Spell Check

Denis Doçi, Neil Rao, Anthony Nedumgottil, Brandon Davis, Katarzyna Krawczyk

Introduction:

Our project is a virtual reality based real-time spelling checker. The user will wear a Microsoft HoloLens, and on their command they will be able to identify text, and check the spelling of the identified text. Initially we went through a multitude of project ideas, but most of them centered around the idea of text recognition and applying some functionality to the recognized text. We thought of recreating the Google Translate application in a virtual reality space so that the user would be able to translate foreign text into their native one as a helpful tool to help students and traveling tourists. This however just seemed to be a recreation of an existing tool, and we wanted to create something more original and innovative. This led us to the spell check idea. With this application, we hope to target a large variety of people. This product would be most useful to people learning how to write English. This includes, but is not limited to, school children and non-native English speakers. We're also looking to target people who have disability that hinders their writing ability such as dyslexia. That being said, this product can be used by anyone. With the use of our application, a normal person's ability to spell is greatly increased. We are therefore augmenting a person's natural writing abilities.

Prior Works Research:

Prior work that we found for this project that we will be usable for ours:

1. Denis

   Processing Error Correction Algorithm using Google Online Spelling Suggestion

One of the potential problems that we're going to run into is the accuracy of our spell checker after we process our characters. With optical character recognition, it is likely that most of the characters that are read in won't be exact. This brings up issues of valid and accurate spelling corrections. In Youssef Bassil and Mohammad Alwani's paper on creating a processing error correction algorithm, they attempt to solve this exact problem of optical character recognition errors. They do this by not just using a dictionary, but instead a linguistic context-based correction algorithm. The benefits of such an algorithm for any optical character recognition project are plentiful. Not only does it allow for a buffer for imperfections when the processing is done, but it also allows for recognition of context spelling errors to be picked up which a normal dictionary-based checker might not.

The algorithm itself is reliant on Google Search's "did you mean" feature. The algorithm begins by inputting the recognized text images to a output text B. B is set up as five separate blocks for five separate words. Then, all blocks are submitted as a search query to Google's web search engine; if the search returns "did you mean: x" where x is the alternative spelling suggestion for a certain block within the sequence, then that block is considered misspelled. The algorithm therefore is not reliant on any dictionary, just Google itself.

This seems like a good approach, however the usefulness of this algorithm for our project might be slim. The main issue is obviously that whatever uses this algorithm must be connected to the internet at all times. This is an issue because we want our application to have wide uses, not just when the user has internet connectivity. That being said, the concept of using a context-based spelling checker along with a dictionary one is something we will implement. This will increase the accuracy and precision of our product greatly.

2. Katarzyna

A Database for Handwritten Text Recognition Research

The article written by Jonathan Hull explains a system of handwritten text recognition using images that are stored on the cloud. The system is presented and used in a local post office. An image database was created with about 15000 images that included city names, zip codes, and state names. Each word has multiple handwritten styles so when the address is scanned it is easily recognized by the machine to be sorted. Creating such a database can be very useful when it comes to small amount of characters. Making a database for the whole dictionary would not work so well since it would need a lot of cloud storage. This article has proved that reading handwritten text and recognizing it electronically can help in everyday life which what our project intends to do.

3. Brandon's

"Using smartglasses for utility-meter reading"

The purpose of this research paper was to develop a way to automate data logging from utility- meters using smartglass. Due to the fact that these meters were mostly analog in nature, workers would have to handle data entry by hand which inevitably leads to mistakes and human-error. The idea would be to have a user use smartglass to first take a picture of a meter, use OCR to gain the data input, and then utilize cloud services to automatically log and store the data. The researchers decided to use the Vuzix M100, an android based smartglass that also has tactile buttons. In addition to this, they also explored using voice and gesture recognition, finding gesture recognition easier and more accurate (they noted the ability to use an infrared sensor helped greatly with this). The conclusion of their research was that they found relative success being able to scan, recognize, and store meter data with good accuracy. The work done in this

research coincides a lot with what we want to do with their utilization of smartglass and OCR. Additionally, the Vuzix M100 is available to us to use, and knowing about their success with it is encouraging to know.

4. Anthony's

   Accuracy of Electronic Spell Checkers for Dyslexic Learners

   This article talks about a study that was published for the Professional Association for Teachers and assessors of Students with Specific Learning Difficulties, or PATOSS, by Abi James & EA Draffan talks about how modern day spell checkers work for correcting typos for people suffering from Dyslexia. First the two established that the most common reason why most spell checkers fail is because the lack of suggestions from a smaller dictionary. This reason applies to all users, not just those with dyslexia. Another reason why spell checkers fail for dyslexic people can be categorized by four different criterias: one letter wrong, one letter omitted, one letter inserted, and two adjacent letters transposed. These errors are a lot more common for dyslexic people, so now more modern day spell checkers changed their algorithm to accommodate and cover more of these issues by turning to a more phonetic approach. This approach checks to see how the word itself sounds and suggests correct words best on the sound. This approach is great as it has increased the amount of words corrected by spell checkers however, we have now learned that dyslexic learners also have trouble choosing the correct spelling from the suggested choices as they are not able to properly read them. These researches also conducted a test between 6 popular spell checkers and found that the spell checking tool, Franklin Literacy Word Bank, had the highest amount of corrected misspellings for the group of dyslexic users who participated in the study. Our plan is to use this spell checker to borrow ideas and features to help us target dyslexic users. However doing so might require a lot more effort and this might not

be able to be completed within the scope of the class. While helping dyslexic students was one of our initial ideas for creating Spell Check, we might not be able to accurately check their spelling.

5. Neil's

Recognizing handwritten text is more challenging than what initially meets the eye. Unlike recognizing specific fonts on printed paper (where each individual letter printed has the exact same curve), recognizing handwriting is a whole different ball game. What makes handwriting so much more difficult to analyze is two-fold. First, there is tremendous variance in handwriting styles, from print to cursive; every person appears to write in a somewhat unique fashion. Second, there is tremendous variance within a single piece of an individual's writing itself; humans, being imperfect creatures, draw the curves of each letter slightly differently each time. While our complex biological pattern-recognizing brains can (through years of schooling) learn to easily differentiate and interpret these variant styles, teaching a computer how to do that involves an entire field of cutting-edge data science research. There are several popular approaches to digitizing text, all belonging to the general category of optical text recognition (OCR). The very obvious first set of algorithms I came across involved a template-matching approach [1], where the geometry of characters were compared in an almost straightforward manner; these algorithms compared input characters to predefined templates, which meant that the characters were either recognized as an exact match or no match at all -- there was simply no room in the algorithms to account for variations such as tilts or writing styles. To try to accommodate these, another method was devised called "Recognition Using Correlation Coefficients" based on the cross correlation of input characters' transforms, but this approach ended up being far too tolerant (on the other end of the spectrum), causing the recognizer to erroneously mix up similar-looking characters like "B" and "8" or "O", "Q", and "0". In a continued

effort to overcome these problems, and as data science progressed, better methods were discovered, mainly involving the fields of machine learning. Various methods were devised involving everything from neural networks to hidden markov models.

Regardless of the algorithms I found, they all followed the same general approach, albeit in different ways. At the highest level, you had a page of written text which was scanned. Upon scanning, usually the image would be processed into a grayscale image, dubbed "gray level image processing," and from there the image would be segmented (i.e. spliced) into pieces where each piece contained no more than a single character. Then, certain *features* of the character would be identified (such as slope, stroke depth, etc.) and then extracted to form *feature vectors*, and from there be classified and then post-processed to combine them into words. TRIER et. al. summarized several existing pre-processing methods such as binarization / two-level thresholding, segmentation, and character representation conversion (skeleton format and contour curves) [2]. It went on to discuss various feature extraction methods, from "Zernike moments" to fourier descriptors, and onto classifiers using methods such as neural networks. There are many mathematical models to approaching this problem, and one of the most intriguing solutions I found was by Mathur et. al. which used not one neural network but several that were each tuned to slightly different writing styles then processed further by a genetic algorithm to select for the best output [3].

Regardless of the specific algorithmic details, it became quickly evident that a machine-learning approach was the way to go. However, none of the dozens of studies really focused on speed. Even the studies that discussed speed did so mainly in informal terms, using vague terminology that an algorithm can run in "moderate speeds" for example. It wasn't that

helpful, so we realize that in order to run an augmented spell checker in real time, we may need to spin up a few instances in the cloud to augment the hololens' hardware capability itself. Then, it'd be a trivial matter to live-stream camera input to the server farm, process the input, and return simple metadata back to the hololens which would be processed extremely quickly. This adds the additional burden of needing internet access (as opposed to being a purely offline solution), unless the hololens turns out to be capable of handling the workload, which is only exactly possible to tell when actually testing with the device while coding our application. Regardless, we will ultimately settle on a single ML-based recognizer approach, but balancing out fast offline recognition with higher-quality (albeit slower) online recognition will take trial and error.

CITATIONS:

1. Bassil, Youssef, and Mohammad Alwani. "Post-Editing Error Correction Algorithm For Speech Recognition Using Bing Spelling Suggestion." International Journal of Advanced Computer Science and Applications 3.2 (2012): n. pag. Web.
   Accessible link: https://arxiv.org/pdf/1204.0191.pdf
2. Proceedings: Seventh International Conference on Document Analysis and Recognition: August 3 to 6, 2003, Edinburgh, Scotland. Los Alamitos, CA: IEEE Computer Society, 2003. Print.
   http://ieeexplore.ieee.org/abstract/document/291440/
3. Vinod Chandra and R. Sudhakar, "Recent Developments in Artificial Neural Network Based Character Recognition: A Performance Study", *IEEE, 1988*.

4. IVIND DUE TRIER, ANIL K. JAIN, and TORFINN TAXT, "FEATURE EXTRACTION METHODS FOR CHARACTER RECOGNITION | A SURVEY," *MSU, 1995.*
5. Mathur, Shashank, and Vaibhav Aggarwal. Information Science and Computing(2008): n. pag. *OFFLINE HANDWRITING RECOGNITION USING GENETIC ALGORITHM.* Sixth International Conference on Information Research and Applications, 2008. Web. 2017.
6. "Using smartglasses for utility-meter reading"
   Written by: Depari, A,; De Domincis, C.M.; Flammini, A.; Sisini, E.; Fassanotti, L.; Gritti, P.
   http://ieeexplore.ieee.org/abstract/document/7133649/?reload=true
7. Accuracy of Electronic Spell Checkers for Dyslexic Learners
   https://www.dyslexic.com/accuracy-of-spell-checkers-for-dyslexic-learners/

Problems and Barriers:

Potential problems we see are difficulties when it comes to transferring programs and builds to the glass when we have it available (such as builds that work on our phones/emulators not working the same when on the Microsoft HoloLens.

Additionally, we expect to find some difficulty when comes to character recognition with handwriting, especially when it comes to the handwritings of different users. We will also need to keep track of how long the whole process will take for the user - a user shouldn't have to spend an inordinate amount of time having to check each word.

We also will have issues on suggesting the correct word depending on how severe the misspelling is. As we have learned, the severity of the misspelling can widely range for users who have dyslexia, therefore we might not be able to correctly spell check their words. One possible fix for this is to have some type of audio feedback on what the correct word should be.

Timeline:

For this project we decided to go with a pseudo-Agile/Agile-lite method of creating this project. As a team we have made a trello board where we will create tasks for the project. Group members will work on tasks as needed and add more in. We call this an Agile-lite as we don't follow all of the traditional Agile methodologies laid out in the Agile Manifesto. Our sprints are done in 1 week rather than 2 so we can match the UIC class schedule. We don't have daily Scrum meetings but rather will discuss in class and meet up in person on the weekends where we are all free. We don't fully felge out our tickets on the scrum board as we want to focus on "Get it done and move on" and we want to adapt our project as we discover new barriers and problems. This timeline only covers working on the project itself and not on the paper or any other assignments for this course.

Week In School	Goals	Issues
Week 7	Research what technology to use and any libraries that go with them.	None
Week 8	Set up programming environment on every group member's system	None(As far as we know)
Week 9	Work on dictionary database	

	backend. Start on how to make the misspelled word indicator.	
Week 10	Refine database and refine misspelled word indicator.	
Week 11	Have some working prototype working on an emulator.	
Week 12	Test out project on emulator.	
Week 13	Test out project on emulator.	
Week 14	Test product with actual device.	
Week 15	Polish up product.	