iSkin Application

Fall 2022

CSCI 6221 Advanced Software Paradigms

Ada Kilic, Chien-Huan Kao, Yu-Sen Chiu

Prof. Yih-Feng Hwang

Abstract:

iSkin is an application created for anyone who wants to know the ingredients in their skincare or cosmetic products. The skincare and cosmetics industry influences people to purchase and use products that may or may not be good for a customer's skin. Customer's then can suffer from the negative long term effects of the chemicals and ingredients in the products. Hence, it is crucial to know what chemicals and ingredients are in the skincare or cosmetic products that you use and to find the right ingredients for your skin. The iSkin application will allow cosmetic users to check the ingredients in products in realtime. Ingredients lists in products are generally written in a very small font and are not the easiest words to search for especially in the midst of shopping. With iSkin, the user can simply scan the ingredient list of a product. Then the application will read the ingredients and display the description of the ingredient and the ranking of the ingredient. The ingredient ranking is on a scale of 1 (Best Quality/Clean) to 10 (Worst Quality/Toxic). This way the user will be able to determine if the ingredient is good or bad for their skin. The iSkin application allows the user to simply check if the cosmetic product is safe by looking at the following criterias:

1) The amount of controversial ingredients (the greater the amount the higher the chance of suffering from skin issues).
2) The concentration (the higher the position in the ingredients list indicates higher concentration).
3) The adverse effects in the ingredients.

iSkin allows for users to choose the right products for users.

Introduction:

Cosmetics are a part of everyday life for many people. Since cosmetics have a large prevalence in society, it's important to be an informed and educated consumer. More and more people are looking for cosmetic products that have healthy and nontoxic ingredients. However, it is not simple to understand whether or not the products are actually healthy and non-toxic even if a brand claims that the product is clean.

For instance, many labels on products claim products are "clean", "natural", or "organic" however this is unreliable. This is mainly because there is no government agency responsible for defining or regulating the manufacture of cosmetics. For instance, the U.S. Food and Drug Administration (FDA) doesn't have the power to monitor cosmetics as closely as it does food and drugs. The FDA has some legal authority over cosmetics. However, cosmetic products and their ingredients do not need to have FDA approval. Hence, the FDA does not have to check if a product that claims to be "clean" actually has clean ingredients. Additionally, the FDA can't recall dangerous or toxic cosmetic products. Furthermore, it is important that cosmetic and skincare users are informed and purchase products that are healthy and safe since some ingredients in certain cosmetic products may be toxic.

Our application, iSkin, uses an ingredient hazard score scale from 1 to 10.
The scores on the scale reflect known and suspected hazards linked to the ingredients.

The hazard score of an ingredient is calculated using a weight-of-evidence approach that factors in all of the hazards or health impacts associated with the ingredient. Furthermore, the safest ingredients score well with a low hazard rating. To use our application a user simply can take a picture of the ingredients list of a product (or choose the picture from their camera roll). Then the application will scan all of the ingredients and show the full ingredient list of the product (in order) with the hazard score of each of the ingredients as well as a description of the ingredient. This way the cosmetic and skincare user will be able to make an educated decision about the product before making a purchase or using the product on their skin.

Related Project Works:

There are a few websites or applications that have a similar idea for iSkin. These usually have an ingredient or product database. However, there are so many cosmetic products and the databases of those websites or applications do not most likely include the majority of them. A few related websites/applications are listed below.

- Cosdna.com
- skincarisma.com
- https://www.ewg.org/skindeep/

Solution/Analyses:

For the application we decided to use Xcode IDE to develop our iOS application in Swift language with Storyboard. In our solution, we use text identifiers to detect ingredients that are listed on make-up products' packages. Then we find each ingredient's description and hazard score using web crawler technology to collect information from the webpage: https://www.ewg.org/skindeep.

After getting the ingredients page, we get their HTML code first. By doing so, we use Alamofire and URL content to get it. Then we will parse HTML code to get score and description. We use Kanna to parse the description by XPath. Since XPath is an absolute address in HTML code, we can find our information. On the other hand, we use Swiftsoup to parse scores. By observing HTML code, we find that the score will show on the Img src link. Then we can get a score from the web page. Finally we can show the ingredient list with a description and hazard score of the product that was scanned for ingredients. For each hazard score, we have our own picture to show. Please note that all of the images in the application were created using Adobe.

During the development of the program, first of all, we encountered that we do not have much experience developing the iOS application, so we spent a lot of time researching how to program an iOS application, especially using a web crawler in Swift. Secondly, the Alamofire

operates asynchronously, so we had to solve the concurrency problem as well. Our solution is to use a global variable, so we can share the information between classes when the search is done. Finally, The application spends a while searching for the data from the website, and this is usually not acceptable to users. Although we restrict some conditions, it takes some time as well. As a result, we will put this into future work.

Summary and Future Works:

The application helps solve a key problem for cosmetics consumers. Many people purchase cosmetics or skincare without considering the ingredients or trusting that the ingredients state “clean”. A consumer may not think twice about the skin cream they rub on their face, the eyeliner they apply before work, or the shampoo they lather their hair. These products, which seem safe, typically contain many ingredients that are difficult to pronounce and even harder to spell. Therefore, while shopping many consumers don’t think about or want to search for the ingredients in a product. Hence, having an application that they can quickly use while shopping can help eliminate this problem. The consumer can simply scan the ingredient list of a product then see how hazardous the ingredients are or whether or not the ingredients are good for his/her skin. Future work for our application will be to include performance, information, and interface. As we mentioned before, letting users wait for a long time is not acceptable, so we must improve our performance. We plan to build our own database, so we can avoid to search the website which takes a lot of time. Moreover, we can add more information such as what symptoms could possibly happen when users use hazardous ingredients. Finally, we want to redesign the interface, and make it more colorful.

Source Code:

```
//get ingredient description
    private func fetchRecruitInfo(url:String, completion: @escaping (String, Bool) -> Void){
        var des = ""
        self.startAnimating()
        AF.request(url).responseString { response in
            self.startAnimating()
            if let html = response.value { //send request
                if let doc = try? HTML(html: html, encoding: String.Encoding.utf8) { //get html code
                    var contents = [String]()
                    for value in doc.xpath("//div[1]/div[3]/section/p") {//use xpath to find html
                        guard let text = value.text else { continue }
                        contents.append(text) //use contents to carry value
                        //dump(contents)
                    }
                    let contentString = contents.joined(separator: "") // change to string
                    des = contentString
                    //print(contentString)
                }
            }
            self.stopAnimating()
```

```
            completion(des, true)
        }
    }

    private func setupVersionTextRecognizemage(image:UIImage?){
        //setupTextRecognition
        var textString = ""
        request = VNRecognizeTextRequest(completionHandler: {(request, error) in
            guard let observations = request.results
as?[VNRecognizedTextObservation]else{fatalError("Recieved Invalid Observation")}
            for observation in observations {
                guard let topCandidate = observation.topCandidates(1).first else{
                    print("No Candidate")
                    continue
                }
                textString += "\n\(topCandidate.string)"
                DispatchQueue.main.async {
                    self.stopAnimating()
                    //self.textView.text = textString
                    self.textString = textString
                }
            }
        })
```

```
    //add some properties
    request.customWords = ["cust0m"]
    request.minimumTextHeight = 0.03125
    request.recognitionLevel = .accurate
    request.recognitionLanguages = ["en_US"]
    request.usesLanguageCorrection = true

    let requests = [request]

    //create request handler
    DispatchQueue.global(qos: .userInitiated).async {
        guard let img = image?.cgImage else {fatalError("Missing image to scan")}
        let handler = VNImageRequestHandler(cgImage: img, options: [:])
        try? handler.perform(requests)
    }
}
```