Willis Allstead
9/10/18
CS 425

1. Project Idea

My project idea is a machine-learning enabled Augmentative and Alternative Communication (AAC) app for mobile devices, specifically starting with iOS devices. Machine learning could be implemented in a number of different ways when it comes to AAC apps, but this one would focus on relieving a particular pain point of some users, and that is the difficulty of stringing words together to express a thought. An AAC device is one that allows someone that exhibits difficulty talking, for any reason. Before mobile phones, extremely expensive AAC devices existed for people that needed them. Essentially, these devices provided a group of buttons that could be pressed to express either one word or multiple words strung together. For example, "I", "need", and "milk" could be strung together. Once apps became available on mobile devices for this, the price for this technology was reduced. One thing that was not changed in this switch was the fundamental way ideas are strung together into a sentence.

What our app would bring to the table is using machine learning to form highly-personal and useful auto-completion strings for the user. This way, common phrases used by a user can be reached without having to go through a potentially large number of screens of symbols every time. And if a user wanted to use a common phrase, but change it just a little, they would be able to do that too.

The target users for this app would be people with autism, Down syndrome, cerebral palsy and other diagnoses. It would be a symbol-based AAC app. The top of the screen would show the sentence you are constructing. The bottom pane would show categories of symbols, such as "Home", "School", "People", "Things", etc. and within those collections could be either more collections or symbols. Each symbol would have a basic drawing for easy recognition by the user. The user would tap a symbol to add it to the sentence, and the "ideas" in the sentence would be draggable to change their order.

This idea was spawned when I visited a family friend a few years ago on a trip my family was on. I met their son who had Angelman syndrome. This meant he was unable to speak more than a word or two without struggling, but he was using an AAC device with impressive speed to express his words. It was amazing, but I couldn't help but notice that every time he wanted something he had to press at least 5 keys to string the symbols together. If he could have pressed 2 keys, for example, and had a good autocomplete option presented to him, he could have spoken significantly faster. Potentially he could even begin to hold actual conversations with his parents/teachers, and maybe that could lead to him developing faster.

Some AAC apps already exist, and a couple of them feature autocompletion. One example of this is called "Predictable" by Therapy Box. One key difference between my idea and this one is that this is a text-based AAC app. This is great for more capable users, but for those who need to rely on symbols, it is unusable. Another example of an AAC app is Proloquo2Go, which seems to be the most popular AAC app on the Appstore. Besides it's (ridiculous) $249.99 price on the store, the main benefit of our app over this one would be the addition of machine-learning enabled autocompletion.

2. Software engineering and my professional career plans

Currently, I am a Junior Web Developer at a marketing agency here in town. I've developed for the web since middle school and always loved it. For at least a year or two, I would like to continue working at my current company. The only downside of working there is the pay. Web developers make less than software engineers, and the work is arguably more fast-paced. I'm sure that depends on the company you work for but I've heard that the agency life is a fast one compared to bigger software companies. The educational requirements for this job were experience programming in HTML, CSS, PHP, JS, etc. and the ability to quickly pick up more languages as they came and to generally be good at communication and problem solving. There were no educational requirements whatsoever. With 0-1 years of experience, Glassdoor gives an average salary of $62,955/yr for a Web developer which is higher than my actual salary which makes sense since I am a Junior.

I am also interested in becoming a UX designer. I have always been interested in collecting data on user behavior to design better interfaces. A UX designer is essentially in charge of designing usable and understandable interfaces while keeping the product looking attractive to the user. An interface can make or break a product. And while this job is less programming-focused, it would be important to know how a product is programmed to design useful user tests and to understand how long things should take to be implemented. On Glassdoor, it gives an average base pay of $107,880/yr. With 0-1 years of experience it gives an average of $77,786/yr which seems like adequate pay for that job to me. Experience required includes knowledge of HTML, CSS, JS and also experience with Photoshop or Sketch. Some positions require a bachelor's in CS but some don't require anything past highschool.

The last job I am interested in is becoming a Software Engineer. Glassdoor gives an average base pay of $115,462/yr or $87,315/yr if you have 0-1 years of experience. I love software development for personal projects, and would like to take the knowledge I've gained as a Web Developer and utilize it in a more structured and process-oriented way. I look forward to learning more about this career in this class. Essentially all Software Engineer positions require A Bachelor's in CS or a similar major. A lot of the jobs also require 3+ years of programming in certain languages. This job would require me to follow engineering processes, as we are discussing in class, to work in a team to create a final product. It could also entail maintaining or evolving an existing product, and all the steps in between.

3. Software Quality

One app that I use nearly every day is Netflix on my iPhone. A lot of the app is developed and designed very well. For example, the way it immediately throws you into rows of lists of content you are likely to watch is a good quality for most cases. The user interface is consistent throughout the experience, and is almost unnoticed, which is a good thing. There is almost no perceived latency in using the app except for maybe a lack of network performance, which is to be expected from an internet-connected app. It even allows you to download some shows for offline viewing, so those in areas with poor connections or no connections can download the episodes ahead of time to avoid interruption. Another small feature of the app is the "go back 15 seconds" button in the media player. The developers and designers understand

the users of the app and what content they are consuming. They don't need to skip ahead much, like in YouTube for example, but they might have missed a small thing and would need to go back to re-watch it. Now when it comes to possible improvements, there are a few that stick out to me. Netflix somewhat recently switched from a 5-star rating system to a thumbs up or down system. The thing about this is that it gives you a less accurate way of rating your content. Not every rating should be binary. Now, there is no way of Netflix knowing my most favorite and least favorite shows. Instead, they just know the shows I either don't like or do like to some degree. Another cool feature could be a sleep timer. This is implemented in the Apple podcast app, where you can set a timer for let's say 15 minutes and if it reaches that time it stops the playback and your phone falls asleep. Sometimes I want to watch a show but don't want to be woken up if I fall asleep during it. This would ensure that wouldn't happen. And finally, a change that could be interesting to add is a "curated" section of the app. Many catalogue-type apps are moving in this direction (think the Appstore or Spotify) where actual humans that work for those companies are payed to create collections of content and display it in a very custom and pretty way. Machine learning is great for some stuff, but when catalogues of content have potentially tens of thousands of possible entries, sometimes your best bet as a user is to find a human that you trust and see what they like.