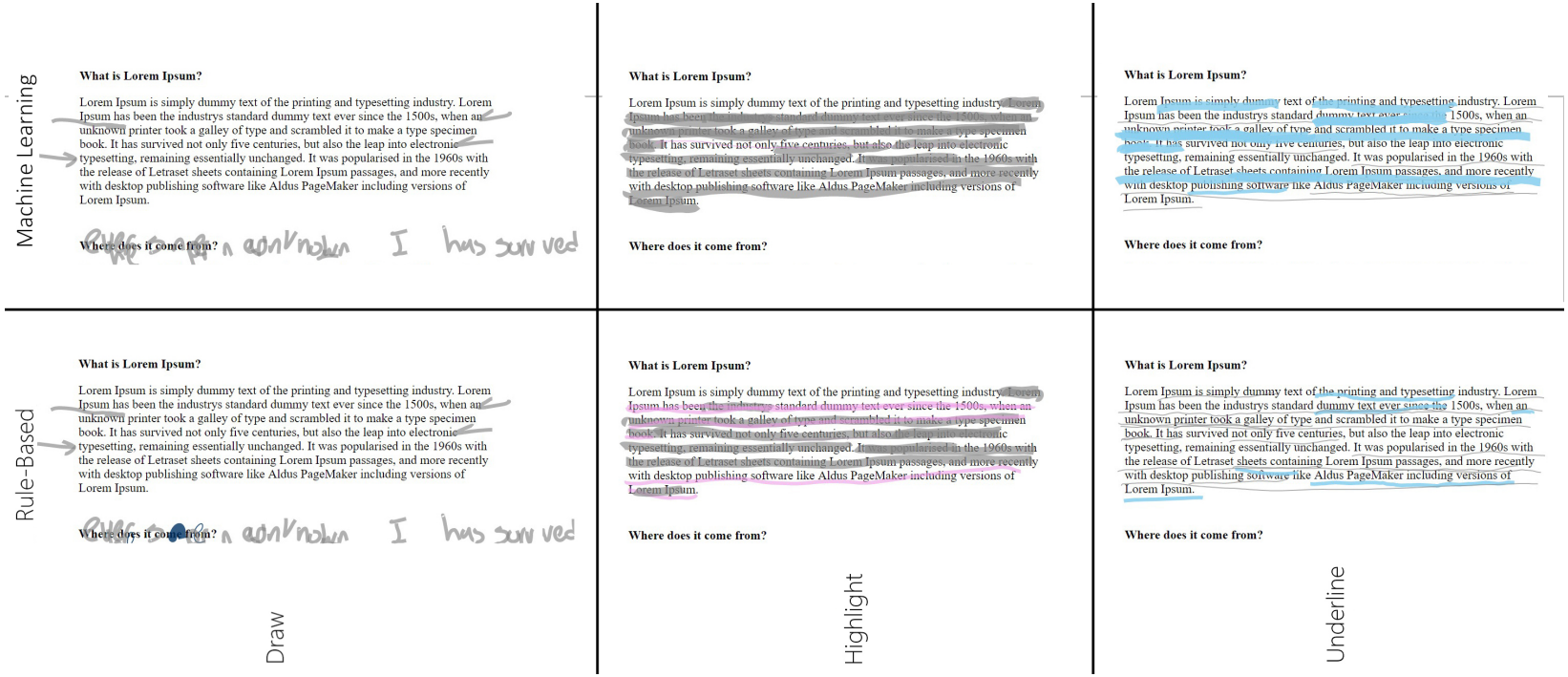


Figure 4.5: A visualization of the errors made using the ML approach (top) and Rule-based approach (bottom) on a sample of participant strokes. The ML approach classified purple correctly while making many mistakes with light-blue. The Rule approach had the opposite: classifying light-blue more effectively but yielding worse results with purple.

draw. Another common error found was that, on documents with very limited vertical spacing between lines, it was common for underline strokes to drift into the middle of word rows, causing classification mistakes.

Figure 4.5 (bottom) shows that the Rule approach can determine draw strokes consistently but makes errors between the other two modes. However, an interesting product of this technique's classification is that the results are reasonably predictable between modes. This predictability can be seen in the underline strokes where the Rule approach misclassified because the stroke at points was not near the top or bottom of the text. The same can be seen with the highlight strokes, where the classification became draw for the same reason. These errors may be mitigated by users adapting their writing to the model, though this goes against the goal of naturalistic automatic classification.

5 Eraser Mitigation

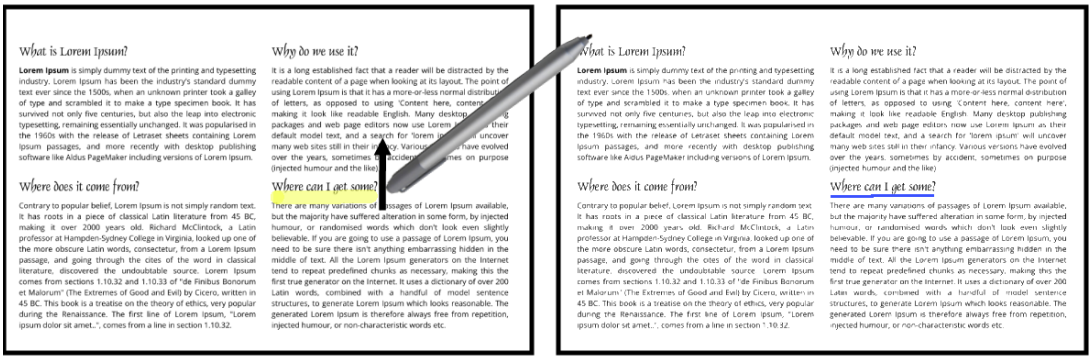


Figure 5.1: An example of the eraser mitigation technique. By using a flick of the pen, an incorrectly-moded stroke can be changed.

Although the stroke mitigation method used aimed to prevent users from manually changing pen modes, there are inevitable classification failures. Incorrect classifications can frustrate users, which may lead them to avoid using a mode-switching technique altogether [59]. The push for such a stroke-mode correction system was found during pilot runs of an evaluation study. Users were found to remove strokes in the (temporarily) incorrect mode immediately instead of continuing to annotate, which typically corrects them. This removal behaviour demonstrated the need to lock in a correct pen mode as quickly as possible, as noted earlier during pilot study runs (see Chapter 4). An error-mitigation technique was introduced to prevent such annoyances, which uses the eraser-end of the pen to perform a flick gesture (see Figure 5.1).

5.1 Design

Since this work aims to avoid traditional toolbar menus, speed was an essential factor to consider when developing a flick gesture. The goal was to develop a technique that would work in tandem with the eraser. In addition, the reliance on a menu was avoided entirely due to the thought that users would not want to use the mitigation techniques if they had to rely on the

use of a menu. Because of this, an approach that utilizes a simplistic menu button to change the modes was avoided.

Initially, the pen tip was considered for this task, as it would be quick for a user to hold a button and perform a gesture on-screen with it since they are already using the pen. However, due to concerns with users being confused about whether they were in the stroke-correction mode, this task was moved to the digital pen's eraser end. Flipping the pen allows for a clear distinction between stroke-correction and inking modes and brings the stroke correction mode in line with the eraser; another corrective mode. The correction gesture is simply a stroke up or down on an empty part of the screen to cycle the last-drawn stroke's mode. The cycling of the pen modes is kept consistent between strokes (Highlight $\rightarrow$ Underline $\rightarrow$ Draw) so that users can learn to flick up or down to get reach the intended mode. The eraser on the pen acts as a regular eraser when moving over an existing stroke, removing the stroke in question and not triggering the mitigation technique.

Although intended to be activated with the digital pen's eraser-end, not all digital styluses have an eraser. For pens with barrel button-activated erasers, this method is also used for error mitigation. Although less common, some styluses also feature no barrel buttons at all. In this case, a toolbar button was added to activate the eraser functionality on the pen tip. When activated, the flick gestures worked with the pen tip as well. It is important to note that these options are not considered optimal but were added to maximize compatibility for the subsequent remote deployment study.

6 Evaluation Study

A remote-deployment study was conducted to evaluate the two automatic mode switching methods against each other and a traditional toolbar menu. The study collected quantitative stroke data from participants using each mode, as well as subjective feedback. The results provide valuable insight into the drawbacks and costs associated with each mode, possibly leading to better implementations of the systems. The institutional research ethics board approved the study (REB FILE #15755).

6.1 Study Design

In this section, the overall setup of the study will be outlined along with the design considerations chosen to best record the participant preference in terms of mode switching methods. In addition, the participant demographics will also be summarized.

6.1.1 Tasks

The study consisted of a series of annotation tasks in a similar format to the initial data collection study. The main variation came in the form of the mode switching technique made available across three conditions. Each condition was colour-coded so that participants would not be biased by the names. The three conditions (techniques) were: Red Mode = ML-based system, Blue Mode = Rule system, Green Mode = traditional toolbar menu. The order of conditions was counterbalanced.

Each of the conditions began with a demonstration video, followed by a practice phase in which participants performed at least five strokes in each of the three pen modes and used the flick gesture at least once. Directions were provided by marking the passages to highlight, underline, and write out in different colours. Passages were randomly selected to be about 15 words long. Participants were allowed to continue until comfortable with the method. The second markup phase asked the participants to annotate a page of a text document, but this time they were asked to annotate all marked passages fully and correctly. Annotations could be made in any order, interleaving modes if desired. Strokes made in practice and markup phases were recorded to investigate learning strategies (practice) as well as performance (markup).

#	Question Criteria	Question
1	None	The technique switched to the correct pen modes I attempted to use.
2	None	I felt efficient with the technique.
3	None	I felt frustrated with the <*colour*> technique.
4	None	I felt distracted by the <*colour*> technique.
5	None	I felt the <*colour*> technique required more effort than usual.
6	None	I felt the result of the <*colour*> technique was predictable.
7	None	I felt I adapted my interaction with the pen as I used the <*colour*> technique.
8	Used Eraser Flick	I found the "flick" eraser gesture to be a fast way to correct the mistake.
9	Used Eraser Flick	I found the "flick" eraser gesture to be an intuitive way to correct the mistake.
10	Did Not Use Eraser Flick	Did you have any errors that you needed to correct?

Table 6.1: The questions used, and the criteria for each question being displayed, within the feedback questionnaire. Techniques were represented by colours in the study — red, blue, or green.

After the markup phase, participants completed a feedback questionnaire consisting of seven 5-point Likert-scale questions and an opportunity for free text feedback. Likert questions consisted of statements in the form "I felt <*adjective*> with the <*colour*> technique." (strongly disagree – strongly agree, see Table 6.1). Questions were designed to explore satisfaction with the technique, as well as hidden costs in terms of mental strain or complexity. Participants were also asked whether they used the mitigation technique (i.e. flick gesture), and if so, whether it was *fast* and *intuitive*. If it was not used, they were asked if they had any mistakes to fix, in order to gauge when people chose to erase rather than use the flick gesture. The complete set of questions is embedded in the results in the following section.

After each condition, participants were offered the opportunity for a break. These steps were repeated for each condition for a total of 18 participants x 3 blocks of trials x 2 screens per trial = 108 annotation tasks. After completing the tasks for all three conditions, a final questionnaire asked participants to rank the techniques from most to least preferred (tied ranks were not allowed). A text field is provided if the participant has any final comments on any of the methods used.

6.1.2 Participants

The criteria for eligibility within the study consisted of having some form of experience with digital pens in the past and owning a Windows-based device that featured pen capabilities (to allow remote deployment during the COVID-19 pandemic). Seven participants used a digital stylus with an eraser on its end and eight used a device that featured barrel buttons. The remaining three used an onscreen button to enter the flick gesture mode. In total, 18 partic-

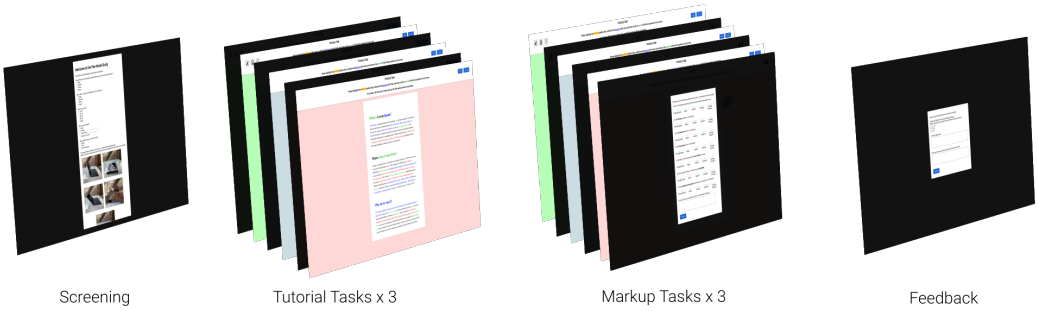


Figure 6.1: An Overview of the Evaluation Study's Task Progression.

ipants aged 18-39 participated (11 male/7 female). 16 participants were right-handed and 2 were left. I did not restrict the participation based on hand dominance after pilot testing with left-handed lab members who achieved similar performance from both ML and Rule methods. 20 participants agreed to do the study, but complete data was only received from 18. Snowball recruitment was through university email lists to students, faculty, and staff, with an open invitation for participants to invite others. Compensation was given in the form of $20 Amazon eGift cards.

6.1.3 Setup

The study was conducted through a website that participants visited on their devices, on their own time, and without synchronous assistance. The study consisted of a linear series of screens in which questionnaires and tasks were administered. Participants were asked to complete the study in one sitting of approximately one hour. Several rounds of pilot testing were used to refine task instructions which were presented in text and video. Due to the study being conducted remotely, it was impossible to control the device orientation. Instead, the device setup was collected along with demographic information in an initial questionnaire. See Figure 6.1 for a visual example of study progression.

6.2 Results

With the exception of one participant that took an extended break, participants took roughly four minutes to complete each phase of the study, which included marking up the provided document and answering questions.

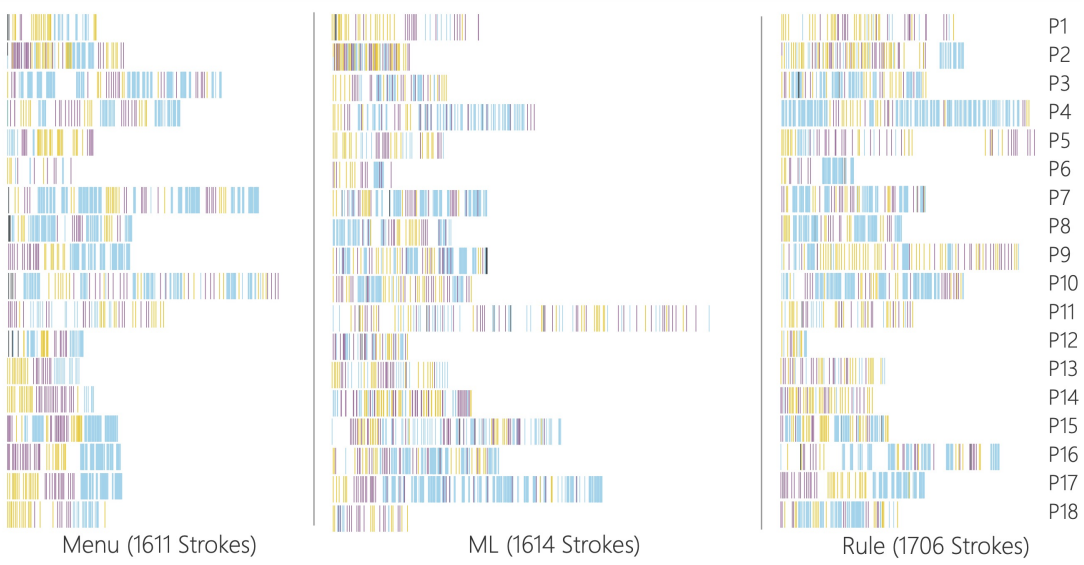


Figure 6.2: Temporal stroke sequences made by each participant, binned for each study phase. Blue represents draw, purple is underline, yellow is a highlight, and grey is unclassified (too short to classify or no mode selected).

6.2.1 Stroke Data

Comparisons between the different techniques and phase types showed extensive use of the opportunity to practice, as well as more strokes made with the Rule technique. The higher strokes in the Rule technique did not correlate to order or to the user's overall technique preference, as most participants made more strokes in the Rule technique regardless of preference.

Inspection of the strokes made within the study and modes revealed some interesting trends (Figure 6.2). In this figure, stroke modes are coloured based on their final mode (if corrections were applied). It was found that when using the Menu technique, participants would commonly annotate the provided document in a block manner, marking up all sections that required one pen mode before moving to the next. This is particularly visible with P9,13,14,16,17,18. For example, in the Menu technique, P13 completed all highlights, then underlines, then draw strokes. However, this pattern was not common with either of the automatic methods. Overall, 29.3% of strokes in the ML technique used a different mode than the previous stroke, compared to 22.2% in Rule, and 10.6% in Menu. The block sequence of modes in the Menu technique may be a result of all stimuli being revealed at once. In a real annotation technique, one would not know in advance all passages to highlight or underline. However, this result shows the different workflows adopted by participants based on the behaviour of the tools, with automatic methods encouraging more mode switching.

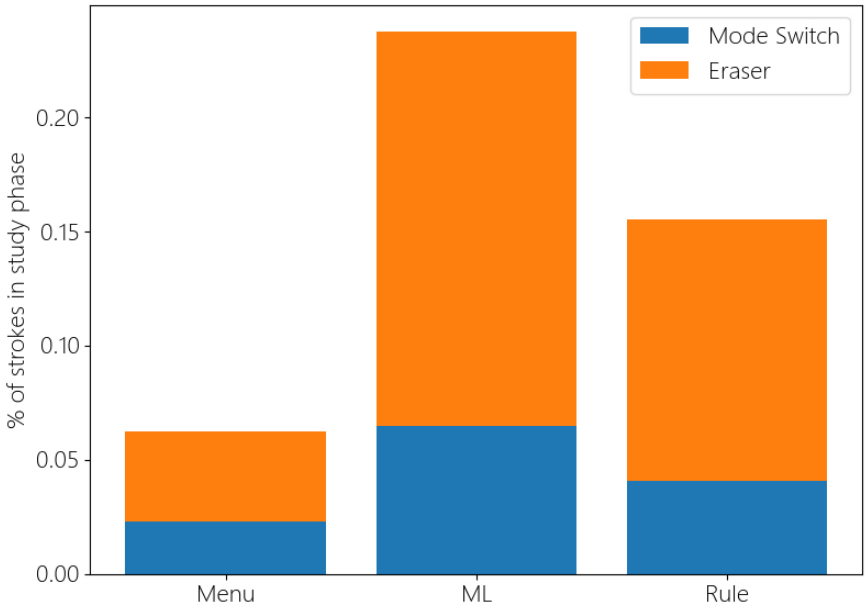
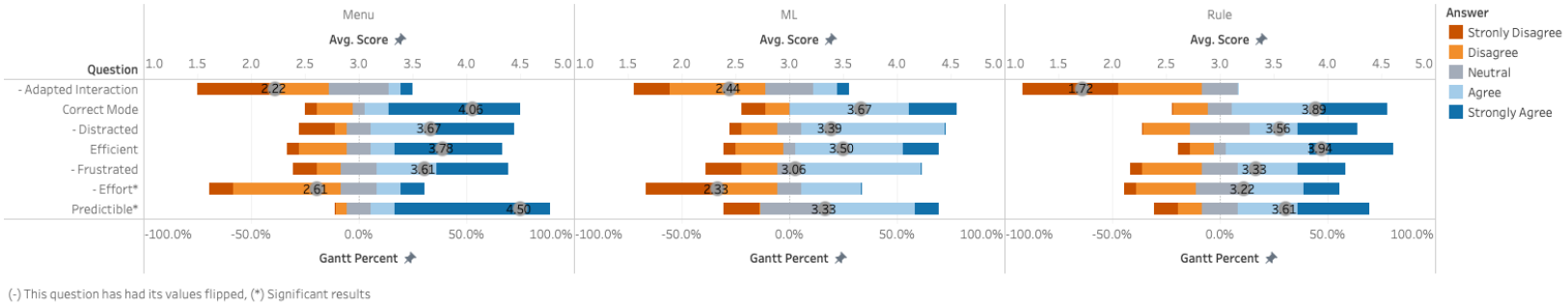


Figure 6.3: The percent of edited (blue) and erased (orange) strokes in each study phase by the technique used.

6.2.2 Corrections and Erasures

Corrected strokes using the flick gesture indicate a mode failure, either through automatic classification or a failure to change the mode correctly with the Menu technique. Erased strokes may also represent mode errors, but may also represent participant mistakes when writing. In total, 749 strokes were edited in some way (15.2% of all strokes made in the markup phases). Of those strokes, there were 212 strokes (28.3% of edited strokes) that were mode-corrected with the flick gesture and 537 strokes (71.7% of edited strokes) were erased. Figure 6.3 shows the results between techniques. The fraction of strokes erased or mode-corrected increased from Menu to Rule to ML for both. The lower rate of mode corrections for Menu is likely due to the user explicitly choosing the mode, but interestingly the correction technique was still used occasionally to correct mode errors. For the Rule method I hypothesize that the rate may be lower than ML because participants learned to adapt their behavior to the predictability of the rules. Participants provided comments to improve the flick gesture, with the two most common themes being concerns about activation accuracy and the need to activate it in open space. P5 said, *"There were multiple times in which I attempted to switch the modes with the eraser flick, it would erase something that I had done previously, causing me to have to redo the lining or highlighting I had just done"*.



(-) This question has had its values flipped, (*) Significant results

Figure 6.4: Subjective feedback for each technique. Significant differences were found for the level of Effort (Rule requiring less effort than ML, $p<.05$) and Predicability, with Menu better than ML and Rule, $p<.05$. Note some scales are reversed as indicated such that higher scores are always better.

6.2.3 Subjective Feedback

After each markup task, the participants were asked to rank each of the three mode switching techniques. In total, 8 participants chose Rule as their favorite, with 7 preferring Menu and 3 choosing ML. In terms of the lowest-ranked techniques, Rule was chosen 4 times, 6 chose ML, and 8 chose the Menu as their least favorite. A Wilcoxon signed-rank test (p = .05) was performed on all the pairs of rankings with results determining that no specific technique was significantly preferred over another. The rankings results show a split between the predictability of the Menu approach (7 favorites) versus the efficiency of one of the automatic methods (11 favorites) — hinting that any automation is a matter of personal preference between participants. This polarization is supported by the fact that the Menu technique received only 3 ranks of '2'.

The responses to the Likert questions are summarized in Figure 6.4. Note that the scales for *Adapted Interaction*, *Distracted*, *Frustrated*, and *More Effort than Usual* have been reversed so that higher values are better for all rows. Measures were compared pairwise with a two-tailed Mann-Whitney U test at p<.05. I see that the Menu technique was found to be most predictable compared to ML (U=96,p=.038) and ML (U=65,p=.002). Menu was also deemed most correct but no significant distances were found. The Rule technique was found to be efficient and fairly predictable. Interestingly, the Rule technique had the best score in terms of effort and was significantly lower than ML (U=99,p=.048). Also, participants noted that they strongly adapted their interaction to use the Rule technique, showing they may have learned the hidden rules to achieve correct classifications. The degree of adaptation was lower for the ML technique but not significant. Participants reported slightly higher correct mode classifications for Rule over ML, which is supported by the higher use of the flick gesture in ML.

Overall, the results suggest that while each automatic technique was not significantly preferred over a traditional menu, automated approaches were preferred by 11 participants. When testing the two mode switching techniques, people favoured the Rule implementation overall. Participant comments point to misclassifications in ML being the main challenge. P4 said *"Red (ML) was very similar to blue (Rule) for me but it seemed like it wasn't as intuitive as the blue mode"*. Specifically short strokes seem to challenge the ML technique, with P8 saying *"I found it difficult when writing horizontal lines in my letters like f and t"*, and P15 and P16 mentioning the dots of i and j being misclassified, leading to the need to erase the whole word. Though, some participants thought the ML technique was best. P2 said,"*The red (ML) technique felt the smoothest of them all*". P16 thought the automatic techniques were *"very helpful and saves a lot of time"*, and were essentially tied, saying *"red (ML) and blue (Rule) methods were both intuitive and saved me time moving the pen across the screen to select the tools"*.

In terms of stroke corrections made, the Rule technique had a higher concentration of errors made within the initial practice phase and had the largest drop when moving to the proper markup phase. I hypothesize that this stems from how comparatively simple it is to infer how the decision tree produces a stable result, allowing participants to adapt their interaction to better use its abilities. This claim is backed by the Likert responses from participants (Figure 6.4), where they signified that they adjusted their interaction more for the Rule-technique than any other. P18 said, *"The Blue (Rule) technique simply had the best 'logic' for handling mode switching"*. However, the simplicity comes at a cost as the Rule technique in its current form would not support a new pen-mode, such as strikeout, whereas the ML approach could possibly be retrained to support it. ML approaches may also be more easily customized to individuals through transfer learning to bias the model toward individual writing styles. These results demonstrate although a mode switching technique might not have the best performance, participants will find ways to be efficient with it given that the mode switching ability is predictable and its result is stable.

In terms of comparing the Rule technique and the standard Menu approach, the decision seemed to be determined mainly by the user's personal preference for automation versus familiarity and predictability. P8 said, *"I liked the Green (Menu) technique since I had more control of the tool that I wanted to use"*. P15 said, *"Green (Menu) technique was easier since I knew I did not have to expect or account for any mistakes other than my own"*. In contrast, others found the Menu overly inefficient compared to the Rule technique and even commented on the requirement of remembering what mode you're in while using it. P2, who did their menu work in mode blocks (see Figure 6.2) preferred the menu, but drew attention to how that preference would change if the work required frequent mode switches, saying *"Would become very frustrating if frequently switching between pen modes"*.

This personal preference aspect suggests that an ideal interface should offer both options in terms of mode switching techniques: an automatic method for frequent switches (aka those that prefer efficiency) and a manual menu-based approach for infrequent switches or for those that prefer predictability. A hybrid of both approaches may be best, as suggested by P13 who said *"I believe a combination of Green (Menu) and Blue (Rule) would be perfect for me"*.

7 Refined Machine Learning Model

One of the most substantial items learned from the evaluation study was that despite its drawbacks, the ML technique best supported the task of automatic pen-mode switching. However, it was also discovered that the "unstable" mode switching behaviour of the ML model could lead to frustrating users more than the technique it is meant to replace (e.g. a toolbar), defeating the purpose of implicit switching. As a result, the ML technique needed to perform better to be seen as effective. Several factors of the model were tweaked as a means to achieve this improvement.

7.1 Updated Mode Classification

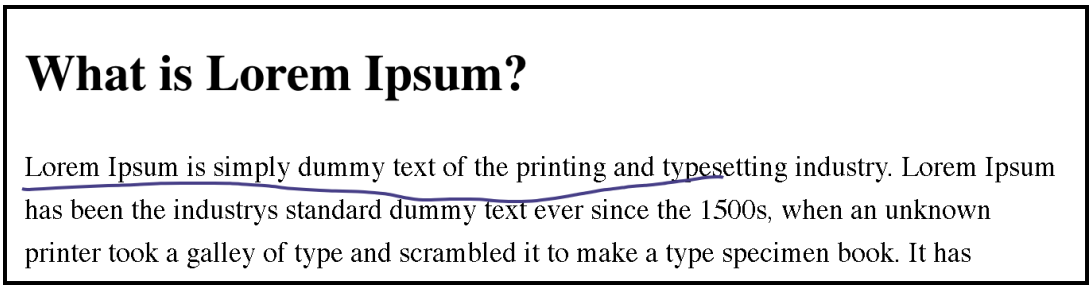


Figure 7.1: A visual demonstrating an underline stroke made by a participant. While intended to be straight underneath the sentence, the stroke drifts towards the sentence below.

After the results of the second evaluation were collected, some of the faults of the ML model became evident. One of these main challenges, as mentioned previously, was the use of user-specific strokes that differed greatly from what the model had learned. An example of this is an underline stroke: typically located at the bottom of a sentence. However, some strokes would drift towards the middle of the words below those intended to be underlined (e.g. Figure 7.1). The model had difficulty with these specific strokes because they ventured so far from what was used to originally train said model, causing it to be unclear about the stroke's intended mode. Heuristic methods were investigated to aid the model in resolving these ambiguous, short strokes by relying on annotation-specific trends.

A separate technique that utilized heuristic-based information was created to aid the model's classification confidence in moments of uncertainty. The first piece of information used within this heuristic technique was the stroke class of the previously drawn stroke. The concept was based on a hypothesis created after observing users of the first evaluation study, where users were found to create strokes in the same mode after one another rapidly. This noticed phenomenon was utilized within the classification pipeline to directly classify a new stroke with the same pen-mode as the last if the time delta between strokes was less than a certain threshold, bypassing the use of the ML model. However, the model is still utilized if the stroke's creation extends past this threshold, leading to both heuristic and model classifications (passed to the mode selection algorithm). In testing, 500ms was optimal as it aided classifications for particularly ambiguous, short strokes (e.g. the stick and dot in the letter 'i'), which, as mentioned before, were found to be particularly hard to classify correctly due to the lack of data available.

However, in events that involve the stroke being drawn to be temporally outside the threshold window of 500ms, the previous stroke's mode is still utilized to some degree, as follows. Despite previous changes to improve model classification performance, it was discovered early in model evaluation that the model would produce a series of low-confidence classifications for select strokes and annotation styles. While normally filtered out via the mode-selection techniques, these low-confidence classifications would occasionally occur often enough in a row to incorrectly change the pen's mode. The previous mode's stroke was used in place of the model's classification to overcome this mistake; specifically, if the stroke was outside the previous technique's threshold window of 500ms and the model's confidence of the last window classification was not above 75%. 75% was selected since any lower values were found to still allow for sub-optimal classifications and higher values began to block the use of new, correct classifications. Unlike the earlier heuristic-based technique, which uses the last stroke's pen mode, this technique does not change the final mode of a stroke, as only a single-window classification is replaced. All classifications must still be processed through the mode selection technique, which adds a filter before changing the final pen mode.

7.2 Updated Mode Selection

As mentioned previously, the mode-selection technique is critical to the implicit pen-mode detection pipeline. It is critical because it is the step that, ideally, allows for correct mode changes while also filtering out incorrect modes. While the initial implementation selected modes based on whether the last X classifications agreed, it was clear after the initial evaluation and further testing that it would need to be refined. The main issue with the technique was that, at lower majority threshold values, nearly every mode change would be allowed; even incorrect ones.

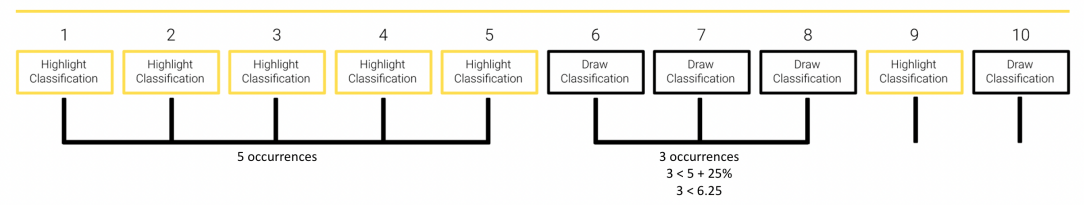


Figure 7.2: A visual demonstrating the revised mode switching technique. In the example, an initial highlight mode switch occurred, while a subsequent draw mode switch did not occur because it did not collect 25% more sequential classifications.

However, when the threshold was increased, there would be fewer mode changes, which affected correct changes. In addition, it would adversely affect shorter strokes, some of which did not have X classifications.

With these new findings, a revised mode-selection technique was created that chooses stroke modes based on the number of sequential occurrences of classifications (Figure 7.2). For a new mode to be used, it must have 25% more sequential occurrences than the current stroke mode. The main benefit to using this technique over the original is that it allows for a form of stroke-mode confidence to be built. If a series of new classifications arrive and surpass the existing amount, the model must be confident in its selection. In addition, it prevents a form of pen-mode thrashing (rapidly changing modes) which would likely confuse and annoy users. While effective in testing, this approach has some limitations that may make it not applicable to other domains.

The first of the limitations to the revised selection technique is that a relatively sizable amount of classifications is required for a formation of confidence to be built. When there are few existing classifications, rapid mode changing can occur. For example, with one classification recorded, only two similar new ones are required to change modes. However, for the use case of stroke mode switching, this limitation was found to be a non-issue. The limited number of classifications generally only occur during the beginning of strokes, which, as previously mentioned, is expected to change its modes rapidly.

The second limitation of the technique is the opposite of that of the first; it becomes potentially difficult to change modes when a large number of classifications are recorded. This restriction is desired for pen-mode switching, as it ensures that the pen's mode will not change (after the beginning) unless necessary. However, a variable occurrence minimum could be used to lessen the impact of this limitation on other domains. For example, a minimum of 25% more than the current mode could be used initially, with a value such as 5% being used when more classifications are recorded.

8 Refined Eraser Mitigation Method

Although the previously implemented mitigation method for incorrect strokes aimed to avoid the potential frustration from making a stroke in the incorrect pen mode (due to the automatic mode-switching), the evaluation study demonstrated that it was not used often. While its lack of use is not entirely understood, it is likely that users preferred to remove and redo their stroke instead of changing its mode. As such, we experimented with methods to improve a stroke's removal, allowing users to redo their annotation quickly.

8.1 Previous Drawbacks

While effective in switching pen modes for an already existing stroke, there were a couple of factors that led to the mitigation method's lack of use. The primary reason was that the technique required its dedicated gesture for activation. Although users were taught and practiced this gesture exclusively, it was evident that the extra thought process required to use it deterred users from utilizing it. A cause for this added thought process or load, is the sequences of pen modes (see Section 5) which were effectively required to be memorized to be effective. For a mitigation method to be effective, it must allow users to switch pen modes quickly while not requiring a great deal of added load.

Another factor that inhibited the method's viability was the restriction of the method to the physical pen's eraser. As participants noted, there were times when they wanted to change modes but would accidentally remove a stroke on-screen instead, requiring extra effort to undo the action or to re-draw it. In testing, it was found that this accidental stroke removal behaviour largely occurred when a user would keep the eraser on-screen for longer than a few short seconds, as is expected with a flick gesture. While the threshold for determining a "flick" versus an "erase" could be tweaked, longer flicking windows would adversely affect the erasing interaction and vice versa. Therefore, for a novel mitigation method to be productive, it must have a clear activation method that can be distinguished.

The main goal of a stroke removal technique was similar to that of the mode-switching technique to reduce (or remove) the frustration felt when the automatic approach incorrectly selected pen modes while stroking. As such, both techniques share similar constraints in their

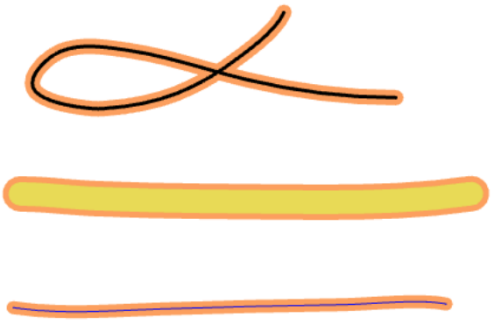


Figure 8.1: A Visual Example of the Stroke Highlighting. Orange was chosen because it minimally overlapped with common highlight colours.

implementation as well. For example, switching to a dedicated "eraser mode" can be slow, tedious, and frustrating if repeated numerous times. Therefore an easy-to-use method to correct the stroke is needed, which previously was implemented via a gesture with the eraser side of the pen. However, since time and mental capacity must be taken to remove and redraw the stroke, it was thought that the need to switch to a dedicated eraser mode might be too costly. This is thought to be the reason for its lack of use, eventually pushing for its removal.

8.2 Design

Different gestures were experimented with for the goal of speeding up stroke removals. The main constraint for the technique was to avoid using a separate eraser mode button since using the button would require a round trip to the toolbar. In addition, using the eraser end of the pen was avoided since it would require the pen to be flipped twice, adding an extra delay before the user can return to annotating. With these erasing constraints in mind, a tap on the main editing area on-screen was denoted as the gesture to remove strokes.

A tap on the screen was chosen due to its prevalence in previous work. For example, Lank et al. found that using an off-hand to tap the screen was faster than other gestures and was easily explainable to users [23]. Due to its easy-to-understand nature, this tap allows for it to be used while not requiring additional effort from the user, preventing the problems of the previous iteration. In addition, a tap on-screen was found to be fairly distinguishable compared to other touch-screen interactions (e.g. touch scrolling), also meeting the previously discovered requirement for mitigation techniques.

A translucent orange highlight (see Figure 8.1) was placed around the stroke to prevent confusing the user about which stroke could be edited. After a tap on the screen, the highlighted

stroke is removed. A translucent orange was chosen as it minimally interacts with the content on-screen while also being easily visible. As a means to not block content on-screen, the highlight was also made to disappear after five seconds. Five seconds was chosen because it was found in testing that users would remove a stroke generally within this five-second window after its creation. Otherwise, the stroke was in its correct mode.

Lastly, to prevent unexpected stroke removals during scrolling, a stroke-removal tap was designed only to be registered if the user taps the screen while not moving their finger far in either the X or Y directions. This limitation was found necessary since users did not tap the screen the same way each time. In addition, it allowed for the touch-screen to still be used for interactions like scrolling.

9 Transfer Learning

After the evaluation study, it became clear that a method to tune the approach towards a user was needed to predict user-specific strokes. To achieve this goal of user-tuning, Transfer Learning (TL) was leveraged to train an iteration of the model on one user's annotations — effectively tuning it towards their interaction style.

TL is an ML method in which a model is developed for a specific task, and then reused as the starting point for a second task. It is a popular approach within the field of deep learning due to its ability to allow for a pre-trained model to be used as a starting point, drastically reducing the compute and time resources normally needed [42]. Since its inception in the early 1990s by Pratt [42], TL has been applied to many different fields for a variety of tasks. For example, TL has been explored for its use in cancer research [22, 51], text classification [12, 49], spam filtering [29, 57], and even gaming [63]. A common factor for the inclusion of TL is that it can increase the performance of a given network, while also not requiring a large dataset to achieve said level of achievement.

9.1 Need in an Annotation Environment

A trend between different applications of TL is that it is used to take a pre-trained model, like those trained to detect cars, for example, and its knowledge learned for another task, like recognizing trucks or boats. However, this is not the only method in which TL can be used. For the use case of automatically detecting user-desired pen modes, we leverage TL to help overcome the performance challenges previously noted. In particular, it was found after the evaluation study that the annotations made by users were specific to the individual using the pen. For example, some participants underlined closely to the bottom of words, while others placed more distance between the word and the underline. A subset of participants went so far as to write closer to the top of the words below. This finding is backed up by some prior research which has also found that writing styles are different between people when using a traditional pen and similarly can be seen with a digital stylus [48]. This user-specificity, combined with the smaller dataset collected, caused great difficulty for the neural network when attempting to create a generalizable solution.

The ungeneralizable nature and small dataset of my annotation task created a perfect scenario for the inclusion of TL. Namely, it allows for a pre-trained "base" model to be further trained with a small amount of user data, effectively tuning the iteration of the network towards that user. The main benefit that comes from this model tuning is that the network can allow for more nuanced annotation styles and interactions to be learned on a per-user basis, which would've likely caused noise within the previously generalizable iteration of the network. Learning these nuanced interactions can help in particularly ambiguous situations where the model might have previously returned an incorrect classification, therefore improving user interaction.

9.2 Performance Evaluation

An evaluation was conducted to determine the effect that TL has on the model. Using the original model created in Tensorflow (Chapter 4), the set of annotation data collected from both studies was used as training data — with the caveat that one participant's data was held out for evaluation purposes (called holdout data). Once training was completed, the model was tested on the holdout participants' data, followed by another TL training sequence. The TL training sequence utilized TL by extracting the initial model's weights for each of its layers and using them within a new, participant-specific model as a basis. Both of the training sequences (initial training and TL) followed the same training procedures as mentioned previously, with the `EarlyStopping` Tensorflow callback used to stop the training if the model's loss stopped improving significantly. When undergoing additional transfer-learning training, a split of 20:80 was used for the training and validation data. This data separation change mimics the extreme data limitation found within the annotation environment, evaluating the effectiveness and practicality of the TL training. Therefore, the standard ratio of 80:20 [15] was reversed to provide the more training-data-restricted ratio of 20:80. A resultant issue encountered within TL training was the model's tendency towards overfitting, something common with small datasets. A lower learning rate of 0.0001 accounted for this noticed overfitting.

As shown in Table 9.1, across 27 participants, most were found to increase classification performance after TL training. The highest recorded increase was from participant #9, from 72.84% accuracy to 86.71%, an improvement of 13.88%. The average performance increase across participants was recorded to be 13.26%. While impressive at increasing performance for most participants, some were found to have performance regressions when using TL. This decline was due to some participants making a few irregular strokes, which the model had not seen before and did not have enough training data to learn effectively. We expect that, when implemented, the model will be able to learn these unique annotation patterns if given enough exemplary data.

Participant ID	% Before	% After	% Change
1	86.41	90.83	4.43
2	85.95	89.26	3.32
3	93.79	95.08	1.29
4	90.37	94.88	4.51
5	89.89	94.20	4.31
6	74.76	85.62	10.86
7	89.54	94.18	4.64
8	89.10	92.01	2.91
9	72.84	86.71	13.88
10	94.16	94.83	0.67
11	89.05	91.40	2.36
12	87.00	87.67	0.67
13	82.13	81.55	-0.58
14	94.96	95.90	0.94
15	72.06	76.50	4.44
16	94.18	97.02	2.84
17	90.65	93.14	2.49
18	94.55	96.52	1.97
19	91.57	93.27	1.70
20	78.20	91.60	13.41
21	93.14	95.23	2.09
22	96.30	96.75	0.45
23	87.59	91.81	4.22
24	68.82	70.18	1.36
25	92.75	93.93	1.18
26	94.25	96.03	1.78
27	94.42	95.08	0.66
		Average:	3.44
		Std:	3.60

Table 9.1: The evaluation accuracy of participant data, before and after transfer learning. 20% of the participant's data is used for transfer learning, with the remaining 80% used for evaluation purposes. The model is trained with all other participants' data before transfer learning and evaluation.

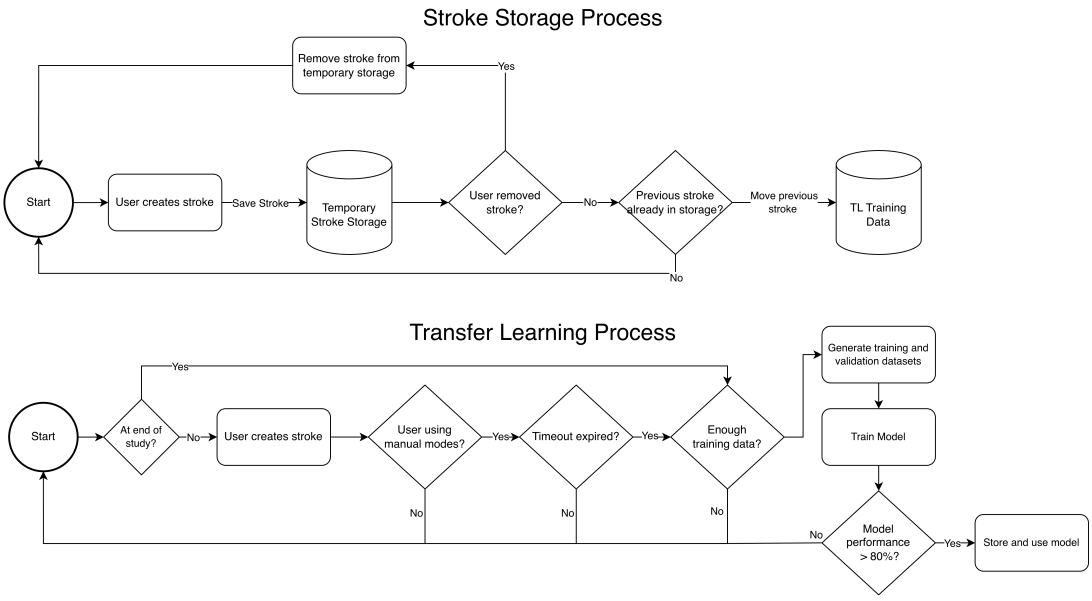


Figure 9.1: A flowchart visually outlining the process used for stroke data collection and TL procedures.

9.3 Design and Implementation

With the performance benefits seen due to the introduction of TL, the next step came to implement it within an annotation interface. Initially, the system was planned to be implemented such that the model would train in the background autonomously, provided that it had enough stroke data to begin with. However, this idea would bring three challenges relating to data collection within the annotation setting. Specifically, these challenges are as follows: (1) When do we collect stroke data? (2) When do we train on said stroke data? (3) When do we propagate updates? Taken together these problems can be centered around a singular question: "What and when do we train?"

At first glance, the solution to the data collection problem is easy. Stroke data could be captured after each stroke is created and then used to train the model. However, in practice, it was found that this technique caused the model to be trained on incorrectly-moded strokes, which coupled with TL, caused lower performance and incorrect stroking techniques to be learned. This challenge is difficult because we can never be sure that a stroke's mode is correct since the user can remove it at any time. In addition, users may even remove strokes that are correctly moded but do not meet their expectations. An assumption is made to solve this issue: if a stroke is in the incorrect mode, a user will remove it before making another stroke. This assumption was based on an observation from pilot runs of the preliminary evaluation study (and in testing of the refined mitigation method) where users were found to remove incorrectly-moded strokes

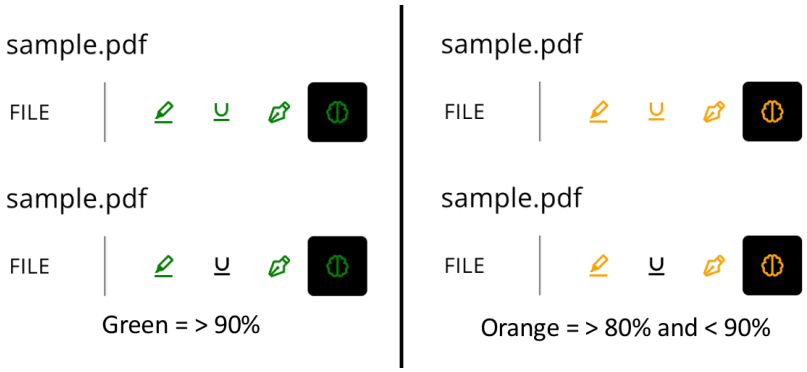


Figure 9.2: An example of the Magic Pen mode button changing colours based on model performance. In the top row, the model was trained on all three pen modes, while the bottom was only trained on two, signified by the colouring of the mode icons. Orange denotes performance between 80% and 90% while green denotes performance greater than 90%.

shortly after creation before making another stroke, even when using the manual modes with the toolbar. With the stroke removal constraint set, we record data from strokes that have not been removed after a new stroke is created, as seen in Figure 9.1 (top).

With the data collection issue resolved, the next problem was focused on when to take the data, and to learn with it. With the goal of having an always-learning model in mind, an immediate problem was found when it was discovered that classifications could not be made while the model is learning. This was largely due to performance issues. Most pen-enabled devices are relatively weaker-performing tablets or laptops, so training the model while making classifications caused many of those same classifications to be delayed, which was not ideal. To combat this challenge, the reasoning for training the model was investigated. The main driver for re-training is the performance improvements it brings, but what if there is no need for extra performance at the given moment? With this consideration in mind, model training was relegated to only occur at the end of an annotation session, when users were asked to not actively annotate. However, this delay does not aid in the event that the user would actively benefit from retraining. So, to provide as many opportunities to teach the model as possible, background training with stroke information recorded from both the automatic and manual systems was implemented when using the manual pen modes with a toolbar, shown at the bottom of Figure 9.1. A timeout of 10 seconds before TL training begins after annotating with manual pen modes was implemented as to prevent accidental training when wanting to solely annotate. In addition, training was made to not be triggered until a minimum of 500 stroke windows (roughly 10-20 strokes) were recorded. This limitation was done to ensure that the dataset used would capture a variety of different annotations.

The third and final challenge with the implementation of transfer-learning within the annotation environment is determining when to propagate (or keep) model updates while also alerting the user to the model's performance. This challenge is two-fold: in the event that the model's performance drops after training, we don't want to retrain immediately due to performance limitations, but we also don't want to use an error-prone model. To prevent the user from using a low-quality model in this case, new model iterations are only kept (and made available) if their performance is greater than 80% on a validation dataset of the user's own stoke samples. This removal of model iterations was done since the previous TL experiments demonstrated that the ML model was capable of learning annotation behaviours with high accuracies, with lower accuracies suggesting an unstable model. However, due to the variances in model quality introduced, it became evident that a method to alert the user to the model's performance became necessary. However, the challenge in alerting the user is that we do not want to paint the model as unappealing, while also describing that performance may be less than ideal. To achieve this level of model visibility, the colour of the automatic mode method's button on the toolbar changes (see Figure 9.2) between yellow (performance between 80% and 90%) and green (performance greater than 90%). Yellow was chosen to avoid the good/bad dichotomy that green and red typically signify in Western regions of the world.

10 Magic Pen Annotator Software

Once the previously outlined changes were thought of, it became evident that a testbed for annotation interactions and features was needed. More specifically, I needed a platform that allowed users to perform "real annotations" (i.e. not forced or faked) with the user's Portable Document Format (PDF) documents. With this intention in mind, the Magic-Pen web software was created (see Figure 10.1).

10.1 Design

Annotation Flexibility

To support the goal of allowing for "real annotations," a system that allowed users to upload their documents was made necessary. More specifically, we wanted a system that users could upload their documents to, annotate freely (using the automatic systems), and then download their strokes merged with the document. PDF documents were chosen as the file format for the uploaded documents due to their extensive use within other commercial annotation software. However, while the document could be uploaded with ease, all of the annotation software created was designed to draw on top of web elements (SVGs), meaning that some form of conversion between the two was necessary.

The challenge in converting between the inputted PDF format, and the required web format for annotation, was that PDFs are not inherently compatible with the SVG standard. While possible to include SVG elements within PDF documents (which can easily be converted), characters and their positions are represented specifically with text elements, making the transition to a web-viewable standard difficult. Luckily, a few open-source projects can perform this conversion, such as MuPDF. MuPDF is a document viewer capable of opening PDF documents and features a highly-accurate conversion library, allowing for the conversion between PDF and SVG documents with minimal graphical errors [32]. Due to its high performance, this library is used on the web application's server-side, converting inbound PDF documents into SVG strings that the application can parse.

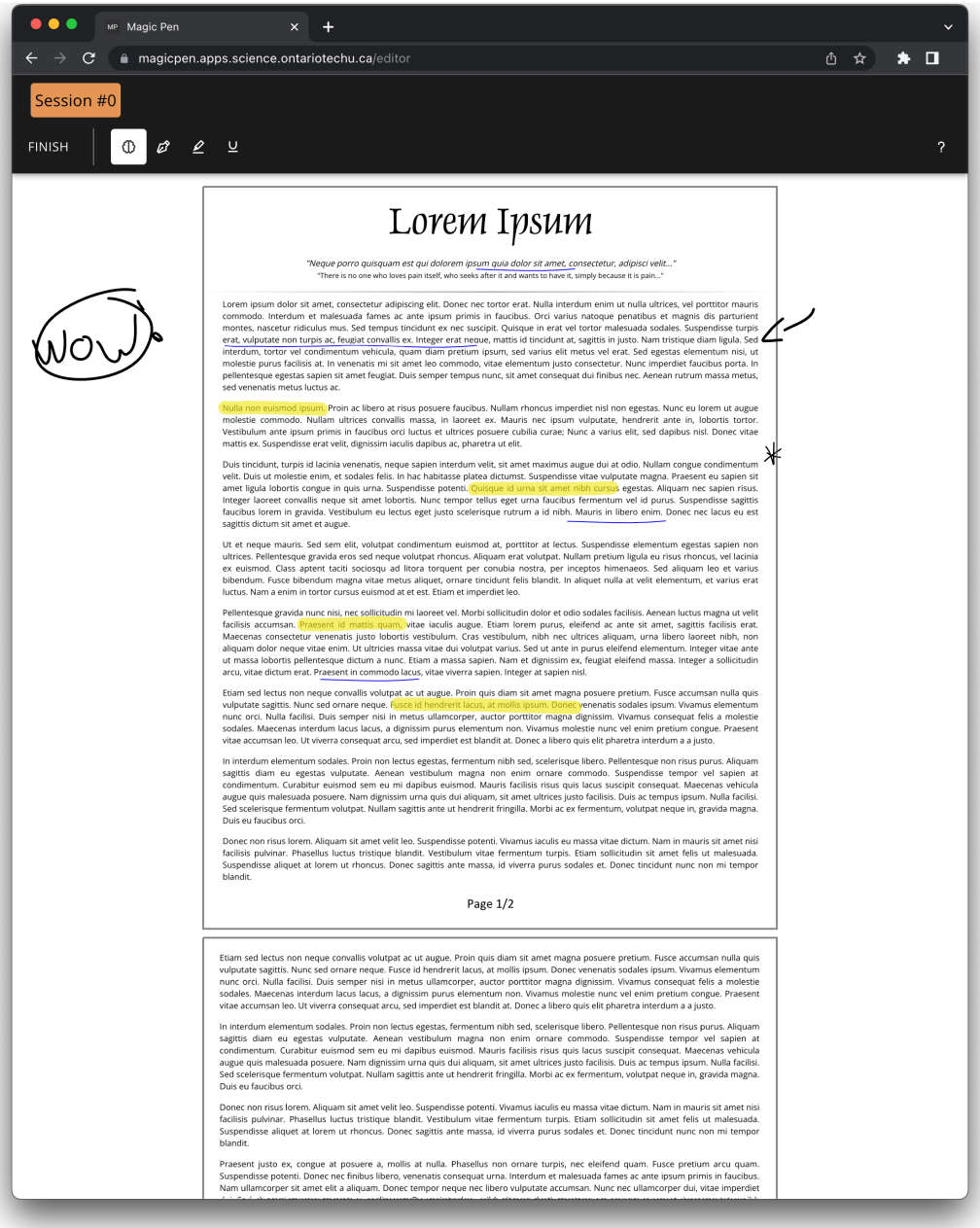


Figure 10.1: All annotations displayed on the document were made using the Magic Pen automatic mode technique.

Modes Available

In attempts to give users ultimate flexibility in which interaction methods they wanted to use, a toolbar was implemented. Also, as a means to also provide access to the smart method, a toolbar button was used. The idea behind this implementation of a toolbar was that it was intended to be directly interacted with infrequently unless otherwise desired. For example, status icons, such as the progress of the model's training, would be relegated to this toolbar alongside study-specific status messages. A toolbar was chosen since users were likely to have had extensive experience with one and knew to look there first in case they got lost.

While the use of the 'smart method' was enabled by default, an option to immediately access it, or switch to other methods, was made accessible in the interface's toolbar. The reasoning for this decision was the work by Negulescu et al. [34], which when evaluating the Inferred-Mode protocol (by Saund and Lank [47]), found that utilizing a dedicated button for the automatic inferencing could aid in users' discovery in the tool that was not previously instructed. This discovery could also aid in the understandability of the technique, which was a limitation of the Inferred-Mode protocol.

Stroke Inferencing Pipeline

Once we decided to leverage transfer learning for the annotation model, the drawbacks of the previous classification pipeline became more evident. Namely, there are two primary reasons why the architectural reliance on a network server to receive and return classifications did not work well. First, the network-based approach was reliant on having a fast and stable internet connection. While the data usage is relatively low, the latency in the network connection was found to cause a delay in classification performance; lowering trust in the technique. In addition, the network approach was found to drop classifications from time to time which also affected performance. With the use of an on-device model, the personalized iterations of the stroke model were able to be computed locally and more efficiently which reduced the overhead and previously-mentioned latency needed to classify strokes.

Secondly, the server-based approach did not allow for user-specific models to be stored easily. This required a database and a unique user identifier to store and retrieve different iterations of their personalized model. In addition, raw stroke information must be fed in to allow the model to be transfer-learned, which may consist of sensitive data. Therefore, potentially sensitive data would be required to be sent online to the server for computation; posing a privacy concern. The move to an on-device model can be seen as beneficial in terms of privacy, as the on-device storage does not require the uploading of user strokes. This ensures that a user's strokes never leave their device. With the mentioned disadvantages (and the benefits of the on-

device method), the Web Annotation System was made to reference a local, on-device model instead of a network-based one.

In order to leverage on-device ML, Tensorflow.js (TFjs) was used. TFjs is a branch of the popular TF library, allowing machine-learning models to be developed, trained, and run using JavaScript [52]. The previously created stroke model was converted using this framework so that it would be stored and run locally on the user's device. A network model is only loaded upon the first load of the Web Annotation System; otherwise, the local model is used for inference and training tasks. A benefit to using TFjs is the availability of GPU compute (through WebGL) within the web browser environment, which offers the potential of greatly improving training and classification performance. However, further testing found that this acceleration caused classification delay issues with the first stroke drawn in a session. This oddity could not be avoided by pre-loading the model beforehand as well. Due to this delay induced, GPU model computation was disabled, and CPU computation was used instead. Another added benefit to utilizing an on-device computation model is privacy. Since the need to send stroke information to a remote server for classification was removed, user annotations were enabled to be kept on-device, allowing the system to be used for privacy-sensitive annotations (outside of a study environment).

The updated model training pipeline, which utilizes an on-device model, is now as follows. Firstly, data is collected in the same manner as before. Then, in a similar fashion to the previous pipeline architecture, a training request that features said data is generated and sent to the model; if enough data is obtained. However, the main difference in this step can be seen in the receiving end of the request. An on-device model now listens for and responds to the request instead of a server. The model will then train in the background until complete. If the performance of the current training process is adequate, the model is stored for future use, and the training data is moved to be used as validation data. Otherwise, the model iteration is discarded.

10.2 Speed Considerations

When the initial implementation of the annotation software was created, it was found to be extremely slow. Between slowdowns when uploading PDF documents to severe lag when annotating, it was clear that improvements had to be made to improve the overall end-user experience. Extensive research was conducted to determine and solve performance degradation sources. These findings are reported in this section to aid future software development.

10.2.1 PDF Rendering

Early on, within the development of the annotation software, it was found that the use of Scalable Vector Graphics (SVG) was not feasible for the task of uploading PDF documents. This conclusion was that application performance dropped significantly when loading documents that contained five pages or more or when creating long strokes. This performance degradation was caused by a combination of high-detail text elements and interactions with said text elements. This performance problem became necessary as the overall user experience became sluggish to use and directly affected the inferencing performance of the ML network.

When using MuPDF to convert PDFs to SVGs, text elements are converted by creating a Path element and ”tracing” the text. While accurate for re-creating various fonts, the large number of points within the Path element caused a significant load on the browser's SVG rendering engine. In addition, to prevent the redefinition of individual letters, MuPDF utilizes the ”<symbol>” tag to define an SVG path once and the ”<use>” tag to reference it throughout the converted SVG. However, it has been found that on many browsers, such as Google Chrome, perform significantly worse when using said SVG elements, therefore lowering interaction performance [21]. Therefore, to help improve the performance (or lack thereof) while rendering with SVGs, points within Path elements were simplified, and <use> tags were replaced with a redefinition of the path defined within the <symbol> tag. Also, to reduce load where possible, offscreen pages were unloaded to prevent the browser from loading the hundreds of SVG paths representing the page's text.

Another aspect of the SVG renderer that affected the application's performance was the interactions with the text elements themselves. To detect when the digital pen was over top of text elements, JavaScript functions that determined the pen's position relative to all objects on screen, such as `getBoundingClientRect` and `elementFromPoint`, were used. However, it was found that these functions were extremely costly, as browsers will re-calculate every object to determine their position relative to the mouse (or pen). This cost was made worse as function calls were needed on every pixel movement of the pen. Calculations made from these function calls were promptly cached as re-used throughout the application to reduce the slowdown caused by their use.

PaperJS

Despite previous changes made to improve performance, limitations with the SVG rendering method still affect the end-user experience. Although lessened, it was found that performance was still low when loading documents that were ten pages or more (an increase from the previous five) or when loading content-heavy PDFs. Therefore, it became evident that a renderer

that used the browser's SVG rendering method became unusable — pushing for an alternative. When looking for another renderer, key emphasis was placed on speed and flexibility, leading to the use of PaperJS, a JavaScript library.

PaperJS is an open-source vector graphics library that utilizes an HTML5 Canvas element for rendering, rather than the typical HTML DOM [40]. The main reason for its use was that it allowed for SVG elements to be imported and converted for use with a canvas, increasing performance significantly. For example, a 10-page document, which would have run at ten frames-per-second (fps) with the old renderer, ran at 60 fps with PaperJS, freeing the device's processor for ML computation. In addition, PaperJS offers several features, like Path Simplification and SVG exporting, which sped up development time and allowed other features to be implemented quickly.

Once the switch to PaperJS was implemented, I decided that a feature to download the annotated documents should be made available. Luckily, because of PaperJS, the conversion process was straightforward. First, the entire canvas on-screen is converted, using PaperJS, into an SVG. During this process, annotations on-screen are merged with the original content. Next, the SVG is converted into a PDF document using PDFKit, a library for JavaScript [41]. Redacted PDFs are also created during this conversion and are stored for evaluation study purposes. Then, the PDFs are transformed into blobs (groups of bytes that hold file data) for ease of storage and download. Finally, the user's PDF copy is downloaded to their machine.

10.2.2 Transfer Learning & Inference Speedups

While the increased flexibility and classification performance improved significantly with the move to an on-device model with TFjs, an aspect of the annotation system suffered — the overall performance. Due to the single-threaded nature of JavaScript by default, all model training and inferences were run on the same CPU thread responsible for UI updates. If a model training session was for an extended period, the UI could be unresponsive. An easy solution to this issue would be to run the model on a GPU, as is standard with traditional ML frameworks [1]. However, GPU computations for TFjs require a browser feature titled "Off-screen Canvas", which is only natively supported on Google Chrome and Chromium derivatives (meaning no experimental settings need to be enabled) as of writing [38]. Even when using Google Chrome, it was found that the GPU added a delay to stroking performance — likely due to the overhead of copying the model to GPU memory.

With GPU computing being unreliable due to decreased UI responsiveness when performing model-heavy tasks, an alternative method to offload the computation from the main JavaScript thread was explored. JavaScript Web Workers are a means for web content to run scripts in background threads [17], and offered a promising solution. Web Workers were set to run the

TFjs model exclusively in this use case, freeing up the main JavaScript UI thread. Communications with the model also changed minimally, with messages containing data being passed into and out of the thread. This messaging behaviour is nearly similar to how the network classification requests were implemented previously, simplifying their implementation. Overall, with Web Workers, TFjs model training and classifications were able to be run simultaneously with the UI, allowing for implicit interactions.

11 Second Evaluation Study

To measure the impact that TL has had on automatic annotation, and to determine if it is preferred to a standard toolbar, a second evaluation study was conducted. The study collected log data, similar to that of the previous evaluation study. The study was approved by the institutional research ethics board (REB FILE #15755).

11.1 Study Design

Participants:

Similar to the previous study, the criteria for eligibility within the study focused on having some form of experience with digital pens in the past and owning a Windows-based device to use for the study. However, participants from varying experience levels were admitted to the study, ranging from those with novice-level experience to self-claimed pen experts. The reasoning for this range of inclusion was to record whether experience affected the performance of the Magic Pen approach and if it would interact with end preference. All participants utilized a digital pen that featured an eraser on its end. In total, 8 participants aged 18-29 (5 male / 3 female) were recruited, all right-handed. Recruitment was also done through the university email lists to students, faculty, and staff, with an open invitation for students to invite others. Although strictly intended to be limited to new participants, there was potential for previous study participants to contribute, as identifying information for previous participants had to be removed per REB guidelines. However, the likelihood of said behaviour is low upon reviewing current participant information. Compensation of $10 was given per session, equalling $50 over 5 sessions. Compensation was given in the form of Amazon eGift cards.

Setup:

The study was conducted through a website that participants visited on their own devices on their own time. Multiple sessions were used over a two-week period in which the user could access the site any time they wished, as long as a minimum of 5 sessions were completed. The study consisted of screens that asked the user to annotate a displayed document. Once com-

pleted to participant satisfaction, a questionnaire was then presented. Participants were asked to complete a session in one sitting of approximately one hour. Due to the study being also conducted remotely, device orientation was similarly impossible to control. Instead, the participants were asked to keep the device in a comfortable position and to try to remain in that position for the duration of the study. All study screens are also provided in the supplemental materials.

Tasks:

The study was structured to consist of similar annotation tasks found within the previous evaluation study. However, the main differentiating factor can be found in the documents annotated. Within the study, the participants are instructed to upload their own PDF-documents and to annotate normally while using the pen modes made available. The pen modes offered consisted of the automatic method, and the manual modes (which trained the automatic method). The reasoning behind these changes comes from the results from the previous study, where participants were found to prefer the Rule-based system over the ML alternative due to performance issues. In addition, the use of real-user documents allowed for authentic annotations to be recorded and investigated, as opposed to the previous proxy which instructed participants to annotate with a certain mode at a certain place. However, in the event that participants did not have documents to annotate, which was particularly likely for the less-experienced users, supplied documents were provided that featured instructions similar to that of the previous study.

While providing documents to participants allows for those who do not have a document to participate, there were some challenges in creating said sample documents. One of these challenges was determining which task to assign the participants. The reasoning for this difficulty was that, as found in the previous evaluation study, the strokes recorded were not believed to be the utmost authentic due to the use of task proxies. However, for sample documents, a task must be given otherwise the participants will not know what to annotate and be confused; potentially leading to less authentic annotations. A series of documents were created asking the participants to edit a series of paragraphs to solve this challenge. The generated documents only describe the type of errors that will be found and ask the user to correct them using whichever method they want instead of instructing which pen modes to use and where. This new task allows the user to perform a series of annotation tasks on a document while also not instructing them on how to annotate.

The second challenge in providing user-supplied documents was finding documents with enough errors for the participants to correct. Originally a corpus of high school student essays with errors was intended to be used but was avoided due to the granularity of a majority of the

#	Question Criteria	Question
1	None	The system predicted my actions correctly.
2	None	I had to correct the system frequently.
3	None	Results from the system were expected or predictable.
4	None	The system was efficient to use.
5	None	What environment did you use the tool/system in (e.g. at a table, on a couch)?
6	None	Were there any modes that worked better or worse than others?
8	Final Session	The system improved itself over time.
9	Final Session	I found myself more distracted when using the system compared to a toolbar.
10	Final Session	I found myself frequently frustrated with the system.
11	Final Session	The system required more effort than a standard toolbar.
12	Final Session	I would use this system to annotate documents.

Table 11.1: The questions used, and the criteria for each question being displayed, within the second evaluation study's feedback questionnaire.

errors; requiring a fairly large amount of attention to detect and solve. Additionally, the public study was not meant to focus on participants' grammatical and spelling mistake detection capabilities. This could have confused participants and limited their participation. However, a means for finding documents which had errors that were fairly easy to notice was required — therefore, a script to generate such documents was created. The script was made in Python, and upon being given a document (such as a standard essay found online), the script would replace random words with either:

`Common Misspellings:` Words would be replaced with their common misspelled counterparts [31].

`Randomly Capitalized Words:` The same words but with random letters capitalized.

`Random Spaces:` Change in pressure applied to the screen since the last sample.

A threshold of 2.5% of words in a document was found to be a good balance between generating random words and keeping the document fairly readable. After creating the document generator, five high school student essays were selected (those without previous errors) and passed through the script to generate five error-prone documents. These essays were made available to participants.

The study began with the annotation of a tutorial document, which explained aspects of the study such as: the automatic pen system, how to train said system, the eraser technique, and how many sessions to complete. For the introduction session, participants were presented with an un-tuned version of the Magic Pen model, with specific annotation instructions being given

so that the model could be trained at least once. Every subsequent annotation session began with the user uploading a PDF document, which presented them with the main annotation interface. From there, participants would annotate as they saw fit, with the restriction made that they made 10 unique, somewhat long, strokes so that the model could be further trained. A questionnaire was made available after annotating (see Table 11.1), which featured Likert scale questions similar to the previous study. While the survey is being filled out, the model is trained in the background, unknown to the participant.

These steps, aside from the introduction session, were repeated for each session throughout the two-week period for a potential total of at least 8 participants x 5 sessions = 40 annotation sessions. At the end of the five sessions, participants were given the option of completing an exit survey to complete their participation in the study. Participants were allowed to use the system to annotate as many times as they desired within the two-week period, with the only restriction that they complete the exit survey during their final session.

11.2 Study Results

Except for P4, who lost a session's worth of data, all participants took roughly 12 minutes to complete each study session, which included marking a document (see Appendix Figure A.1) and answering questions. Of all the participants, only one provided their own, with the rest using the provided sample documents.

11.2.1 Introductory Session

As a part of the first session (session 0), the participants were asked to annotate an introductory document that included overall study instructions and information on using the Magic Pen and mitigation systems. This session acted as a tutorial for participants. Participants were instructed to annotate in a specific method similar to the previous evaluation study, with an initial version of the ML model. As a result, the session acted as a method to evaluate the model's capabilities before any TL took place. Figure A.2 summarizes the responses to Likert questions across sessions. Note that the scales for *Corrected Mode*, *Distracted*, *Frustrated*, and *More Effort than Usual* have been reversed so that higher values are better for all rows, similar to the previous study.

Overall, all participants found the model accurate, with one strongly agreeing, resulting in an average score of 4.375 (standard deviation of 0.48, see Appendix Figure A.2a). The participants also felt that the model was efficient and predictable in their usage, netting both average scores of 4.5 (std of 0.71). Only two participants did not strictly agree in both cases, resulting in scores of 3. The base model's improvement can also be seen in the stroke correction question,

with 7 participants indicating that they did not need to correct their strokes post-creation, netting an average score of 4.375 (std of 0.70). Although the sample size is small (8), the results demonstrate that the overall technique and the base model before TL likely improved since its last iteration.

To further determine whether the base model has improved since its previous evaluation, I investigated the number of strokes created and stroke mode changes and removals (i.e., stroke edits). It was found that across all strokes from all participants during session 0 (see Appendix Figure A.3a), an overall average of 62.5 strokes were made, with 76 total strokes edited. While the overall number of strokes was low, the result was significantly skewed by P4, who made an overall 141 strokes. Of the strokes created, P4 removed 40. This finding is opposed to the other participants who made roughly 30 correct strokes each, except for P5, P7, and P8. The said participants made annotations which consisted of writing out entire words, inflating the overall stroke count. Upon investigation, it was discovered that the stroke increase with P4 was due to them simply making more strokes overall (which was not required as a part of the tutorial). However, of the removed strokes, the majority were intended to be underlined, which matched P4's qualitative response of, "underline mode worked worse than the others, it constantly recognized it as highlights". This sentiment towards the underline classification performing the worst was common amongst participants, as seen during initial model performance evaluations (Figure 4.4). Despite this skewing of data, the overall ratio of stroke edits to strokes made was fairly low among all participants in session 0, suggesting reasonable base model performance. It should be noted that the stricter nature of the instruction sessions could have affected the model's performance. The instructions were similar to previous studies, which were found not to be the most accurate proxy for natural annotation.

11.2.2 The Impact of Transfer Learning (TL)

Participant questionnaire scores were evaluated over the various sessions to determine the impact of TL on the annotation experience, aside from session 0, since it used the base model. As seen in Appendix Figure A.4, the ranking results show either a net positive or neutral trend as the sessions continued for the question relating to perceived model accuracy. All participants had a noticeable decrease in perceived accuracy during one or two sessions but had their scores either level off or improve by the end of the final session. The behaviour suggests that the models improved overall performance, yet the raw performance information of the TL suggests otherwise. By the final session, most participants had a model accuracy decrease of 5% on average compared to their first TL session. However, it was discovered that the accuracy metric could not be used to fairly compare sessions and participants since both the annotated documents and the validation data used varied between sessions and participants. In addition,

a probable cause for the slight degradation in raw performance is the training and validation dataset used. It was discovered that the earlier sessions used datasets that were often magnitudes smaller than those used at the end (see Appendix Figure A.7). As a result, these smaller datasets limited the amount of new information that could be fed into the model and made it easier to achieve higher accuracy values overall. This behaviour possibly explains the significantly higher accuracy metrics, with the evaluation getting harder as the sessions continued.

Despite the model's raw TL performance, all but one participant ranked it either a 4 or 5, showing that it did not affect users' perceived predictability of the technique. An example of this behaviour can be seen with P1 in Appendix Figure A.4a, where they were not able to predict the model's results initially but recorded a linear improvement (from a 1 to a 5) by the final session. While not a clear proxy for accuracy, the previous evaluation study demonstrated that model predictability could be related to overall technique accuracy due to the user's understanding of how to make the technique serve them best. As a result, I believe that the predictability metric might signify an overall model improvement. This claim is backed by additional textual responses from participants, which point to the method improving over the sessions. For example, in response to the question, "Were there any modes that worked better or worse than others?" P1 initially said *"Underlining seemed to be less accurate"* and *"Highlight was right the most, while I felt like I had to correct underlining the majority of the times I tried to use it"* during sessions 1 and 3. In comparison, during sessions 4 and 5, they said *"All worked well, underline and highlight sometimes got confused, but that was usually my fault"* and *"No, all worked well"*.

Inspection of the uses of the manual methods revealed some interesting trends amongst participants. As seen in Appendix Figure A.3, participants 1, 3, and 5 made extensive efforts to invoke the model training, with some of them making more strokes in the manual mode than in the automatic approach. While initially confusing, examining the participants' annotated PDF documents revealed that for the majority of its use, users only made a single type of stroke with the manual method. The problem with this approach was that, due to mechanisms preventing model overfitting, the model would not train in events where the training data is severely skewed. This behaviour effectively stopped the model from TL for participants, aside from when it trains at the end of sessions. While dialogue popups in the toolbar were used to alert the user in this specific event, it was clear that they did not help participants in practice. Of the 13 manually-invoked training sessions that were allowed to continue (due to having a properly balanced dataset), eight were not allowed to be used due to their accuracy being lower than 80%. Two failed training periods resulted in accuracies lower than 60%, four between 60% to 70%, and two more within the above 70% range. This result suggests that the accuracy percentage minimum for models should be set lower to account for unique stroking interactions. Despite the failed attempts to include training manually, the automatic TL periods at the end of

the sessions were found to have high accuracies overall, with only four being lower than 80%. This result suggests that when invoked manually, the participant will likely create error-prone strokes for the model, meaning that training restrictions should be removed.

As previously found within session 0, there seems to be a commonality amongst participants in mode errors, with the underline mode struggling the most. 7 out of the 8 participants complained about the model's performance with the underline mode and mentioned how well Draw/Highlight worked at the end of the first session. Interestingly, the difficulty for the model seemed to fade as the participants continued through the sessions, with seven participants mentioning that all the modes worked similarly by the second-last session. This drop in mistakes can be seen in the number of stroke corrections needed over the sessions by participants (Appendix Figure A.3). On average, the number of strokes edited for each participant through sessions 1 to 3 was near 15.3, dropping to 6.125 between the last two sessions. Two participants with significant errors were P1 during sessions 1 and 3 and P5 during session 1, with the number of correct strokes nearly equalling those removed. It was found that for P1, the main cause of errors was the documents used. For session 1, a document with tight line spacing was used (see Appendix Figure A.8, a known issue (see Chapter 4) which causes a significant amount of underlines to be in the middle of words. For session 3, the document used did not contain text elements (see Appendix Figure A.9, white page in the redacted view), and the model could not pick up on word-level features. However, despite not having all the features and producing somewhat ambiguous results, the model was still able to correctly classify the pen modes occasionally, demonstrating the potential strength of the model and TL. As for P5, most of the removed strokes were seemingly correctly moded (see Appendix Figure A.10), suggesting that the participant removed them for other reasons. The drop in corrections could be another proxy for overall model performance, signifying fewer errors were made throughout the study. As a result, I believe the number of stroke corrections made might also signify overall model improvement.

11.2.3 Final Session Preferences

Likert responses from the last session were compared to determine user preference in terms of Magic Pen (see Appendix Figure A.2f). As a result, the Magic Pen technique was slightly preferred among participants, with an average score of 3.875. Curiously scores for all other categories were all positive or neutral with the exception for *Distracted*, *Effort*, and *Frustrated*. These lower scores were largely due to P1 and P3, who did not prefer using Magic Pen. They were compared to the other qualitative metrics to determine a cause for the lower preference metrics by computing the Pearson Correlation Coefficient. The results showed that of all the other qualities asked about in the questionnaire, none impacted the overall *Preference* scores

as much as *Frustration*. Between *Preference* and *Frustration* a R-value of 0.7841 was obtained, with a p-value of .02126 making the result significant at $p < 0.05$, lining up with the *Frustration* scores for P1 and P3. Despite feeling frustrated, both participants noted that the system actively learned throughout the study and was perceivably accurate by the final session. However, as seen in both Appendix Figures A.4 and A.5, both P1 and P3 only felt accurate and predictable with the system near the end of the study, and P1 had a significant amount of issues with documents uploaded earlier, signalling that their overall Frustration might stem from their earlier interactions. When P1 is removed from the study due to issues, the overall preference average becomes 4.0. While the other participants disagreed on the *Frustration* aspect, they all shared the same scores for the other questions, such as *Corrections Made*. These results demonstrate that overall technique preference, while ultimately dependent on model performance, can depend on several other factors, such as TL and the time needed before the model is suitable for their use.

The Magic Pen system's overall preferences align with the previous study's final suggestion. Implicit annotation interfaces should offer an option that is both predictable (therefore easy to understand) while also being performant not to frustrate the user. As noted within the previous study, and with the current model predictability metrics, there is reason to believe that the use of TL allowed the method to become more predictable or understandable to users as the model was iteratively geared towards them — improving their performance as a whole. While concrete assertions cannot be made due to the limited amount of participants used in this study, there is a noticeable improvement to the perceived user experience throughout the use of the model due to the use of TL. I believe there is a rationale further to investigate predictability and understandability within an implicit annotation environment.

12 Conclusion

This chapter discussed the contributions, work limitations, conclusion and some ideas for possible future work directions for this research.

12.1 Contributions

One of the contributions is the Machine Learning model that innovates by allowing dynamic implicit mode-switching to improve user experiences when using a digital pen (or stylus). The classification pipeline balances stroke correctness and latency by automatically classifying strokes near the beginning of their creation while filtering out potentially-incorrect mode changes. This pipeline allows for intuitive mode switching, which is both easy to use and predictable in its execution. Interaction designs have also been outlined, which detail the challenges and considerations one must make when implementing an implicit system for use with a digital pen. The combination of the ML model and the interaction designs allow researchers to experiment within the field of implicit pen-mode switching without the need to discover these insights from the ground up.

Furthermore, the use of Transfer Learning (TL) within an annotation environment was experimented with to improve implicit interactions. The TL implementation and annotation-specifics were outlined to explore how TL could be used in an implicit manner within an annotation setting. By training in opportune moments with captured user data, the model was able to iteratively improve and be geared towards users, improving their performance and the understanding of the overall approach.

An explicit eraser interaction was also designed and implemented based on the finding that users would remove their strokes and redraw them if they were moded incorrectly. The eraser-mitigation techniques require some form of non-distracting gesture to change a stroke's mode (or to remove it entirely), allowing users to focus on their work and preventing frustrations with needing to correct stroke mode errors. However, while experimented with throughout both evaluation studies, a further experiment is needed to determine if the methods provide the intended benefit.

An annotation environment was designed and created that pairs extensive stroke logging with a traditional annotation settings, with the added benefit of running on the browser and being able to annotate user-supplied PDF documents. This test-bed allows for different augmentations to annotation interactions to be recorded and experimented with quickly, enabling information discovery. With the use of custom user PDFs, more natural and comfortable interactions are able to be recorded, also furthering discoveries. A series of performance considerations have been included as a part of the environment's documentation, which may help others overcome specific challenges when creating an annotation interface.

Finally, two evaluations and one data collection study have been prepared to record and evaluate the effectiveness of implicit pen-mode switching. Overall, the studies have discovered nuances with implicit annotation interactions such as users' preferences towards understandability versus raw performance. In addition, specific implications for TL within an annotation environment have been outlined including the need for a method that can allow for both automatic and intentional training.

12.2 Implications for Future Research

In addition to the various contributions made to the field of digital pen annotations, several items related to the design of annotation interfaces and studies were also discovered through the execution of the various studies. It is believed that these implications could be used to help design better exploratory studies and interfaces that utilize a digital pen.

12.2.1 Data Collection Study

While preliminary in terms of the study format, there were some significant findings in running the first data-collection study. The first noticed behaviour was that due to the mix of experience levels with a digital pen, there were differences in how quickly participants adopted the digital pen. It was found that, as expected, more advanced users could adjust to different annotation environments and techniques than previously used. This byproduct of the digital pen experience could be vital to the level of training provided to participants and can directly affect the recording of natural annotations.

Another factor that affected the recording of natural annotation styles was using "fake" documents to annotate. "Fake" in this case refers to documents that the user did not own. As a result, these fake documents were likely less interesting to said users. These documents produced unnatural annotation data when used in an annotation environment with specific instructions. This uncommon collection of features can adversely affect the design and creation of pen-mode switching techniques, as they were found not effectively to convey the interchange

of pen modes. For example, instead of recording strokes that changed modes frequently (e.g. a highlight stroke followed by a draw stroke), strokes in similar modes followed each other like blocks (e.g. highlight strokes, then all draw strokes).

The recording of natural annotations was discovered to be affected by another aspect of the study: device orientation. The device used (a Microsoft Surface Pro) throughout the study was taped flat to a table. While effectively restricting the number of unknown variables within the annotation dataset, this restriction also causes the recorded data to be slightly inaccurate. Users have varying preferences for device orientation during annotation, and the change in this orientation can significantly affect the features recorded. By not allowing the device to be adjusted toward the specific user, the quality of the recorded data could be adversely affected.

12.2.2 Preliminary Evaluation Study

With the introduction of the preliminary evaluation study, much of the study structure was changed to better support previous discoveries. However, the study discovered new implications for digital pen evaluations. Firstly, while restrictive instructions were altered to be more open-ended (in an attempt to allow for more dynamic interactions), mode-specific instructions affected the recorded annotations. When given directions like "highlight ten sentences", some users were found to perform the instructions in a linear fashion, similar to before. It is thought that using a specific number causes participants to complete the tasks in a linear order. While this behaviour is superior to that used in the previous study, there is room for improvement in collecting authentic notes.

In terms of the mode-switching techniques, one noticeable effect was the importance of algorithm understandability amongst users. As noted previously, users were found to prefer the Rule-based technique despite its lower accuracy. This adoption was due to the easily-understood nature of the method, causing users to adopt their annotation styles to that of the method expected. This emphasis on understandability and predictability could be used to better design and create digital pen interfaces and techniques, which will boost user engagement. The behaviour was in direct opposition to the ML method, which while performing better, was harder to understand.

Despite participants' lacklustre adoption of the ML method, its use led to a better understanding of the preference for the Rule-based alternative. More specifically, it was found that within techniques which were not as easily understood, incorrect pen mode switches (errors) were found to annoy users more than the alternative methods significantly. This behaviour coincides with Nielsen's ten heuristics, specifically heuristic one (Visibility of System Status) and nine (Help users recognize, diagnose and recover from errors), in which users should be made aware of any errors and be made to understand why the errors occurred in the first place [36].

An increased level of understandability with an annotation method can potentially prevent this frustration and help users recover from errors.

12.2.3 Secondary Evaluation Study

The discoveries made from the previous studies were implemented within the third overall study with the main goal of supporting natural annotations to be recorded while also evaluating the effectiveness of the pen-mode-switching methods. While the study effectively allowed for data closer to natural annotations to be recorded, there were a few more lessons learned throughout the study. Primarily, it was found that despite the option to upload user-supplied PDF documents to annotate, users preferred to annotate sample documents that were supplied to them. While the exact reasoning for this choice is not entirely known, it is thought that the users were either afraid of having the contents of their documents online (participants were informed of the otherwise beforehand) or did not have any documents ready on hand to use, since the study was run during the summer, and the participants were largely students. As a result of this behaviour, an emphasis was placed on the sample documents provided.

To better support natural annotation, the instructions of the sample documents were revised. Despite using the same instruction format as the previous two studies, the required numbering of annotations was changed to ask users to annotate errors however they liked. This change required users to read the document and annotate simultaneously, causing users to annotate more naturally. An added benefit was that users, even those less experienced with pen technologies, had fewer questions about the specific annotation types required.

The final behaviour noticed amongst participants was that the manual method training mechanism was primarily used to correct strokes that the model had previously misunderstood. This behaviour is opposed to the automatic training mechanism, which trains the model to 'keep up' with the users changing annotation styles. While the manual mechanism was designed to correct incorrect strokes, a limit was placed to only allow for models that achieve performance greater than 80% to prevent error-prone models from being used. However, this strict limitation also prevented some more unique stroking patterns from being not learned due to the model's initial challenge in learning them, causing greater frustration for certain participants. A more flexible boundary for the manual method might allow better interactions to be learned and understood, improving the end-user experience.

12.3 Limitations and Future Work

Although the Magic Pen technique successfully allowed for implicit mode switches, there were some drawbacks in its design and implementation. I believe that these areas of improvement

provide an opportunity for expansion within the digital annotation research community, allowing for a future category of implicit annotation work to be created.

12.3.1 Machine Learning Prediction

To successfully infer a user's desired pen-mode based on how they utilize their stylus, a relatively small neural network based on a LSTM was used. While different machine learning methods and larger models could have been used, the computational cost and data required also grew exponentially. In addition, utilizing larger and more complex models increases the possibility of overfitting a dataset, worsening overall model performance. As far as the current situation is concerned, using a smaller ML model yielded better performance, allowing quicker training and inference times on devices tested. Nevertheless, areas within the feature selection and model preparation could have been improved; primarily, the model currently classifies between three different pen modes (Draw, Highlight, and Underline). While effective for those modes, this limitation strongly affects what can and cannot be done implicitly, which is not ideal. In future work, better feature curation and model development can allow for a more robust annotation experience with limitless mode-switching potential.

Another avenue for improvement is the use of the Rule-based model. While developed based on observed annotation heuristics, the model lagged behind the ML alternative in terms of performance. However, despite this lack of raw classification power, some users preferred its explainable mode-switching due to its stability. Sometimes simpler methods achieve more-desirable results. In future work, alternative architectures and different feature sets can be explored (such as Random Forest) to provide an understandable inferencing experience without using a neural network.

12.3.2 Model Transparency & Understandability

In the preliminary evaluation study, an observed effect was that some users preferred to use a more easily understandable technique than a more accurate one. The likely reason for this was that by understanding how the technique worked (breaking down the 'black box'), users were allowed to create a mental model of the interaction as a means to have it serve their needs (for pen-mode switching) better. In addition, it was found to annoy users less; which was likely due to the user's understanding of why the model did not perform as desired. These behaviours were only improved with the introduction of TL, which improved end-model predictability for users. Other than TL, what are some other techniques that can help break down the complexity of the neural network? In the past, extensive research has been dedicated to aiding model understandability, some of which focuses on extracting easy-to-understand rule sets [11, 58].

In future work, understandable ML algorithms can be used to provide a coherent annotation experience while also boosting inferencing accuracy.

12.3.3 Improved Implicit Interactions

In the interaction design, an emphasis was placed on inferring the user's intent without disrupting their work. For example, determining whether a stroke was correct based on whether the user created a new one. The reasoning is that we are using implicit interaction, which brings with it a goal of not disrupting the user and requiring less cognitive load than traditional techniques. Otherwise, the technique will become meaningless. However, within this work, there is an opportunity to better discover the user's intents outside of annotating. Specifically, more opportune moments for collecting correctly-moded stroke information and for TL training could be found to prevent the collection of error-prone data and the disruption of the user during crucial work. In addition, a method to allow for training while also using Magic Pen could allow the model to adapt to users' changing annotation styles without needing to either wait or manually invoke the training. In addition, other non-pen signals could be used to aid in the classification process, such as device orientation.

12.3.4 Naturalistic Annotation Studies

A significant portion of the two evaluation studies' design focused on enabling the recording of naturalistic annotations. These annotations were a requirement due to the result of the data collection study, in which it was found that when given strict markup questions, participants would complete them in a very linear fashion — for example, completing all drawing tasks before moving on to underlining. While all right in some scenarios, these restrictive annotations limited the model's growth early on as the linear annotations tended to have little variation, which is vital for training a ML model. The following studies resolved this by asking participants to edit a document laden with spelling mistakes without any further instructions. While effective, the technique used was still a proxy for naturalistic annotations. Future work within the field of annotation study design could allow for the better recording of strokes, leading to better-trained techniques.

12.3.5 Preferences and Cognitive Load

When creating the two evaluation studies, a Likert scale was used to measure qualitative user metrics, such as Frustration and overall Preference. However, while capturing the user-perceived metrics, some metrics could potentially differ from what the users truly felt. For example, user-perceived frustration might differ from their actual frustration compared to a menu-based ap-

proach. In addition, the wording used within the questions asked (see Table 11.1) might have skewed their results as well. Different measures for such attributes have been explored in other works, such as using the positive/negative affects of frustration after using a specified system [9, 37]. Future work within the design of annotation studies could explore the effects of factors such as frustration with the use of validated instruments.

12.3.6 Magic Eraser

During the design of the heuristic eraser mitigation methods, it became evident that the techniques solved a similar challenge to that of the pen-mode switching. Namely, both methods attempt to determine whether the user would like to change the stroke's visible mode (first mitigation method) or remove the stroke altogether (second method). In addition, when it comes to erasers in annotation environments, two types are typically used: the pixel and object erasers [13]. The pixel-eraser functions similarly to a traditional eraser, allowing for strokes to be removed gradually. On the other hand, the stroke-eraser removes entire strokes on first contact with the eraser.

While not implemented, I believe that the methods discussed could be applied to change eraser modes dynamically. A similar method to the Rule-based model would implicitly infer what the user would like to do before they erase. Due to limitations of the Microsoft Surface Pen's eraser tip (and likely most other pens), the only feature available is the eraser's location on-screen. With this restriction in mind, the drafted technique would dynamically use the eraser location and velocity to change between pixel and stroke erasers and eraser size. Additional heuristics relating to the last-erased stroke could also be used to aid in the inference. Unlike the original design, which aids in stroke creation, special consideration must be taken with an eraser technique as there is a large risk of error, such as erasing a user's work, which can be particularly frustrating. On the other hand, allowing for dynamic eraser-mode switching can benefit the typical editing sequence and allow users to focus on creating new strokes instead of dealing with old ones.

12.4 Conclusion

My stroke inferring pipeline, pen-data-based neural network, implicit mode-changing system, and eraser mitigation techniques are promising steps toward reducing the cognitive load needed within annotation interfaces and improving the overall end-user experience of digital annotations from a human-computer interaction perspective. The method discussed can determine how the user is holding and using a digital pen and feed that information into an estimation model to predict the stroke the user would most like to use. Additional techniques

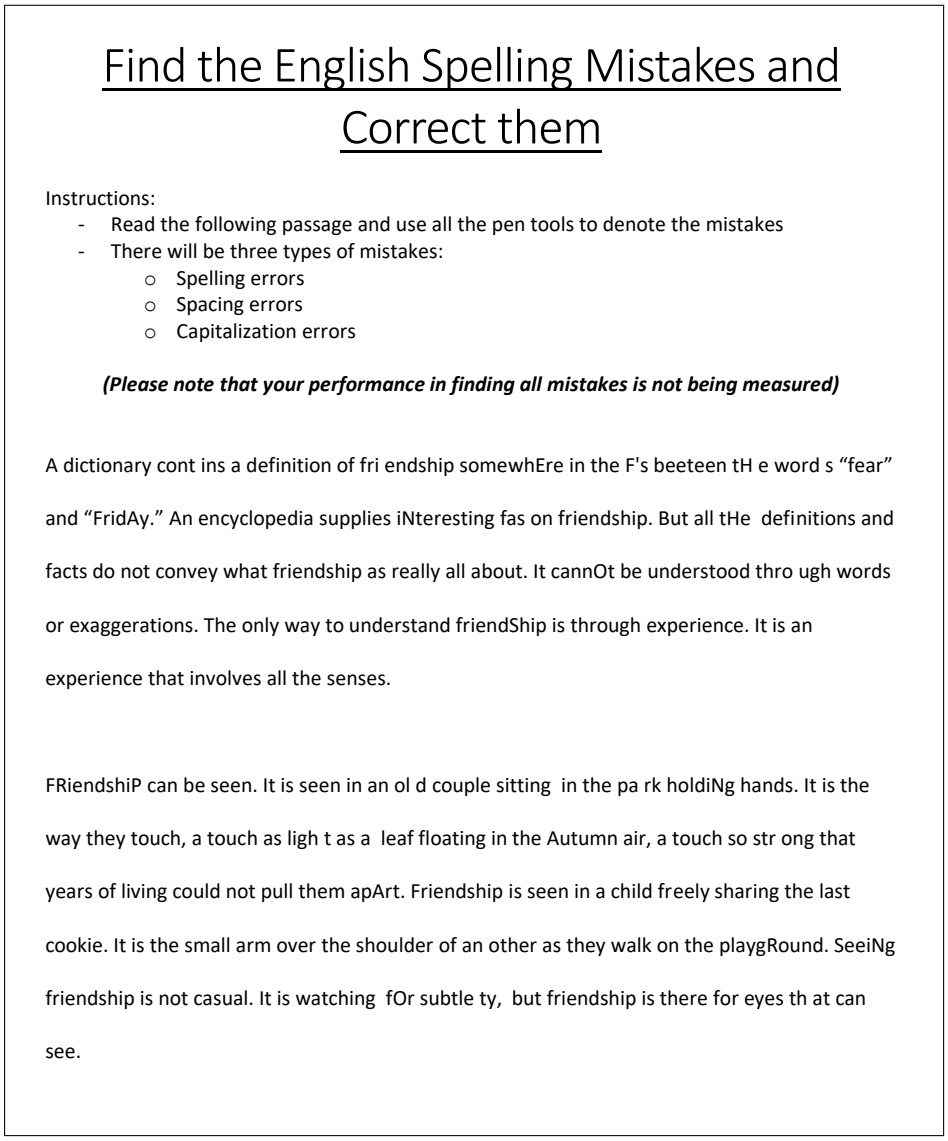
are used to help iteratively improve the model over time, tuning towards its specific user. These methods help improve usability within annotation interfaces and aim to increase the appeal of digital stylus-equipped devices.

With two evaluations of the discussed systems conducted, it was evident that there is user appeal for a system that allows for the pen mode to be automatically changed without telling the system beforehand. Although there is room for improvement with the methods implemented, it is clear that they are successful in this goal of allowing for implicit annotation. In summary, I was able to accurately predict stroke modes across different form factors and use cases, design and implement a system that supports implicit annotations, craft a method to support that system with iterative learning improvements, and implement a method to aid users in the event of an error.

Acronyms

LSTM	A form of Recurrent Neural Network
ML	Machine Learning
PDF	A file format developed by Adobe in 1992 to present documents
SVG	An XML-based vector image format for defining two-dimensional graphics
TF	A Machine Learning library by Google
TFjs	The Tensorflow library for JavaScript
TL	A Machine Learning technique which focuses on storing knowledge gained while solving one problem and applying it to a different but related problem

Appendix



Find the English Spelling Mistakes and Correct them

Instructions:

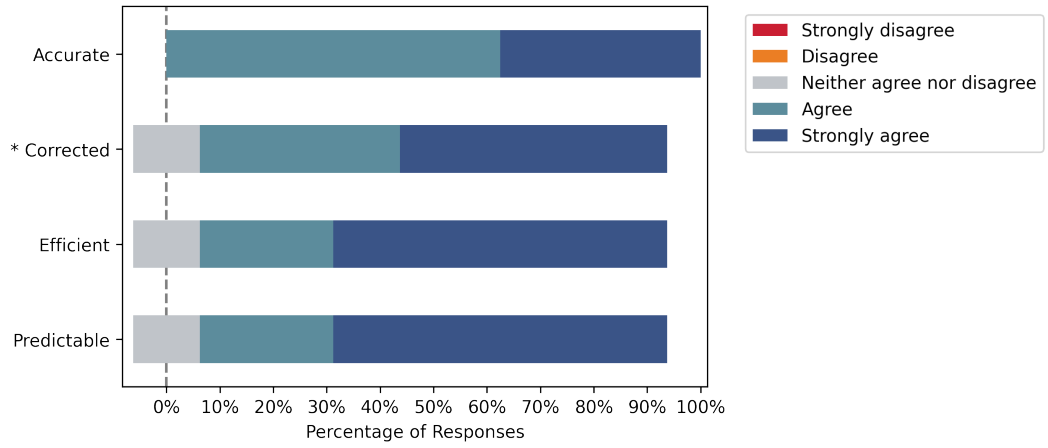
- Read the following passage and use all the pen tools to denote the mistakes
- There will be three types of mistakes:
  - Spelling errors
  - Spacing errors
  - Capitalization errors

(Please note that your performance in finding all mistakes is not being measured)

A dictionary cont ins a definition of fri endship somewhEre in the F's beeteen tH e word s “fear” and “FridAy.” An encyclopedia supplies iNteresting fas on friendship. But all tHe definitions and facts do not convey what friendship as really all about. It cannOt be understood thro ugh words or exaggerations. The only way to understand friendShip is through experience. It is an experience that involves all the senses.

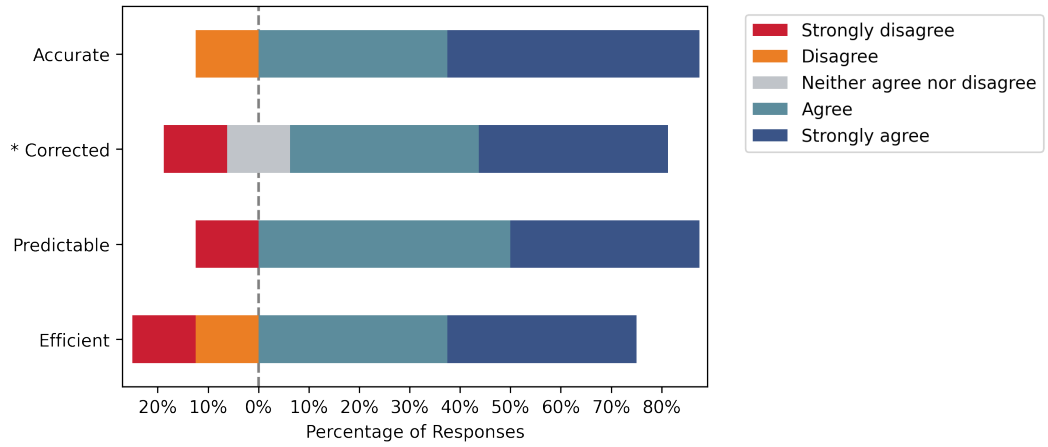
FRiendshiP can be seen. It is seen in an ol d couple sitting in the pa rk holdiNg hands. It is the way they touch, a touch as ligh t as a leaf floating in the Autumn air, a touch so str ong that years of living could not pull them apArt. Friendship is seen in a child freely sharing the last cookie. It is the small arm over the shoulder of an other as they walk on the playgRound. SeeiNg friendship is not casual. It is watching fOr subtle ty, but friendship is there for eyes th at can see.

Figure A.1: One of the provided sample documents for participants within the evaluation study that did not have documents to annotate on. Instructions were chosen to allow for as natural as possible annotations.



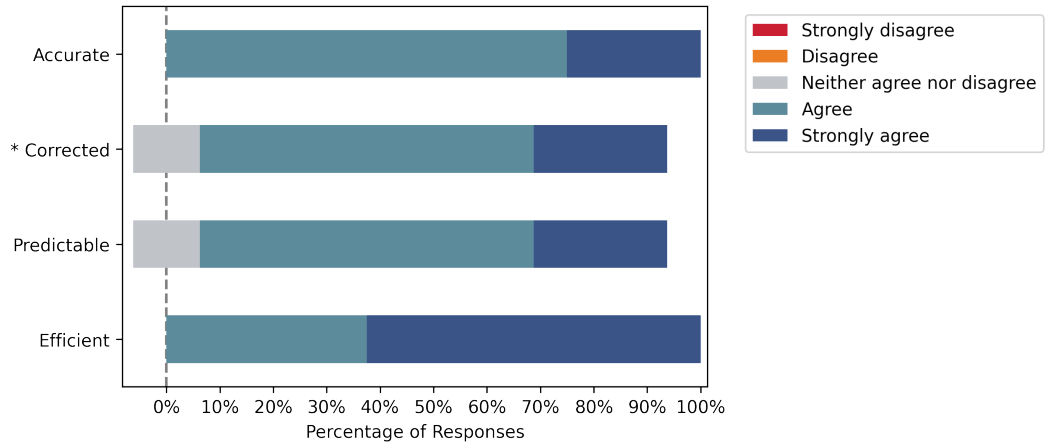
(*) This question has has its values flipped

(a) Likert Responses From TheTutorial Session



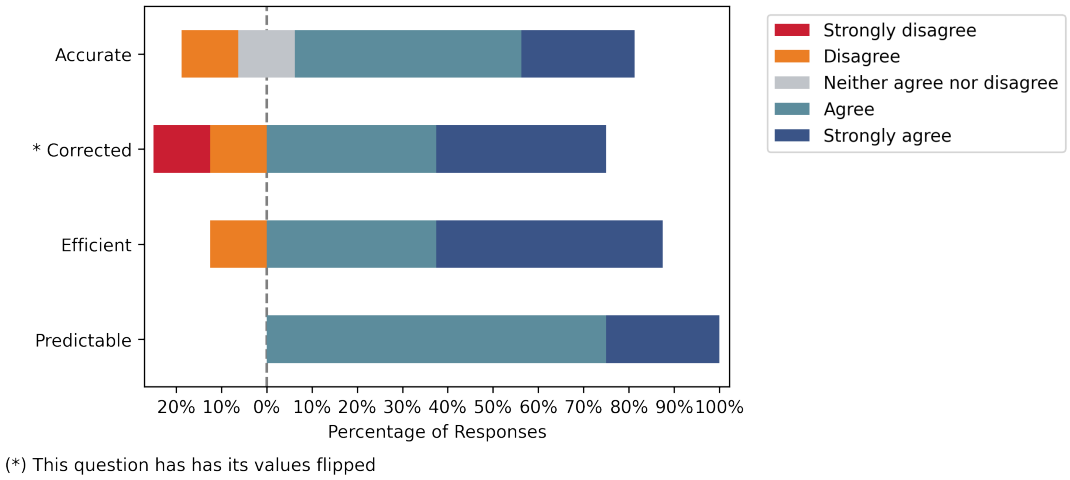
(*) This question has has its values flipped

(b) Likert Responses From Session #1



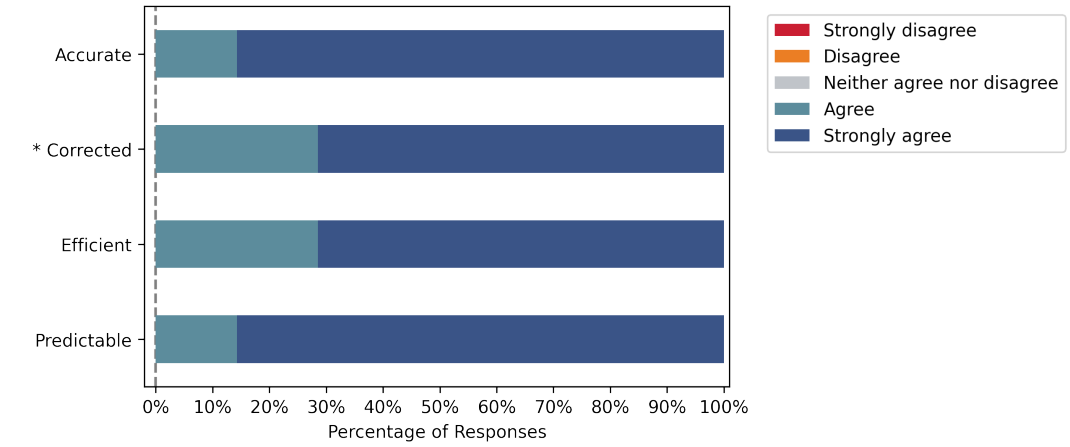
(*) This question has has its values flipped

(c) Likert Responses From Session #2



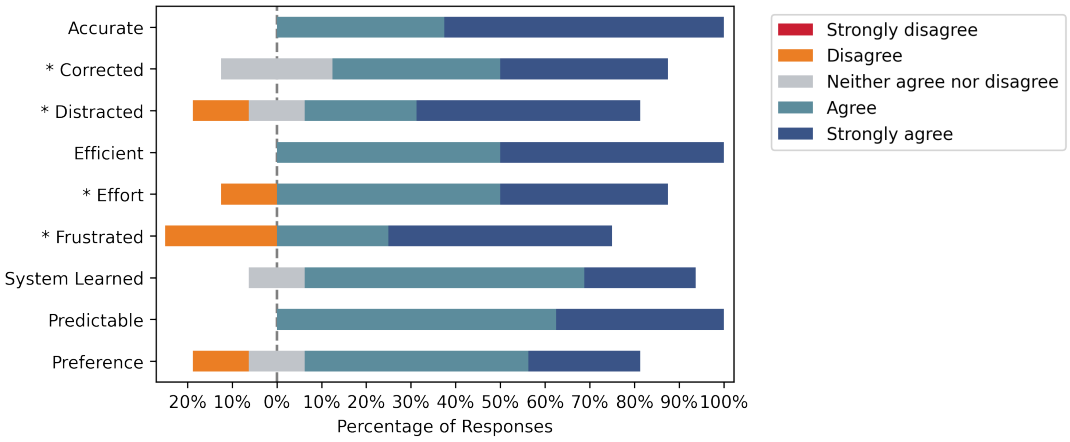
(*) This question has has its values flipped

(d) Likert Responses From Session #3



(*) This question has has its values flipped

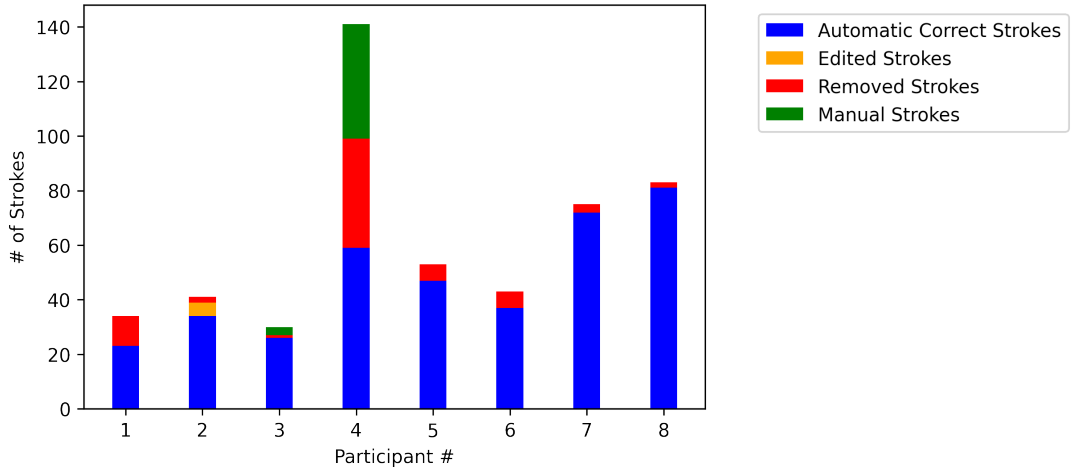
(e) Likert Responses From Session #4



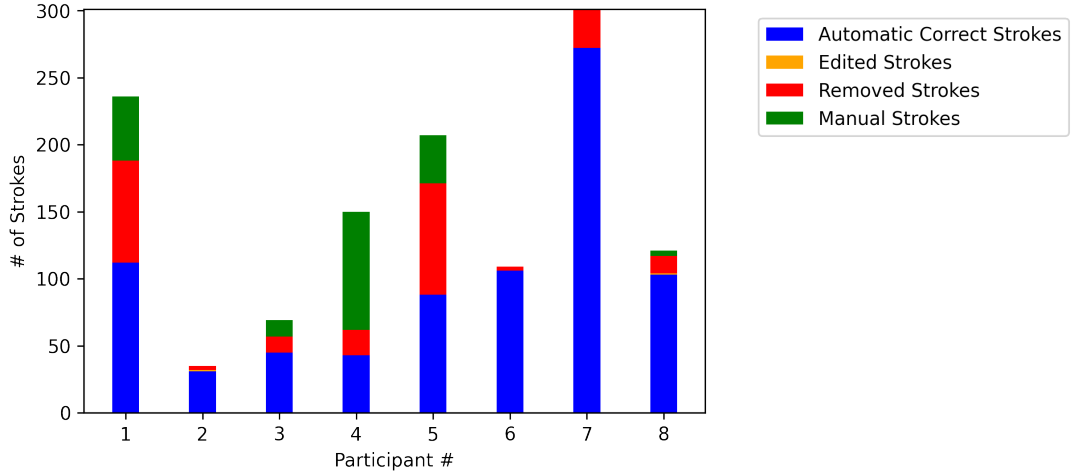
(*) This question has has its values flipped

(f) Final Likert Responses From Session #5

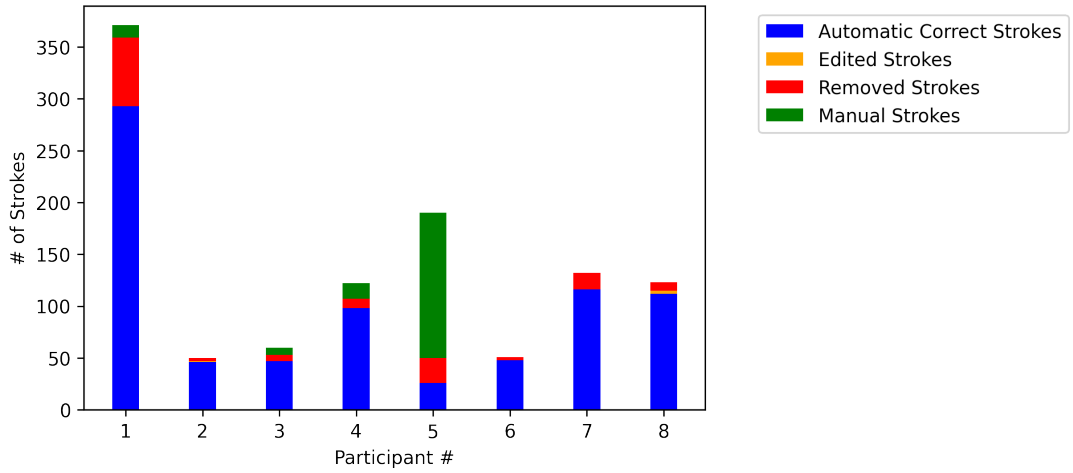
Figure A.2: Likert responses from each participant across sessions. Note that Corrected, Distracted, Effort, and Frustrated have had their values so that higher values are better.



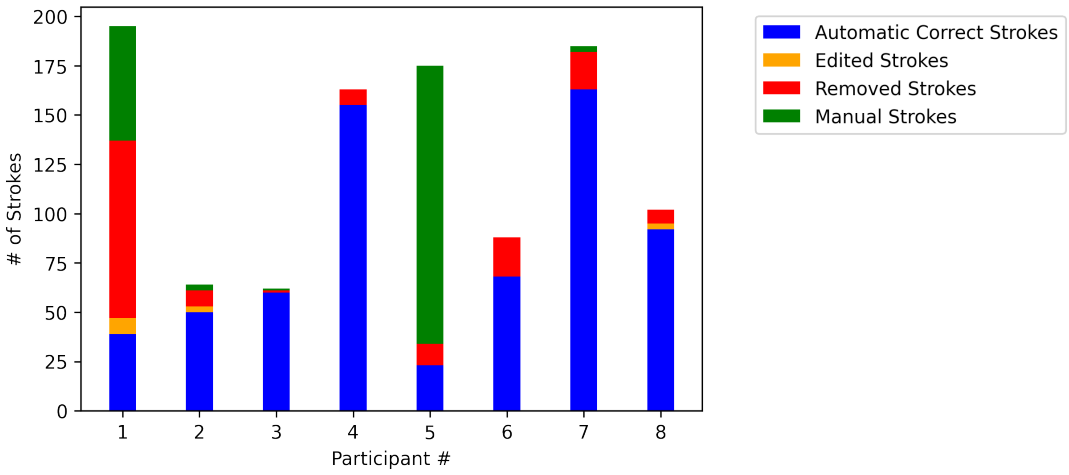
(a) Stroke Edits Made From The Tutorial Session



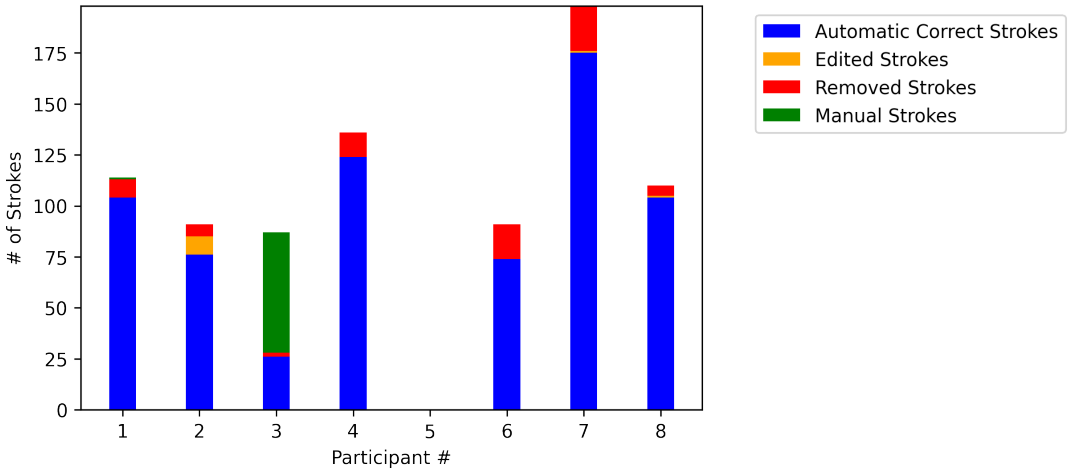
(b) Stroke Edits Made From Session #1



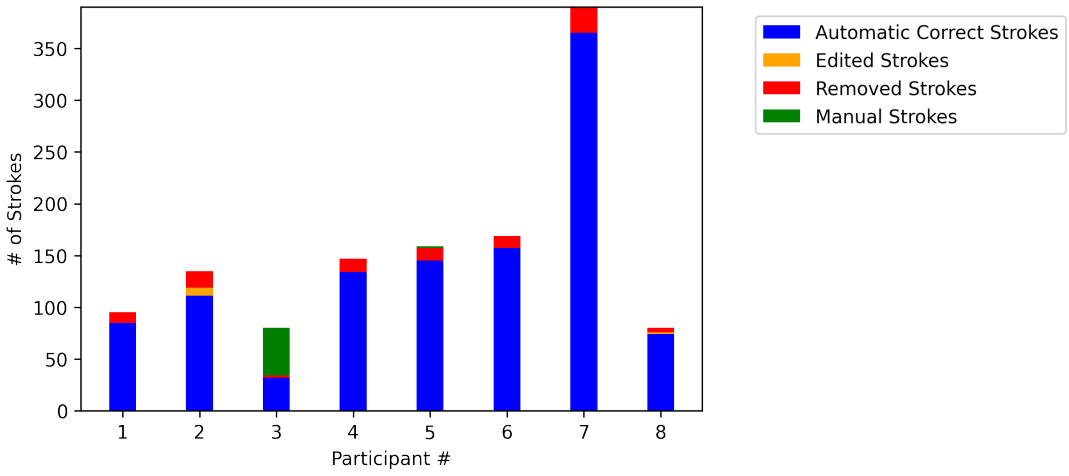
(c) Stroke Edits Made From Session #2



(d) Stroke Edits Made From Session #3

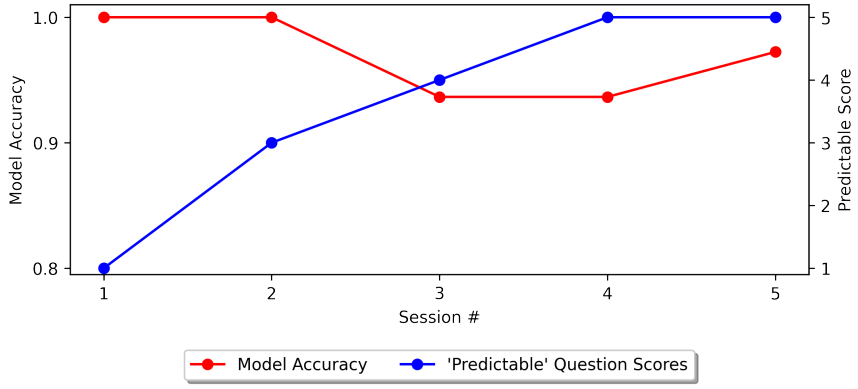


(e) Stroke Edits Made From Session #4

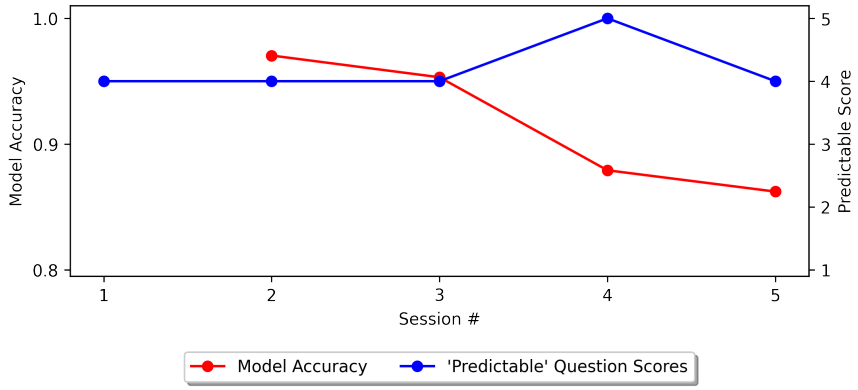


(f) Final Stroke Edits Made From Session #5

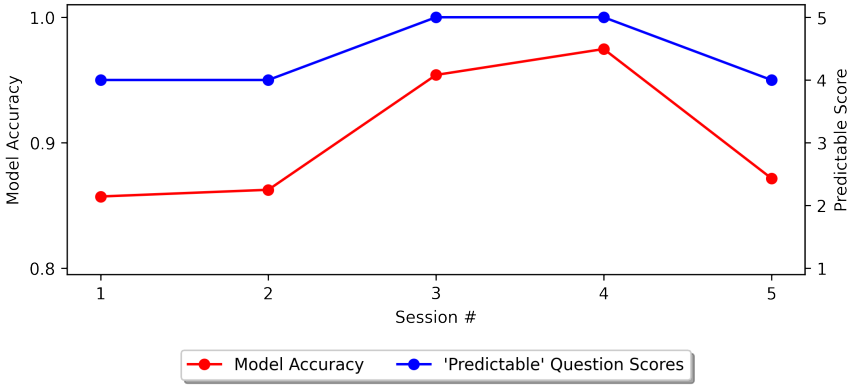
Figure A.3: Stroke edits made by participant across sessions. Note that P5's session #4 is missing and is empty.



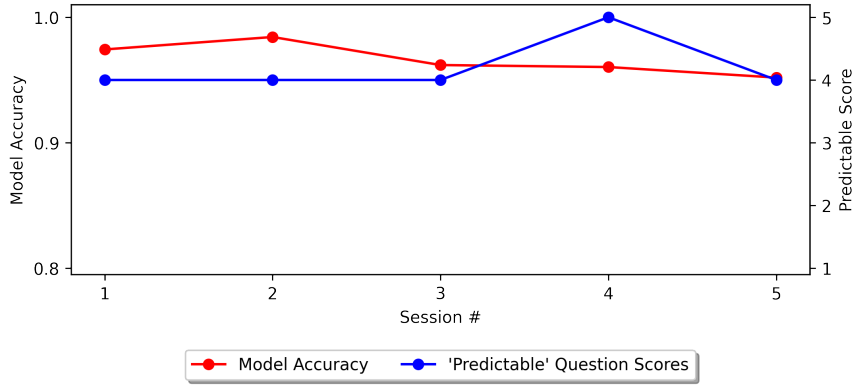
(a) Model Accuracy vs. Predictability For P1



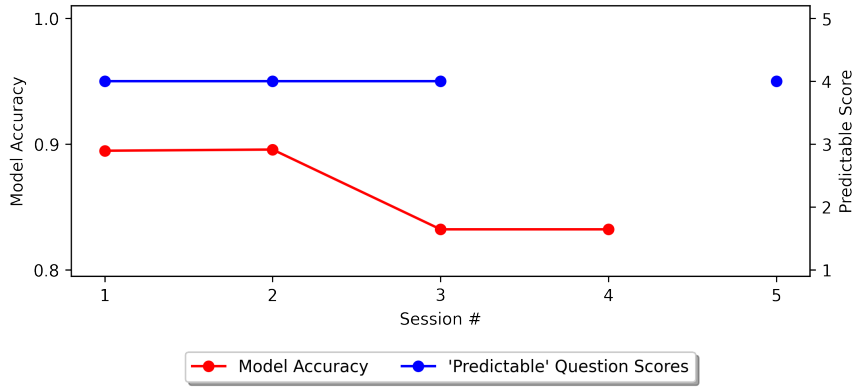
(b) Model Accuracy vs. Predictability For P2 (Note: model did not train after session 0, leaving a gap)



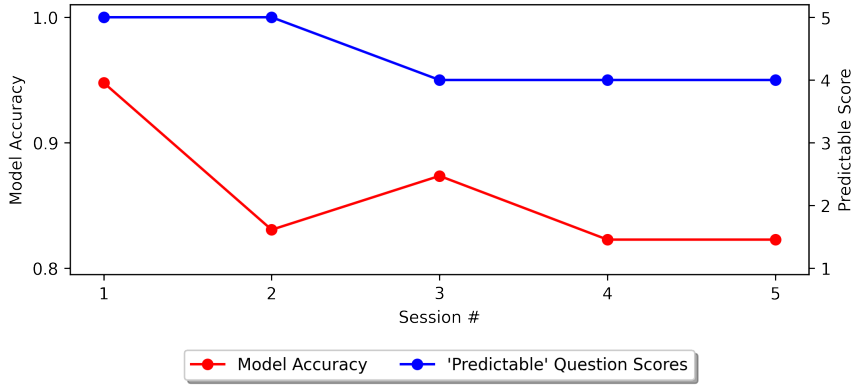
(c) Model Accuracy vs. Predictability For P3



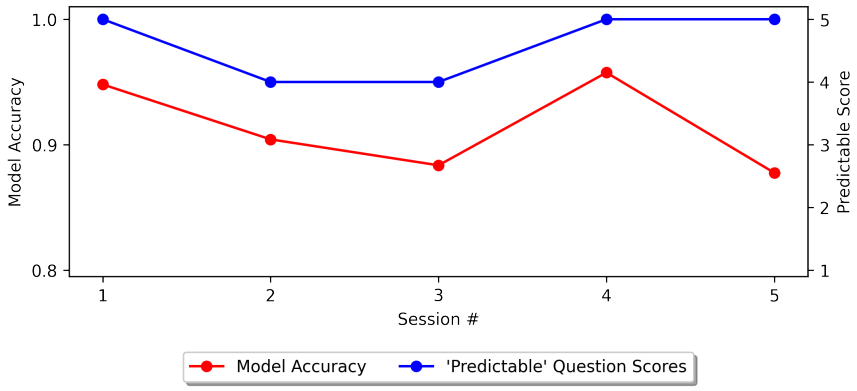
(d) Model Accuracy vs. Predictability For P4



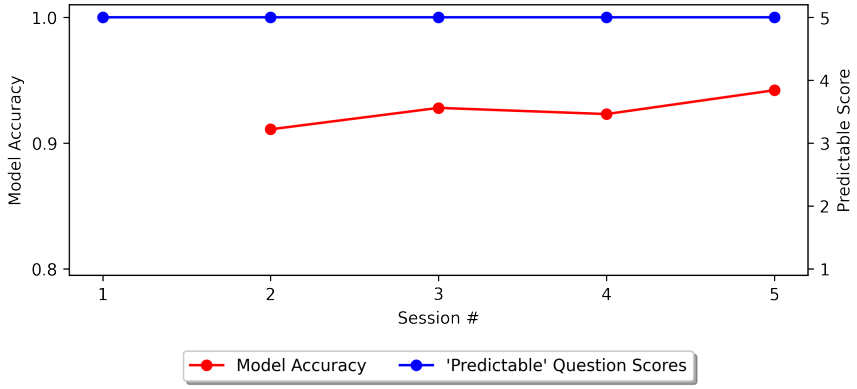
(e) Model Accuracy vs. Predictability For P5 (Note: missing session 4 data, leaving two misaligned gaps due to axes shift)



(f) Model Accuracy vs. Predictability For P6

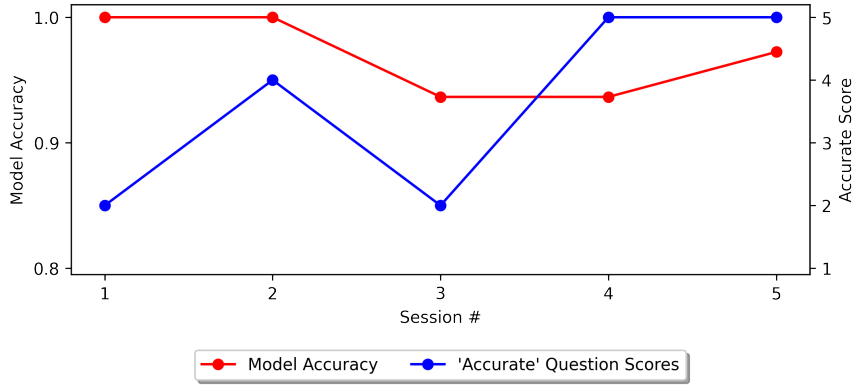


(g) Model Accuracy vs. Predictability For P7

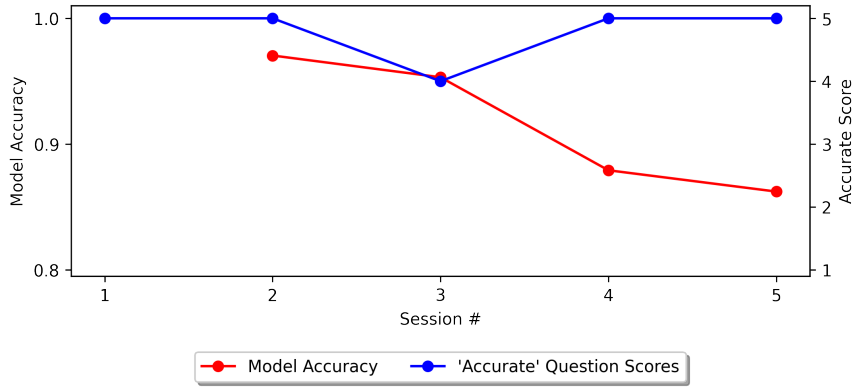


(h) Model Accuracy vs. Predictability For P8 (Note: model did not train after session 0, leaving a gap)

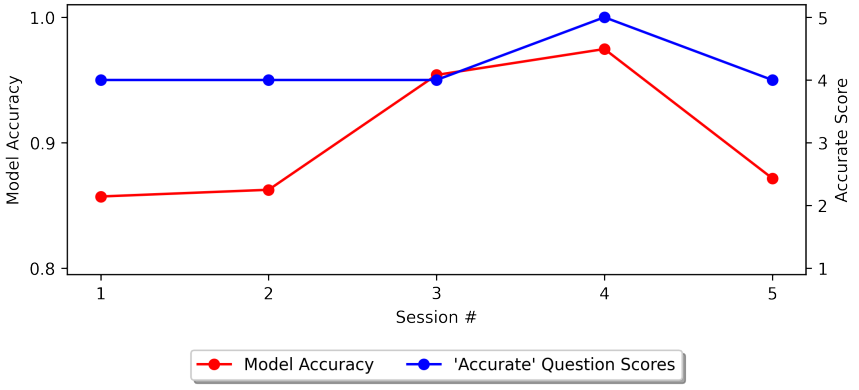
Figure A.4: Model accuracies compared to perceived predictability metrics over all sessions. Accuracies were shifted left (session 5 removed) and question scores right (session 0 removed) by one so that the model lines up with the impacted questionnaire.



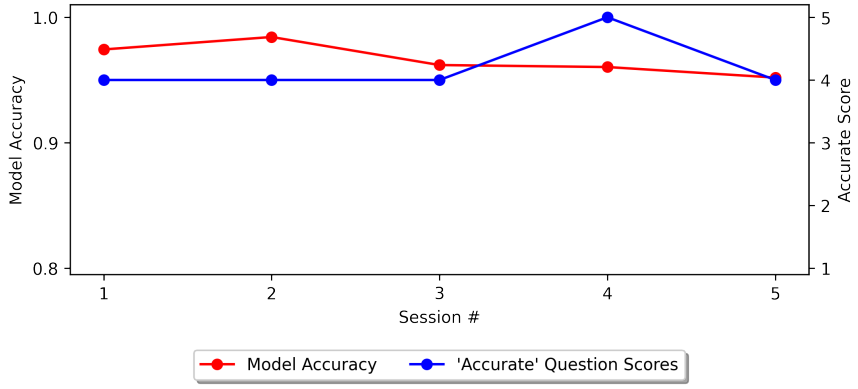
(a) Model Accuracy vs. Perceived Accuracy For P1



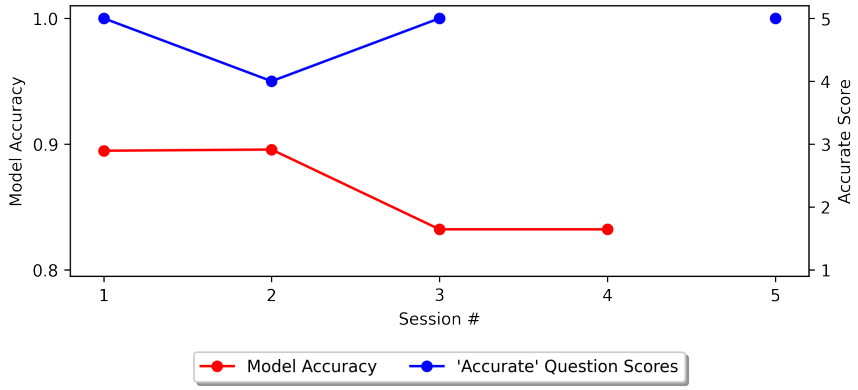
(b) Model Accuracy vs. Perceived Accuracy For P2 (Note: model did not train after session 0, leaving a gap)



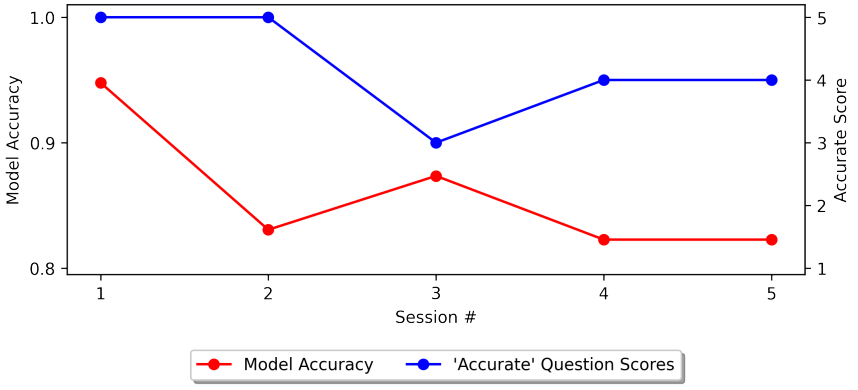
(c) Model Accuracy vs. Perceived Accuracy For P3



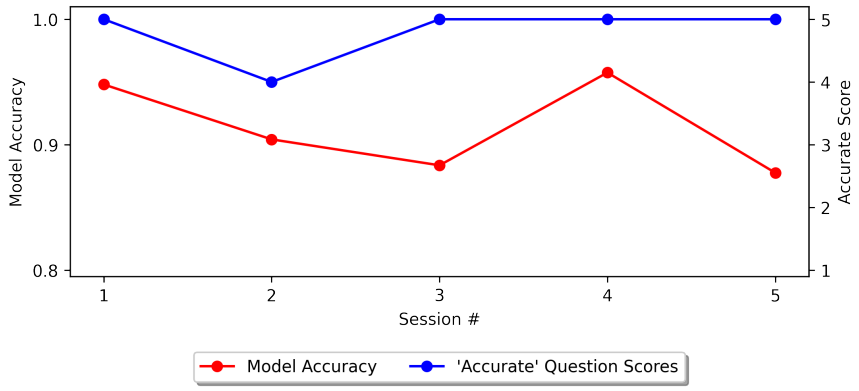
(d) Model Accuracy vs. Perceived Accuracy For P4



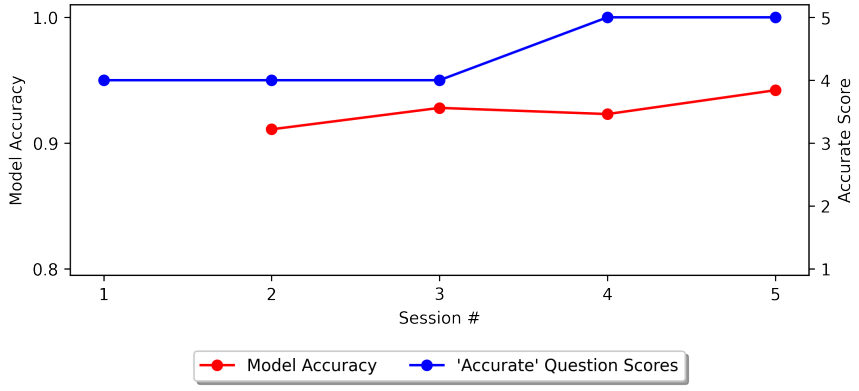
(e) Model Accuracy vs. Perceived Accuracy For P5 (Note: missing session 4 data, leaving two misaligned gaps due to axes shift)



(f) Model Accuracy vs. Perceived Accuracy For P6

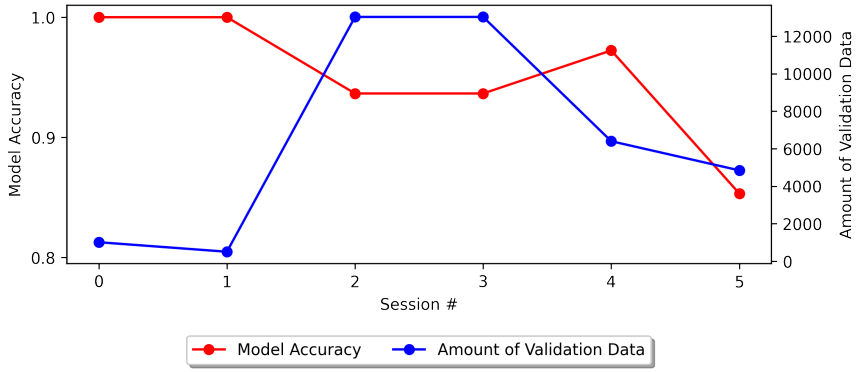


(g) Model Accuracy vs. Perceived Accuracy For P7

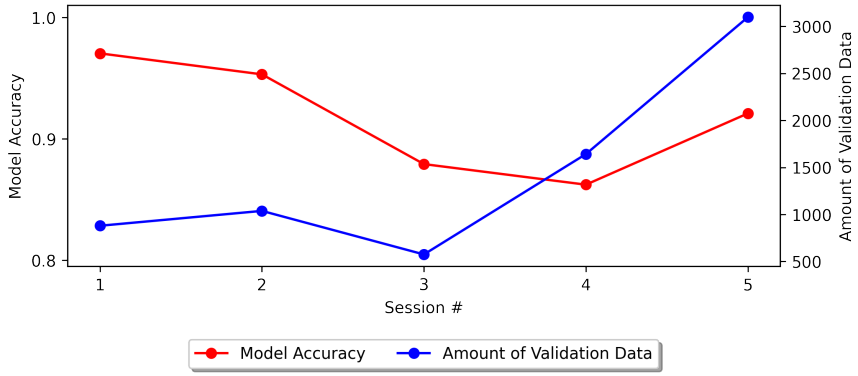


(h) Model Accuracy vs. Perceived Accuracy For P8 (Note: model did not train after session 0, leaving a gap)

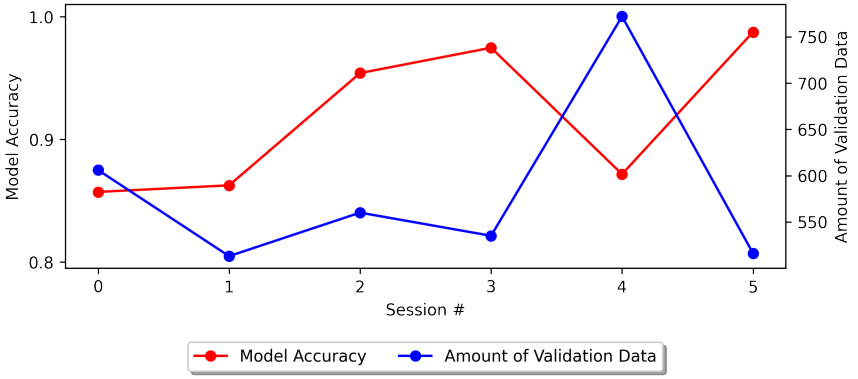
Figure A.5: Model accuracies compared to perceived accuracy metrics over all sessions. Accuracies were shifted left (session 5 removed) and question scores right (session 0 removed) by one so that the model lines up with the impacted questionnaire.



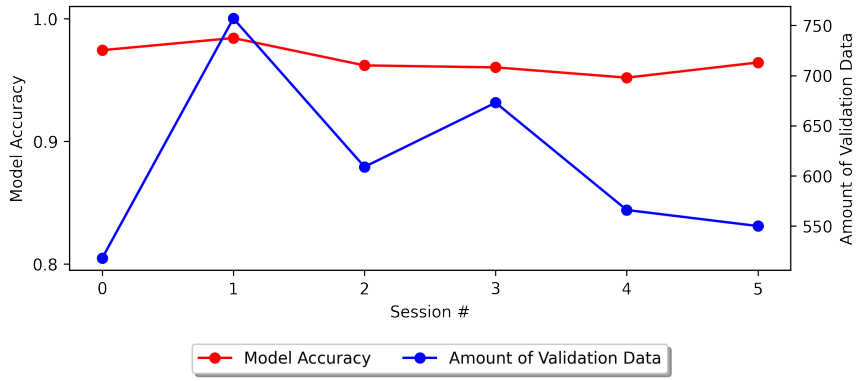
(a) Model Accuracy vs Amnt of Training Data For P1



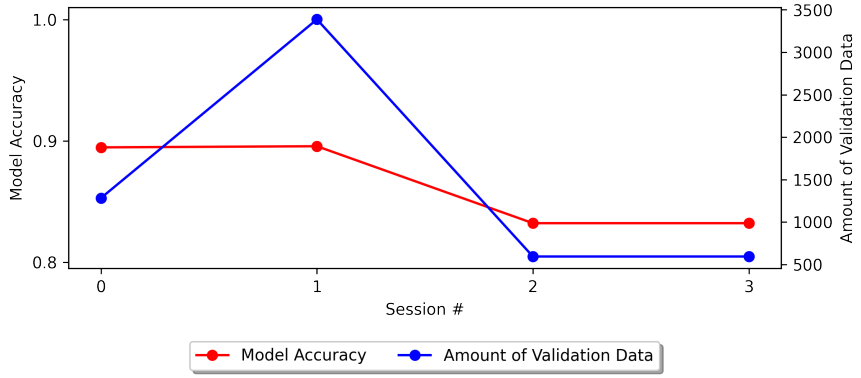
(b) Model Accuracy vs Amnt of Training Data For P2 (Note: model did not train after session 0, skipping 0 on the x-axis)



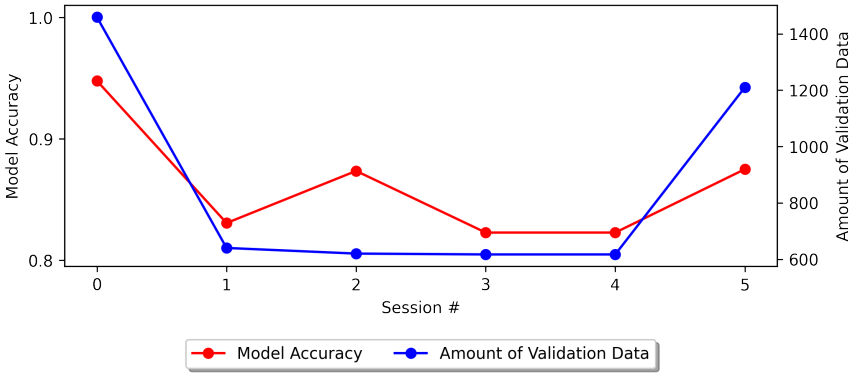
(c) Model Accuracy vs Amnt of Training Data For P3



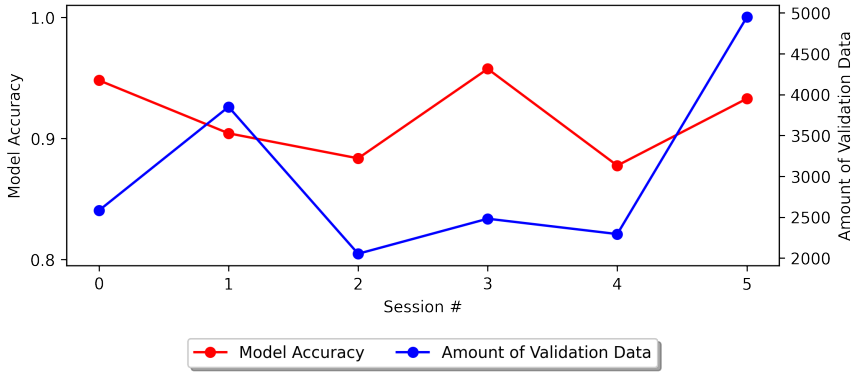
(a) Model Accuracy vs Amnt of Training Data For P4



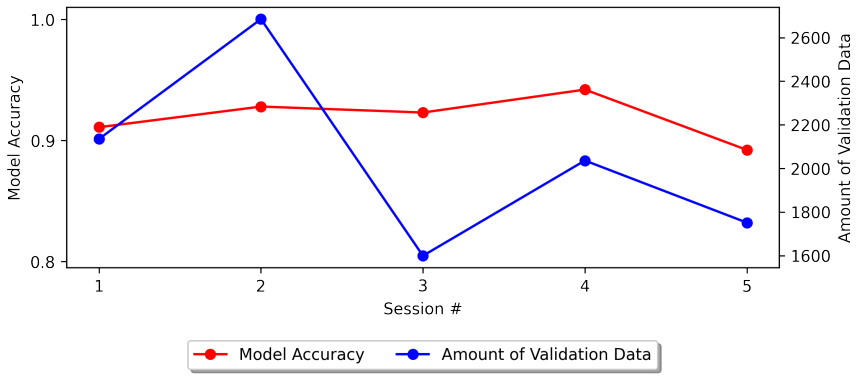
(b) Model Accuracy vs Amnt of Training Data For P5 (Note: missing session 4 data and model did not successfully train during session 5, skipping 4 and 5 on the x-axis)



(c) Model Accuracy vs Amnt of Training Data For P6



(d) Model Accuracy vs Amnt of Training Data For P7



(e) Model Accuracy vs Amnt of Training Data For P8 (Note: model did not train after session 0, skipping 0 on the x-axis)

Figure A.7: Model accuracies compared to the amount of training data recorded over all sessions.

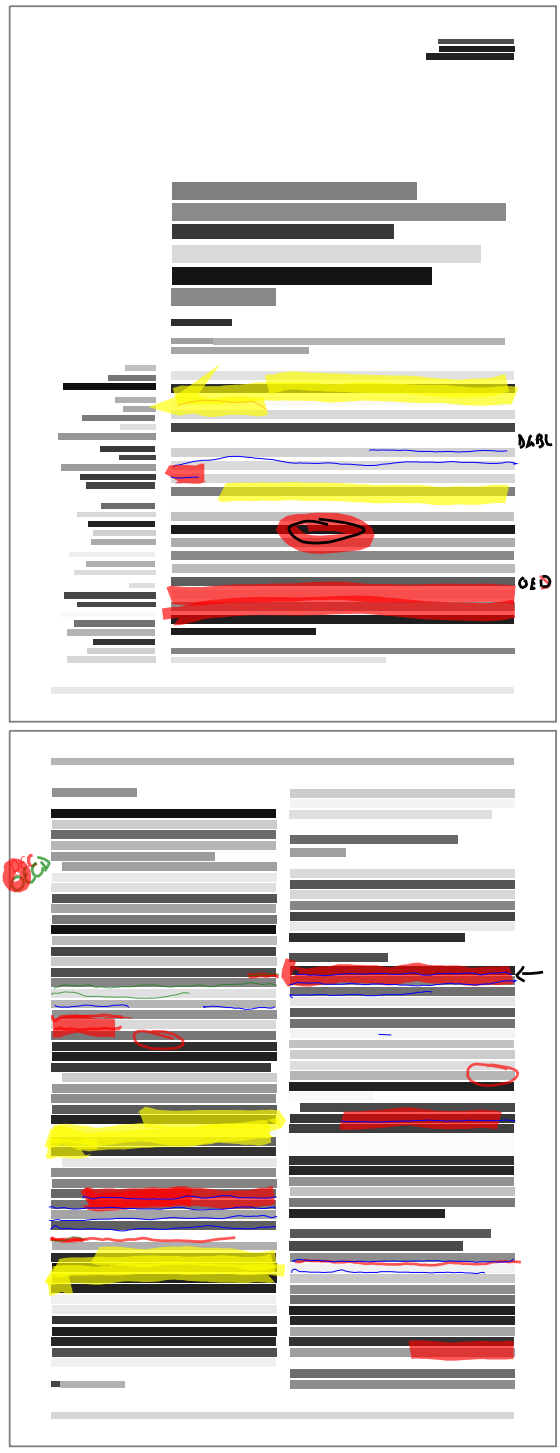


Figure A.8: A sample of annotations made by P1 during session 1 of the evaluation study. Red represents removed strokes, green is for manually moded strokes, and yellow, blue, and black strokes are used for highlight, underline, and draw modes, respectively.

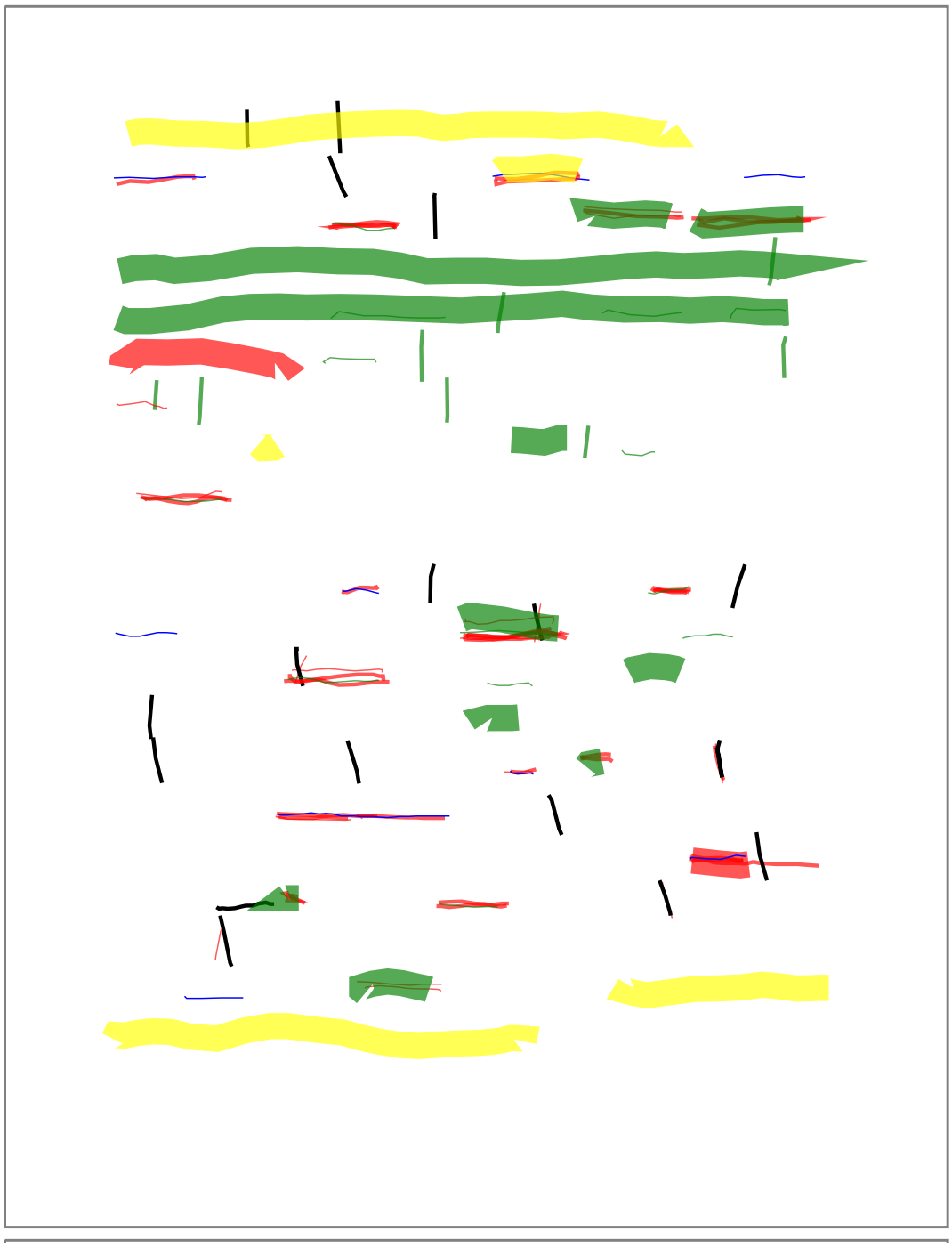


Figure A.9: A sample of annotations made by P1 during session 3 of the evaluation study. Red represents removed strokes, green is for manually moded strokes, and yellow, blue, and black strokes are used for highlight, underline, and draw modes, respectively.

Figure A.10: A sample of annotations made by P5 during session 1 of the evaluation study. Red represents removed strokes, green is for manually moded strokes, and yellow, blue, and black strokes are used for highlight, underline, and draw modes, respectively.

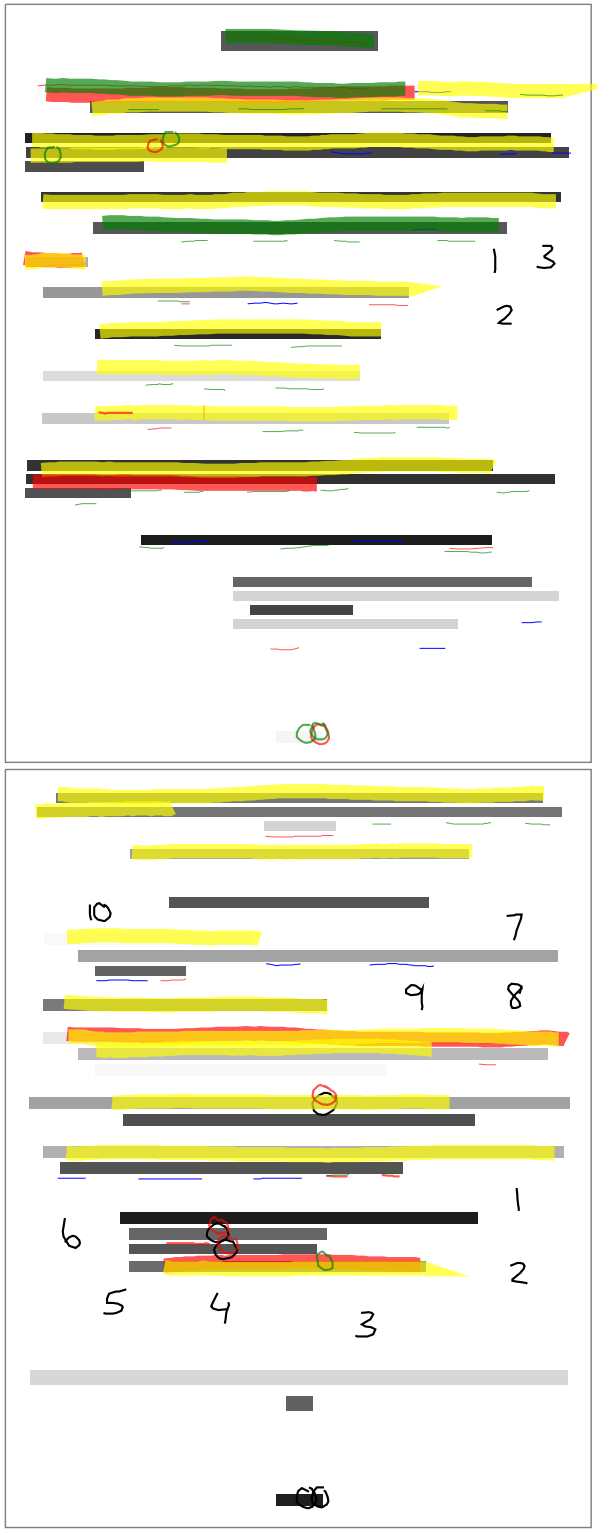


Figure A.11: A sample of annotations made by P4 during session 0 of the evaluation study. Red represents removed strokes, green is for manually moded strokes, and yellow, blue, and black strokes are used for highlight, underline, and draw modes, respectively.

Bibliography

1. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". *arXiv Preprint arXiv:1603.04467*, 2016.
2. M. Annett. "(Digitally) Inking in the 21st century". *IEEE Computer Graphics and Applications* 37:1, 2017, pp. 92–99.
3. M. Annett, F. Anderson, W. F. Bischof, and A. Gupta. "The pen is mightier: understanding stylus behaviour while inking on tablets". In: *Proc. Graphics Interface*. 2014, pp. 193–200.
4. M. Annett and W. F. Bischof. "Hands, hover, and nibs: understanding stylus accuracy on tablets". *Proc. Graphics Interface*. 2015, pp. 203–210.
5. M. Annett, A. Gupta, and W. F. Bischof. "Exploring and understanding unintended touch during direct pen interaction". *ACM Transactions on Computer-Human Interaction (TOCHI)* 21:5, 2014, pp. 1–39.
6. M. Annett, A. Ng, P. Dietz, W. F. Bischof, and A. Gupta. "How low should we go? Understanding the perception of latency while inking". In: *Proc. Graphics Interface*. 2014, pp. 167–174.
7. M. K. Annett. "The fundamental issues of pen-based interaction with tablet devices", 2014.
8. J. Arvo. "Computer aided serendipity: The role of autonomous assistants in problem solving". *Graphics Interface*. Vol. 99. 1999.
9. K. Bessiere, J. E. Newhagen, J. P. Robinson, and B. Shneiderman. "A model for computer frustration: The role of instrumental and dispositional factors on incident, session, and post-session frustration and mood". *Computers in human behavior* 22:6, 2006, pp. 941–961.
10. M. Brubeck, R. Byers, P. H. Lauke, and N. Zolghadr. *Pointer Events*. W3C Recommendation. W3C, 2019.
11. J. Chorowski and J. M. Zurada. "Learning understandable neural networks with nonnegative weight constraints". *IEEE Trans. on Neural Networks and Learning Systems* 26:1, 2014, pp. 62–69.

12. C. B. Do and A. Y. Ng. "Transfer learning for text classification". *Advances in Neural Information Processing Systems* 18, 2005.

13. *Draw in apps with markup on iPhone*. URL: https://support.apple.com/guide/iphone/draw-in-apps-iph893c6f8bf/ios.

14. C. Frankish, R. Hull, and P. Morgan. "Recognition accuracy and user acceptance of pen interfaces". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 1995, pp. 503–510.

15. A. Gholamy, V. Kreinovich, and O. Kosheleva. "Why 70/30 or 80/20 relation between training and testing sets: a pedagogical explanation", 2018.

16. A. Graves, N. Jaitly, and A.-r. Mohamed. "Hybrid speech recognition with deep bidirectional LSTM". *IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. 2013, pp. 273–278.

17. I. Green. *Web workers: Multithreaded programs in javascript.* " O'Reilly Media, Inc.", 2012.

18. K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. "Design and analysis of delimiters for selection-action pen gesture phrases in Scriboli". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. ACM. 2005, pp. 451–460.

19. J. Hunt. "Graphing with Matplotlib pyplot". In: *Advanced Guide to Python 3 Programming*. Springer, 2019, pp. 43–65.

20. S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". *arXiv Preprint arXiv:1502.03167*, 2015.

21. R. Kalkur. *Tuning SVG performance in a production app*. 2014. URL: https://slides.com/rovolutionary/tuning-svg-performance-in-a-production-app.

22. S. Khan, N. Islam, Z. Jan, I. U. Din, and J. J. C. Rodrigues. "A novel deep learning based framework for the detection and classification of breast cancer using transfer learning". *Pattern Recognition Letters* 125, 2019, pp. 1–6.

23. E. Lank, J. Ruiz, and W. B. Cowan. "Concurrent bimanual stylus interaction: a study of non-preferred hand mode manipulation." *Graphics Interface*. 2006, pp. 17–24.

24. Y. Li, K. Hinckley, Z. Guan, and J. A. Landay. "Experimental analysis of mode switching techniques in pen-based user interfaces". *Proc. of the SIGCHI Conf. on Human factors in computing systems*. 2005, pp. 461–470.

25. J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay. "DENIM: finding a tighter fit between tools and practice for Web site design". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2000, pp. 510–517.

26. C. E. Lui. *Adaptive input pen mode selection*. US Patent 6,683,600. 2004.
27. C. E. Lui, K. L. Parker, and D. W. Altman. *System and method for automatically switching between writing and text input modes*. US Patent 6,552,719. 2003.
28. C. C. Marshall. "Annotation: from paper books to the digital library". *Proc. of the ACM Int. Conf. on Digital libraries*. 1997, pp. 131–140.
29. J.-n. Meng, H.-f. Lin, and Y.-h. Yu. "Transfer learning based on svd for spam filtering". *Proc. Int. Conf. on Intelligent Computing and Cognitive Informatics*. IEEE. 2010, pp. 491–494.
30. D. R. Millen. "Pen-Based User Interfaces". *AT&T Technical J.* 72:3, 1993, pp. 21–27.
31. R. Mitton. URL: https://www.dcs.bbk.ac.uk/~ROGER/corpora.html.
32. *MuPDF*. URL: https://mupdf.com/.
33. E. D. Mynatt, T. Igarashi, W. K. Edwards, and A. LaMarca. "Flatland: New Dimensions in Office Whiteboards". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 1999, pp. 346–353.
34. M. Negulescu, J. Ruiz, and E. Lank. "Exploring Usability and Learnability of Mode Inferencing in Pen/Tablet Interfaces." *SBIM*. Citeseer. 2010, pp. 87–94.
35. M. D. Network. *Pointer Events API*. https://developer.mozilla.org/en-US/docs/Web/API/Pointer_events. 2020.
36. J. Nielsen. "Enhancing the explanatory power of usability heuristics". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 1994, pp. 152–158.
37. A. Nylund and O. Landfors. *Frustration and its effect on immersion in games: A developer viewpoint on the good and bad aspects of frustration*. 2015.
38. *Offscreencanvas: Can I use...* URL: https://caniuse.com/offscreencanvas.
39. I. A. Ovsiannikov, M. A. Arbib, and T. H. McNeill. "Annotation technology". *Int. J. Human-Computer Studies* 50:4, 1999, pp. 329–362.
40. *Paper.js*. URL: http://paperjs.org/.
41. *PDFKit*. URL: https://pdfkit.org/.
42. L. Y. Pratt, J. Mostow, C. A. Kamm, A. A. Kamm, et al. "Direct Transfer of Learned Information Among Neural Networks." *Proc. AAAI*. Vol. 91. 1991, pp. 584–589.
43. S. Ramachandran and R. Kashi. "An architecture for ink annotations on web documents". *Proc. Int. Conf. on Document Analysis and Recognition*. IEEE. 2003, pp. 256–260.

44. Y. Riche, N. H. Riche, K. Hinckley, S. Panabaker, S. Fuelling, and S. Williams. "As We May Ink?: Learning from Everyday Analog Pen Use to Improve Digital Ink Experiences." *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2017, pp. 3241–3253.

45. H. Romat, N. Henry Riche, K. Hinckley, B. Lee, C. Appert, E. Pietriga, and C. Collins. "ActiveInk: (Th)inking with Data". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2019, pp. 1–13.

46. H. Romat, E. Pietriga, N. Henry-Riche, K. Hinckley, and C. Appert. "Spaceink: Making space for in-context annotations". *Proc. ACM Symposium on User Interface Software and Technology*. 2019, pp. 871–882.

47. E. Saund and E. Lank. "Stylus input and editing without prior selection of mode". *Proc. ACM Symp. on User Interface Software and Technology*. 2003, pp. 213–216. DOI: 10.1145/964696.964720.

48. M. Schrapel, M.-L. Stadler, and M. Rohs. "Pentelligence: Combining pen tip motion and writing sounds for handwritten digit recognition". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2018, pp. 1–11.

49. T. Semwal, P. Yenigalla, G. Mathur, and S. B. Nair. "A practitioners' guide to transfer learning for text classification using convolutional neural networks". *Proc. SIAM Int. Conf. on Data Mining*. SIAM. 2018, pp. 513–521.

50. B. Serim and G. Jacucci. "Explicating" Implicit Interaction" An Examination of the Concept and Challenges for Research". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2019, pp. 1–16.

51. R. K. Sevakula, V. Singh, N. K. Verma, C. Kumar, and Y. Cui. "Transfer learning for molecular cancer classification using deep neural networks". *IEEE/ACM Trans. on Computational Biology and Bioinformatics* 16:6, 2018, pp. 2089–2100.

52. D. Smilkov, N. Thorat, Y. Assogba, C. Nicholson, N. Kreeger, P. Yu, S. Cai, E. Nielsen, D. Soegel, S. Bileschi, et al. "Tensorflow. js: Machine learning for the web and beyond". *Proceedings of Machine Learning and Systems* 1, 2019, pp. 309–321.

53. H. Song, H. Benko, F. Guimbretiere, S. Izadi, X. Cao, and K. Hinckley. "Grips and gestures on a multi-touch pen". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2011, pp. 1323–1332.

54. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". *J. Machine Learning Research* 15:1, 2014, pp. 1929–1958.

55. T. Su, S. Jia, Q. Wang, L. Sun, and R. Wang. "Novel character segmentation method for overlapped Chinese handwriting recognition based on LSTM neural networks". *Proc. Int. Conf. on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 1141–1146.

56. L. Tesler. "The Smalltalk Environment". *Byte*, 1981, pp. 90–147.

57. V. S. Tida and S. Hsu. "Universal Spam Detection using Transfer Learning of BERT Model". *arXiv preprint arXiv:2202.03480*, 2022.

58. H. Tsukimoto. "Extracting rules from trained neural networks". *IEEE Trans. on Neural Networks* 11:2, 2000, pp. 377–389.

59. D. Vogel and R. Balakrishnan. "Direct pen interaction with a conventional graphical user interface". *Human–Computer Interaction* 25:4, 2010, pp. 324–388.

60. D. Vogel and G. Casiez. "Conté: multimodal input inspired by an artist's crayon". *Proc. ACM Symp. on User Interface Software and Technology*. 2011, pp. 357–366.

61. Y. Xin, X. Bi, and X. Ren. "Natural use profiles for the pen: an empirical exploration of pressure, tilt, and azimuth". *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. 2012, pp. 801–804.

62. R. Zeleznik and T. Miller. "Fluid inking: augmenting the medium of free-form inking with gestures". *Prog. Graphics Interface*. 2006, pp. 155–162.

63. Z. Zhu, K. Lin, and J. Zhou. "Transfer learning in deep reinforcement learning: A survey". *arXiv preprint arXiv:2009.07888*, 2020.