SI 650 / EECS 549: Final Project Report

Upasana Thakuria, Shashank Gupta, Riya Agarwal

Introduction

In today's world, cosmetics are a prime element in enhancing style, beauty and overall personality for the majority of people. According to Statista, the global cosmetic industry generated a revenue of US$80.74 billion in 2021 and is projected to earn a revenue of around US$131 billion by 2026. This booming industry is the result of more and more people becoming health-conscious and careful about the ingredients, quality and eco-friendliness (vegan, cruelty-free, etc.) of the personal hygiene, skincare, makeup, hair, fragrance, and oral hygiene products. Finding the proper product with ease and minimal effort (without wasting hours on the internet researching about products) is of great importance in such cases and will be of great use to a large number of people.

We have created a vertical search engine for cosmetic products search and a recommendation system for the same as well. There are numerous cosmetic recommendation systems such as that of sephora and Ulta beauty, but those systems are not fully automated and humans are required for the job. Also, these companies usually give biased opinions on products (eg. recommending a product from their sponsor more often). Our vertical search engine has no such bias and is fully automated.

Our search engine catered towards cosmetic and skincare products, and a recommendation system which recommends products based on the type of ingredients that the consumer wants. Since most commercial websites recommend products based on past user activity or highly rated products, our goal is to personalize recommendation from the perspective of the ingredients found in cosmetic products. Our approach worked better than we expected. We think this would be helpful for users to discover products which are suitable for their skin type, and provide more transparency into the products themselves, based on whether they are vegan, cruelty-free, etc. After an initial survey (from rating/review websites) of frequent problems users encounter with respect to cosmetic products, we realized that there is a wide range of population who suffer from varying skin diseases. Therefore, specifically for this user group, this system might be extremely useful, since instead of checking the ingredients of every product on the result page, the user would be able to receive personalized recommendations based on the ingredients which are suitable for their skin condition.

From this project, we learned how to improve a ranking pipeline by incorporating the learned predicted relevance scores by a deep learning model to result in better predictions. We used these results to create our vertical search engine and recommendation system.

Data

For the vertical search engine part of the project, we have primarily scraped the data from amazon website, we have extracted the important details which we think are useful for search engine and recommendation both. The features that we extracted are "Product Title", "Product Description", "rating", "price", "ingredients", "other products similar to the products", "Frequently bought items". Our general approach to scrap the data was to first use a chrome extension

named “Data miner”, each team member came up with 10 general queries to extract the urls of products associated with the query we wrote on the amazon search bar. We then wrote a python script to extract the information mentioned above. Few examples of these queries are :- “Eyeshadow”, “sunscreen”, “nail paint remover” and so on. More queries can be found out here. The document and information obtained from these general queries become the set of the document that we will be using in our project.

We also constructed a specific queries set (20 per team member). These are the queries which are useful for training and testing. These are the detailed queries which users would likely to search and thus align with our project theme of cosmetics. Some examples of these queries are:- “facewash for oily skin”, “cruelty free foundation”, “anti aging serum for youth” and so on. More such queries can be found out here. For a single specific query, we collected the relevant information from ~70 product urls. We collected all this data inside a csv file for convenience and better visualization. We then hand annotated this data. For simplicity, we rank the documents for these queries set using 3 ranking function in PyTerrier i.e. BM25, TF-IDF, and PL2. We asked each of this ranking function to extract 50 relevant documents and then we took union thus collecting ~55 relevant documents per queries. The annotation is done on a 5-point scale, 1 being least relevant and 5 being the most relevant. This entire process of scrapping, collecting, ranking and annotating took us 2 weeks to complete. You can find the annotated data here.

Statistics for Search Engine	
General Queries Set	30 (10 per person)
Specific Queries Set	60 (20 per person)
Number of Documents	2801

We are also attaching detailed statistics after indexing from PyTerrier.

```
Number of documents: 2801
Number of terms: 4274
Number of postings: 70394
Number of fields: 0
Number of tokens: 105257
Field names: []
Positions:   false
```

For the recommendation system, we are using data from the kaggle link given here. This data is scrapped from Sephora and thus aligned with our theme. There are six categories which the data focuses on - “moisturizing cream”, “facial treatments”, “cleanser”, “facial mask”, “eye treatment”, and “sun protection”. It has roughly 1472 items in total. It also contains information about the brand, price, the rank, skin and ingredients. Our focus for this part of the project is about recommendation by ingredients. We think it makes more sense to recommend by

ingredients rather than the users itself. Every person has a different underlying condition and sensitivity which can not be generalized. Every individual person has a entire sets of different conditions, the range for which is vast, for eg. me and my friend might be able to use the same moisturizer but I may not be able to use a sulfate-parabens shampoo like she could. The variance of each person is too much which leads us to believe it would be better to recommend on the basis of ingredients rather than item-item collaborative type of recommendation.

Data downloaded from kaggle contains a lot of information but not in the format we would like to process. So the first preprocessing step we did is to make sure the entries are unique, we removed duplicates from the raw data. We lowercase all characters, remove punctuation and split the text into smaller chunks which can be further processed. We then took the ingredients column and tokenized it into a bag of words. We then used this data to create Ingredients-cosmetic Matrix (which is document term matrix in our case). Just to point out, there are 6 categories of data and there are four different skin types. We will be creating a recommender system with each category and each skin type color. Let us also summarize the statistics of recommendation data in the below table.

Statistics for Recommendation data								
Number of Instances	1472							
Categories of product (percentage)	Moisturizer (20%), Cleanser (19%), Treatment (17%), face mask (18%), eye creams (14%), sun protection (11%)							
Skin Types (Number of Instance)	Combination		Dry		Normal		Oily	
	Label 0	506	Label 0	568	Label 0	512	Label 0	578
	Label 1	966	Label 1	904	Label 1	960	Label 1	894

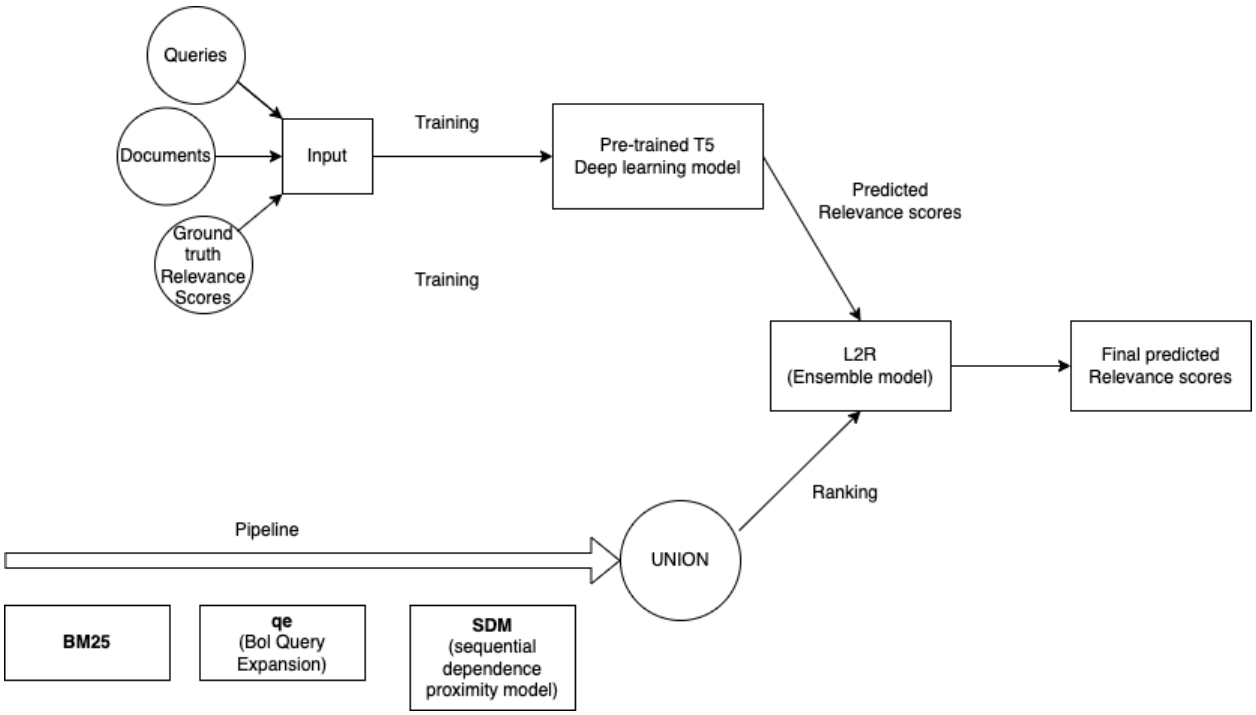
Related Work

Yoon et al. [1] worked on developing a recommendation system for cosmetic products using item-based similarity algorithms, and processed the attributes of the products using TF-IDF model, which then recommends the top 5 products most closely resembling the customer's query. Their aim was to help the users find the best products from the plethora of information available on the Internet and reduce the purchase time of cosmetic products. The authors implemented a recommendation system based on big data using R, which focuses on similarities of cosmetic characteristics, similar to websites like Amazon and Sephora. Li et al. [2] explore the application of machine learning and deep learning algorithm development on human face and skin intelligence recommendation platform. They use YOLOv4 for facial skin recognition, and develop an algorithm which accurately identifies the user's skin condition and recommend suitable products to avoid skin damage. Hsia et al. [3] propose a system which

analyzes the severity of facial skin and acne to recommend care products. They use computer vision technology to analyze facial images and perform skin type analysis. Tangsripairoj et al. [4] develop an android application SkinProf to provide knowledge about cosmetics and skincare products by helping consumers check the safety level of each ingredient and decide which ingredients they should avoid. This allows the users to search products based on the ingredients suitable for their skin type and select the most appropriate product. Chan et al. [5] propose a recommendation system for skincare products which combines multi-feature processing with ML classification technology and DL semantic segmentation technology. This system recommends skincare products based on the analysis of skin type and acne status. Iwabuchi et al. [6] implement a recommendation system based on the ingredients of cosmetic products. The authors implement the IF-IPF method to extract the ingredients of the products. They define the user attributes using a combination of skin quality and age, and recommend products with high cosmetic effects for each user attribute. Nakajima et al. [7] implemented a recommendation system for cosmetic products where they first extracted the cosmetic ingredients from review websites which are thought to have the best effect, and then recommend products to users which have these ingredients as their main ingredients. The authors use IF-IPF method to extract these ingredients from review websites.

Our project aims to construct a search engine specifically catered towards cosmetic and skincare products, in conjunction with a recommendation system which, based on the type of ingredients that the users want for their products, outputs suitable products. Although previous work has been done in this area, majority of people continue using commercial websites like Sephora, Ulta Beauty, Amazon, etc., which are profit driven and make it difficult for the consumer to find products based on the ingredients which might be suitable for the consumer's skin type. With the increase in cases of skin cancer as well as many other skin diseases, it's extremely important for the consumer to have a platform where they can continue their search for cosmetics and skincare products without any hindrance due to their underlying skin conditions. We aim to provide such a platform which allows the user to have a seamless experience of finding products suitable for their skin type, where the results would be free of any bias like race, gender, location, etc. Some related works also use computer vision technology to identify the skin condition of a user accurately and recommend products accordingly. We would like to explore this avenue for further enhancement to our project, and as an easier pathway for the users to interact with our system to convey details about their skin condition.

Methods



Our project is divided into two major milestones, creating a vertical search engine and recommending the user other cosmetic products based on ingredients.

For the vertical search engine part, we have used a pre-trained deep model to enhance the performance by a typical ranking function like BM25. As we can see from the flowchart above, the algorithm is divided into two parallel paths 1) Training a deep learning model based on T5 architecture, and then incorporating the feature from the deep network with the typical IR pipeline to create an ensemble model to later use it to train “Learning to rank” models.
The detailed methodologies of the vertical search engine is explained below.

a) **Data Collection :-** We have already touched base this in the “Data” section and will conveniently assume the reader is aware of the data collection, basic data processing and data cleaning.
b) **Preprocessing for Deep Learning Model :-** For our task the deep learning model takes the input sequence in “Q: {Query}, D: {Document}” format and outputs another text sequence “Relevance Score: {relevance_score}”. Since the T5 model is text-to-text transfer model and would expect encoded text sequence at the input and the output. That’s why we decided to stick with the above input and the output format.
Pre-processing step here involves creating query document pairs alongside relevance scores for the model to preprocess. While creating the sequence, we have applied basic cleaning of removal of punctuation. And for the encoding, we have used “***T5Tokenizer***” function preloaded from “***t5-base***” model, this function tokenize the input and the output sequence and thus giving us encoded input-output pair. This was fed to the

"***Datacollector***" class we had built for the T5 model to process the input output pair in batches. Also note that, we have tried using the T5 model as classification giving a label as the relevant score but the results aren't impressive enough. We have tried a bunch of the Input - Output pair but only the above version worked, you can find more details in the "Other things tried" section.

c) **Specifics about T5 Model :-** There are multiple pretrained models available on the huggingFace website, and we decided to go ahead with "***T5ForConditionalGenerator***" because it suits our input-output sequence theme. The weight is preloaded from the 't5-base' model. Once the data cleaning is done, the dataset is ready to feed to the T5 model for fine tuning. We trained the model for 3 epochs, using "***Trainer***" API from the transformer package. Once the training is done, we expect the model to generate the relevance score in the format discussed above in "data cleaning and pre-processing" step.

d) **Incorporating the deep learning prediction with the typical IR framework :-** For the fixed test set, we calculate the prediction from the T5 model and store it in csv for further use. We build a classical IR pipeline by considering the feature from BM25, and by doing query expansion and also combining the feature from the Sequential dependency model. An ensemble model is constructed using the union of all the above features by training two learning to rank models (namely, random forest, and fastrank). The final ranking score is constructed using the combined features.

We have summarized all this step in a flowchart for better understanding. Some more details about the implementation of Naive and Baseline are mentioned in the below points.

a) **Running dataset on Naive model**
    1. We used randomly generated scores for each document as a naive relevance to our dataset then we computed the ndcg scores after comparing with the true relevant scores.

b) **Running dataset on Baseline model**
    1. We used BM25 as our baseline model taking the annotated data as true relevance scores.

For our customized pipeline, we used BM25 score, sequential dependence model scored on BM25, tf idf score, pl2 score, and the relevance scores from the T5 model. Using this pipeline, we trained "learning to rank" models like random forests, FastRank, and LightGBM.

Recommendation System :-

For the recommendation system, we are using the Kaggle dataset to analyze the products based on their ingredients. The products, like Moisturizer, lotion etc. are also categorized on the basis of which skin type they can be used for, for example, for a particular moisturizer from lakme, we can check whether it can be used for normal, dry, oily and sensitive skin. Based on this, we are initially prompting the user to enter the kind of product they would like to search, along with their skin type. Once this is done, we get the subset dataset of the products, and get a tokenized list of the ingredients for the products in the resulting dataframe. Once this is done, we create the binary index matrix, where each row corresponds to the product, and the columns

correspond to the total list of ingredients we have. The matrix will be populated with 0s and 1s, based on whether a product contains that particular ingredient or not. Once this is done, we use the t-SNE model in order to visualize and calculate the similarities between the products. We can then plot this graph, and use the graph to visualize which products are placed closely together on the map, and which aren't. Using this, if a person searches for a particular product (for now limited to the products we have in the dataset), then they will be able to get similar products based on the ingredients of the product entered by the user.

Evaluation and Results

To evaluate search engines, we used MAP, NDCG, NDCG@5, and NDCG@10. We will also be comparing our results with two baseline approaches. This baseline method will set up the margin we require to beat.
Baseline :- BM25 ranking system.
Naive Approach :- For the naive approach, we will be generating a random number to pick relevant documents for the given query

The total number of queries we had for our project was 60. Following is the breakup of these for training and testing purposes:
Training set: 35 queries
Validation set: 10 queries
Testing set: 15 queries

The customized pipeline we used included the following features:

1. BM25 score
2. Sequential Dependence model scored by BM25
3. TF-IDF score
4. PL2 score

We have completed the baseline and Naive Approach and results are accumulated in the below table. We trained multiple models like random forest, lightGBM, and FastRank, and from amongst these, FastRank model outperformed our baseline BM25 model. These are evaluated on the same real data which we have described above in the DATA section.

Performance for search engine				
Method	MAP	NDCG	NDCG@5	NDCG@10
BM25	0.4564	0.5760	0.5727	0.5393
Naive	0.23	0.45	0.475	0.455
T5+Pipe -> Fastrank	0.4598	0.5783	0.5788	0.5399

We tweaked a number of hyperparameters that helped to get a good relevance score for the T5 model. Some of the hyperparameters were as follows:

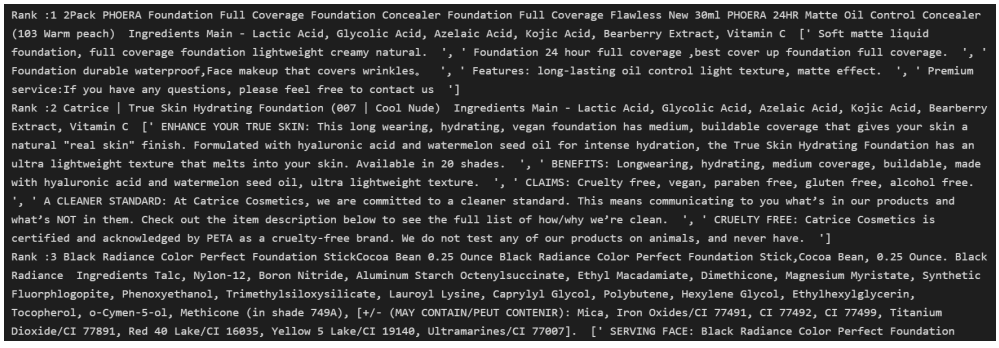
1. Length of the encoder
2. Learning rate
3. Optimizers
4. Different tokenizers
5. Using different versions of t5 like t5-base, t5-big, t5-3b.
6. Loss functions

For thoroughness of our project, we are planning to evaluate the recommendation system using Mean squared error. Baseline and the naive approach are mentioned below.
Baseline :- Item-Item Collaborative filtering.
Naive Approach :- Mean rating

For our recommendation system, we used a learning rate as 200 and random state as 42.

Performance for recommendation system	
Method	RMSE
Naive	3.32
Item-item collaborative filtering	2.26
Our Approach	2.12

Atlast, output on the input queries “Foundation” :-



Discussion

We had several ideas but our approach to beating the BM25 scores was by using the base version of the pretrained text-to-text sequential transfer model “T5” called “t5-base”. It is an encoder-decoder model which runs on a text-to-text input output. The input sequence is fed to the model using the ‘input_ids’ parameter. For our case, we had queries, documents and the ground truth relevance scores so for input_ids we concatenated the query and document to represent them both in the training data. The training of the T5 model for 3 epochs took 3.5 hrs

on our dataset. We used the evaluation metric F1 score to measure the training and validation accuracy. The training accuracy F1 score was 0.72 while the validation accuracy F1 score was 0.68. We later realized that our data was imbalanced with the majority of the relevance scores being in the range 1-3 on a scale of 1-5. We used weighted sampling on relevance scores to help combat the minority score category but it did not improve the scores. The baseline bm25 had a ndcg of 0.5760 and FastRank produced ndcg of 0.5783. We feel we could have pre processed the data more to make sure that each relevance score category was represented uniformly. Furthermore, we could use a more advanced text-to-text model for training on doc-queries and ground truth relevance scores like data-to-text NLG model. More fine tuning of the T5 model by tweaking the hyperparameters, also adding neural networks to the base t5 using transfer learning is something that can be tried in the future. The performance of the T5 model was very poor on unweighted training data which made us understand why most of the relevance scores produced by the model were in the lower range of 1-3. We understood how tweaking hyperparameters like maximum length of the encoder and giving different attention masks made a drastic difference to the performance, initially we had a f1 score of just 0.53. To the end-user, this means getting a more relevant vertical search result. The results for an end-user were ultimately quite satisfactory. In the future, this project can be developed more along the lines of deep learning and using more and more features in the ranking pipeline.

Conclusion

In this project, we proposed a vertical search engine for cosmetic products, and a recommendation system based on the ingredients the user wants. For our search engine, we used T5 reranking model to generate the relevance scores for our dataset, and then used these relevance scores, along with BM25 score, sequential dependence model scored by BM25, and query expansion scored by BM25 as features to create our pipeline. We finally used ranking models FastRank and random forests which used this custom pipeline, and compared their performance with our baseline model BM25. We found that FastRank performs better than the baseline model, while random forests has a lower performance as compared to the baseline model. We also implemented a recommendation system to recommend the products based on the ingredients of cosmetic products, as well as a basic GUI to interact with the model. The concept of receiving recommendations personalized to our skin type was very intriguing, and for future work, we'd like to explore more along the avenues of how we can use facial recognition to accurately identify skin type, color, and any skin condition, and get recommendations based on these features.

Other Things We Tried

While training the T5 model, it took us a long time to figure out how to configure the different settings like how to convert our dataset into the format that could be fed to the DataCollator. We also got confused whether to use the Collator module or DataCollator to create the data collator. It took us some significant time to figure out how the data collator was taking the query, documents and the concatenated input sequence of the query and document. Designing the DataLoader and DataCollator took a lot of time and effort. We tried a lot of combinations of configurations and parameters while designing the training script, one being the using different tokenizers like the BERT, T5 and the Auto Tokenizer. We were confused about how to feed the

data to the trainer and lost a lot of time figuring out how to do it and what format of input it accepted. When the T5 model was taking a lot of time to train, we almost lost all hopes on it. We tried a BERT model instead but then we realized the model used a different tokenizer and hence a different method to feed the input data. The preprocessing of the data took a long time as we had to change the format of the input starting from scratch. There was no proper documentation on the training of the T5 model and we required different parameters to put so we had to try out different combinations of inputs and output pairs. There were a lot of other things that we tried like using different types of tokenizers, many iterations of data loaders, figuring out what the labels to be to the T5 for training as we only had the ground truth relevance scores. We realized that we would have to tokenize the input and convert them in tensor form.

What You Would Have Done Differently or Next

We learnt a lot of things during this project, especially how to debug the code and delve deep into the code to know how a system is working. We wish we had opened the configuration files of DataCollators and dataloaders among the others way earlier to see how they are configured to take the input and what operations they intended to perform in depth. Debugging was 60% of our time, the rest being spent on annotating our dataset to allot relevance scores. Our data was quite imbalanced as almost 80% of the documents were rated with low relevance scores of 1-3 while very few were rated as high as 4-5. This uneven distribution of data caused a bias to predict a low score as opposed to a high relevance score. We could have resampled our data by undersampling the data having low relevance scores and oversampling the high scores data. We could have curbed this by using a weighted loss giving relevance to the minority scores.

Bibliography

[1] “A Big Data Based Cosmetic Recommendation Algorithm.” *Journal of System and Management Sciences*, 28 June 2020, 10.33168/jsms.2020.0203.

[2] Li, Hsiao-Hui, et al. “Based on Machine Learning for Personalized Skin Care Products Recommendation Engine.” IEEE Xplore, 1 Nov. 2020, ieeexplore.ieee.org/abstract/document/9394031. Accessed 19 Nov. 2022.

[3] System for recommending facial skincare products - MYU Group. (n.d.). Retrieved November 19, 2022, from https://sensors.myu-group.co.jp/sm_pdf/SM2335.pdf

[4] Tangsripairoj, Songsri, et al. “SkinProf: An Android Application for Smart Cosmetic and Skincare Users.” IEEE Xplore, 1 July 2018, ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8457178. Accessed 19 Nov. 2022.

[5] Chan, Hung-Tse, et al. “Smart Facial Skincare Products Using Computer Vision Technologies.” IEEE Xplore, 1 Dec. 2021, ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9689474. Accessed 19 Nov. 2022..

[6] R. Iwabuchi *et al*., "Proposal of recommender system based on user evaluation and cosmetic ingredients," *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*, 2017, pp. 1-6, doi: 10.1109/ICAICTA.2017.8090967.

[7] Y. Nakajima, H. Honma, H. Aoshima, T. Akiba and S. Masuyama, "Recommender System Based on User Evaluations and Cosmetic Ingredients," *2019 4th International Conference on Information Technology (InCIT)*, 2019, pp. 22-27, doi: 10.1109/INCIT.2019.8912051.