INF558: Building Knowledge Graphs
Project Summary: Skincare Products and Ingredients Knowledge Graph

Wei-Fan Chen weifanch@usc.edu
Hsin-Yu Chang hsinyuch@usc.edu

May 1, 2020

1 Introduction

Thousands of skincare products are listed on beauty websites, choosing the right products is a time-consuming experience considering the ingredients and how they might affect our skin. Our main goal is to enable users to pick the right products without grasping information from different sources and also select suitable items based on users' needs.

The application allows users to search for skincare product information and its ingredients. Our interface also provides users to search a list of products without chemically conflicts with the skincare collection they have in hand.

To achieve our objectives, we build a full-stack application, Apache Jena Fuseki is used as our knowledge graph database, and Flask is the backend API to receive the requests from front-end Vue and query the knowledge graph on Fuseki.

2 Challenges

Our first challenge was to crawl data using *Selenium* and *BeautifulSoup*. To avoid hitting the website too hard and cracking our crawler, we set a random timer to delay the crawling speed. One task during crawling was that the website was loaded dynamically which made our initial crawler failed if the page was not loaded completely before we parsed. To ensure we retrieved the relevant data for each product and compound, we re-crawled our source page more than two times. 2,400 products, 6,000 compounds, and 600K customer reviews were collected from our sources.

For data cleaning, the ingredients were unstructured text with irrelevant content. The ingredients text were formed out of the description of key ingredients, full ingredient list, reminders and qualification of marks the product has, the full ingredient list is the only part we need, therefore, we eliminated other parts with their patterns.

One of the entity resolutions in our project was product linking since a product would have limited edition and regular packaging with different URLs on the website, however, they shared the same functions and ingredients. We use *rltk* to match information such as brand, product name, brand, ingredients, unit price. Among all products we crawled from sephora.com, 30 products were deleted after manually checked the potential duplications.

Another entity resolution we met was among compounds, a compound appeared in different ingredient lists with its synonyms, e.g. H_20, water, and aqua. We used reference matching with PubChem as the deduplicated reference table and computed the string similarity with the synonyms of the compound, roughly 0.5% of compounds were eliminated.

To recommend users' skincare products which are similar to the functions and customers' reviews of the selected product, jaccard similarity of product functions and cosine similarity of the average stars vector of each type of customer (ages and skin type) were aggregated to detect the matching products.

We also designed our own ontology of skincare products, including product, compound objects, and properties that capture the relationships which were needed for query. We also utilized existing ontology, such as *schema,* and *rdfs.* To generate the triples needed in our KG, *rdflib* python library was used to construct the ontology with the data we retrieved.

Our user interface was based on Apache Jena, Flask and Vue. We constructed the SPARQL query by taking users' inputs as constraints. Our system supports searching on a product, compound, and also filtering, users can find information before purchasing.

3 Conclusions

While dealing with the real-world data, we realized that gathering needed data and preprocessing it to a well-structured data were the two most important components throughout the pipeline. Moreover, validation before moving on to the next step is essential, either manually or automatically, since we have to ensure our results are implemented rigorously. Last, we believed knowledge graph-structured data makes the implementations on the searching and filter components easier. In sum, we hope our application can help users find the right products considering the ingredients and their needs before they purchase instead of just its big brand name.

Here is the link to our demonstration!