HackOn with Amazon

Prototype Submission

TEAM SALESPERSON

Chirag Jain
Shashwat Gupta
Shubham Srivastava
Sanket Agarwal

Theme: Shopping Experience (Offline & Online)

PROBLEM STATEMENT

Our main aim is to bridge the gap between online and offline shopping.

Major difficulties in online shopping are:

- There is no individual like a salesperson who can assist in deciding the right product for a consumer.
- To get a good idea of a product, one has to do the tedious job of going through lots of reviews about that product.
- For fashion products like clothing, sunglasses, etc., one is not able to judge how it would look on them.

SOLUTION

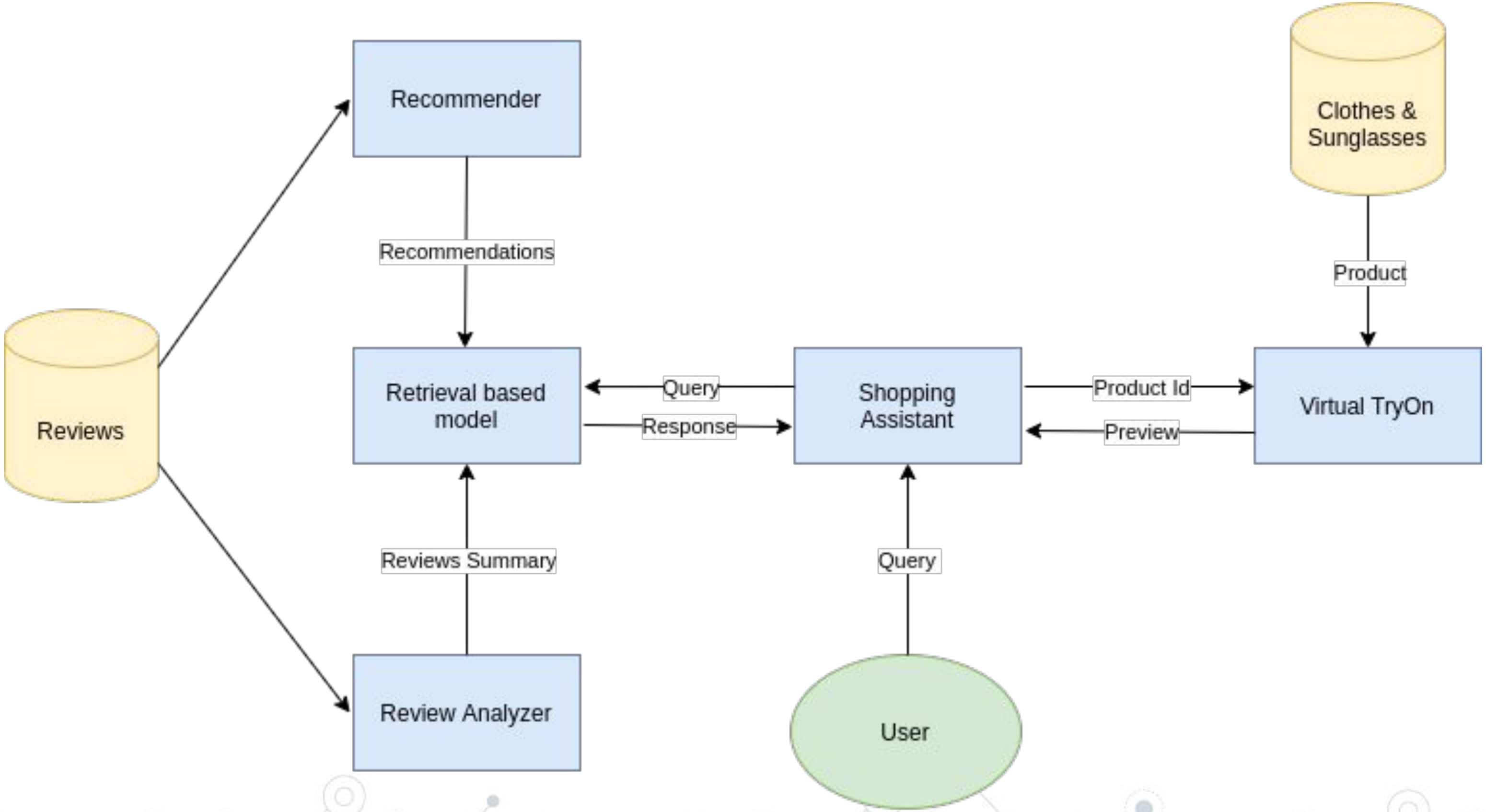
- The problem can be solved by Shopping Assistant , a chatbot, which can assist consumers in deciding the right product.

- It will give some suggestions to the consumer depending upon his needs.

- It will also provide a summary of all the reviews about that product, which will help the consumer to make a wise decision.

- It also helps the consumer to virtually experience fashion products.
  E.g. If a consumer needs to try a dress or a spectacle our shopping assistant gives him real time experience of how that product would look on him/her.

WORKFLOW


Recommender
Recommendations
Clothes & Sunglasses
Product
Reviews
Retrieval based model
Query
Response
Shopping Assistant
Product Id
Preview
Virtual TryOn
Reviews Summary
Query
Review Analyzer
User

METHODOLOGY

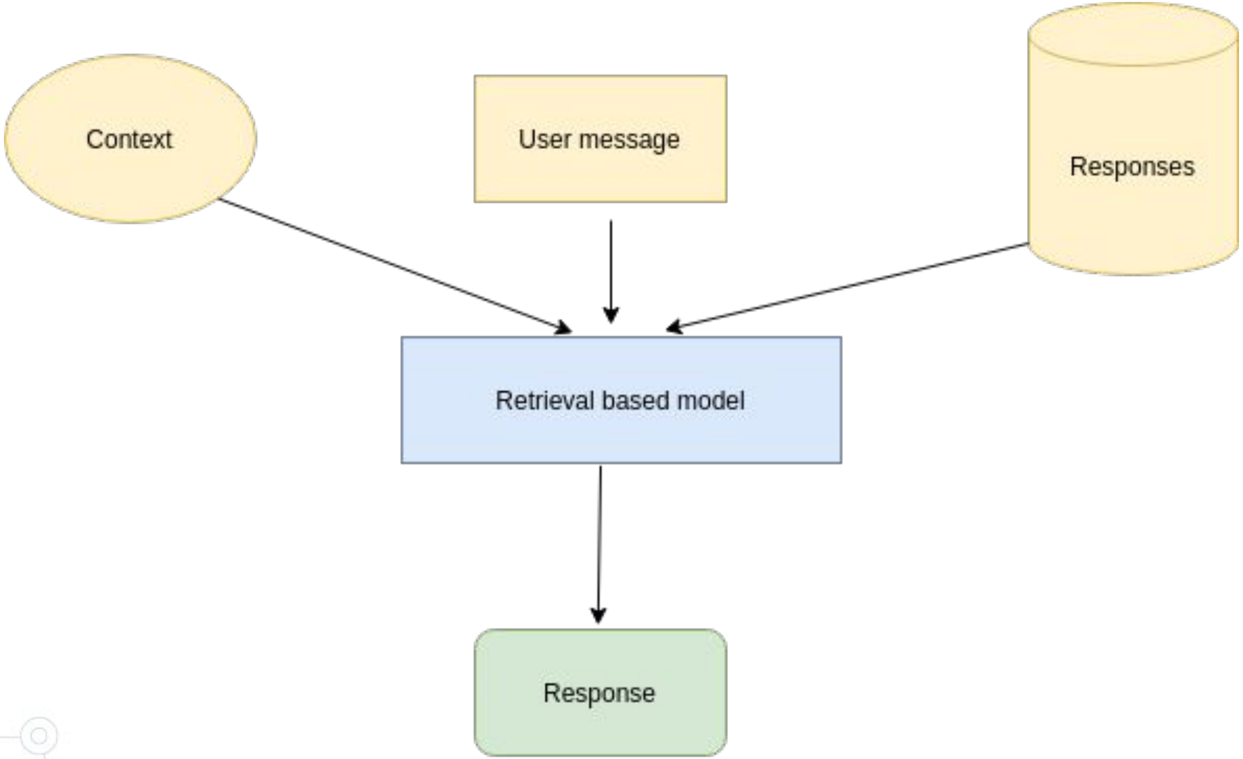
CHATBOT

- Works on a retrieval-based model

- Uses weighted TF-IDF and cosine similarity to quickly retrieve the closest response for a query.

- The responses were categorised mainly into 4 types:
  1. Product suggestions
  2. Summary of reviews of a product
  3. Virtual trial of wearables like T-shirts, sunglasses, etc.
  4. General talk (greetings, thanking, etc.)

RETRIEVAL BASED MODEL

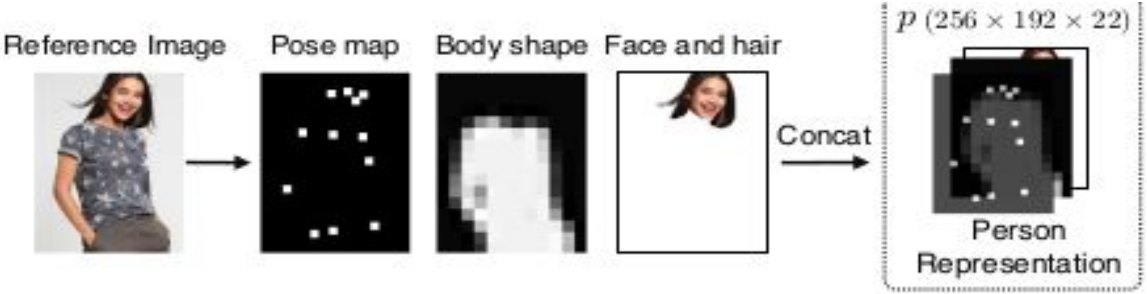


VIRTUAL TRY-ON

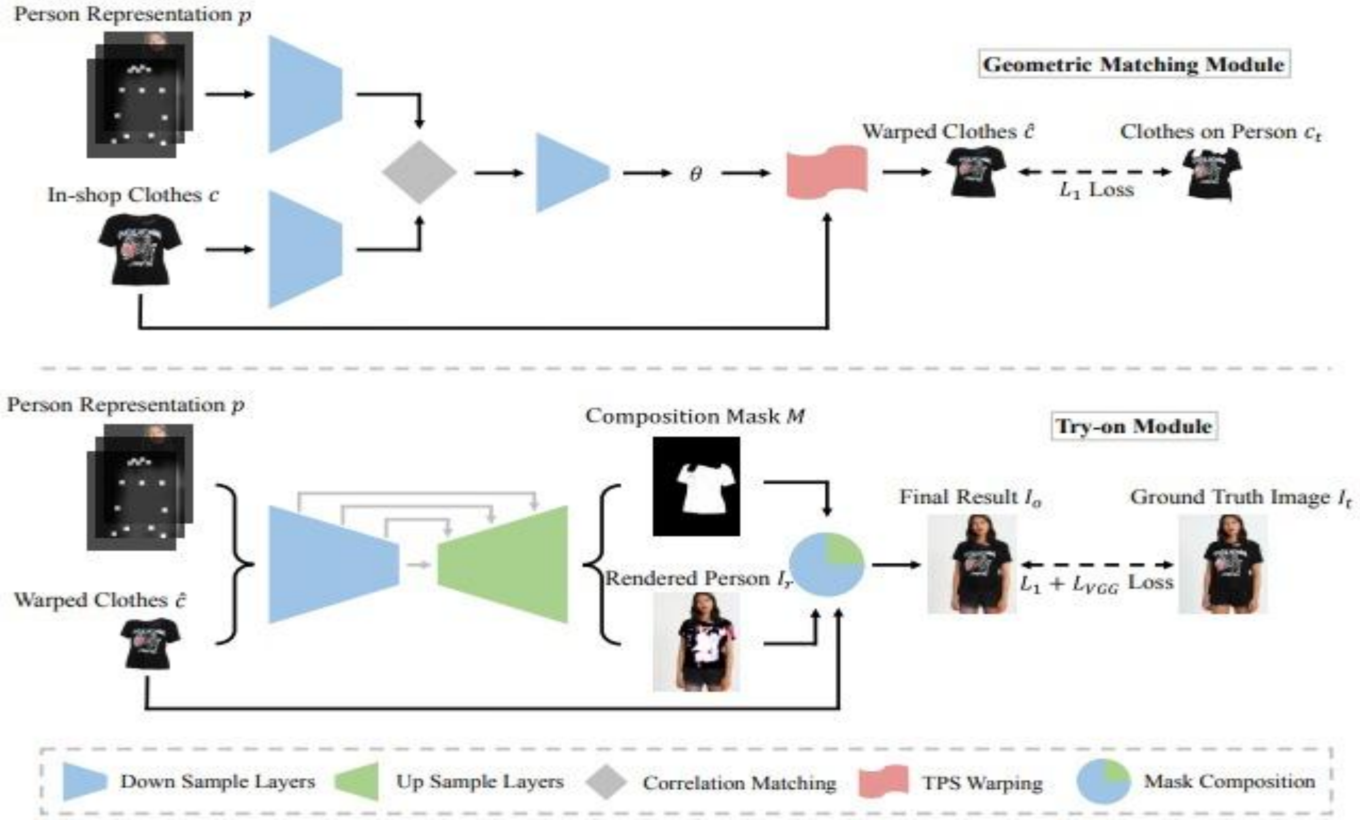
DATA PREPROCESSING

- The clothing-agnostic person representation includes :
    - **Pose Heatmap:** Heatmaps containing information about pose are generated using “openpose” library. These are collected from 18 key-points of body.
    - **Body shape:** A 1-channel binary mask covering different parts of body.
    - **Reserved Regions:** A RGB channel for preserving face and hair of body.

- This representation is fed into the Geometric Matching Module (GMM).



VIRTUAL TRY-ON ARCHITECTURE



The architecture mainly consists of two components :-

- Geometric Matching Module (GMM) : It transforms the target cloth to warped cloth so that it can be aligned according to the person’s body shape and pose.

- Try-On Module (TOM): It fuses the warped cloth with the target person and synthesizes the final try-on result.

GEOMETRIC MATCHING MODULE (GMM)

GMM is used to transform the target clothes c into warped clothes ĉ which is roughly aligned with input person representation p.

GMM consists of four parts :

- Two networks for extracting high-level features of p and c respectively using downsampling by convolution layers.
- A correlation layer to combine two features into a single tensor as input to the regressor network.
- The regression network for predicting the spatial transformation parameters θ.
- A Thin-Plate Spline (TPS) transformation module T for warping an image into the output $\hat{c} = T_\theta(c)$.

Try-On Module (TOM)

A Try-On Module (TOM) is used as generator to generate image as final output which is the try-on result of the desired cloth on input person.

- A TOM consists of encoder-decoder architecture like Unet in which a concatenated input of person representation p and the warped clothes $\hat{c}$, are fed simultaneously to render a person image I_r and predict a composition mask M.

- The rendered person I_r and the warped clothes $\hat{c}$ are then fused together using the composition mask M to synthesize the final try-on result I_o.

$$I_o = M * \hat{c} + (1 - M) * I_r$$

DATASET

- The dataset used was **MPV (Multi-Pose Virtual try on)** dataset.

- It consists of 37,723/14,360 person/clothes images, with a resolution of 256x192. Each person has different poses.

<table>
<tr><th rowspan="3">Resolution: 256 x 192</th><th>Train / Test Set</th><th>No of Samples</th></tr>
<tr><td>Training Set</td><td>52,236</td></tr>
<tr><td>Test Set</td><td>10,544</td></tr>
</table>

TRAINING

- The GMM module was trained using the pixel-wise L1 loss between the warped result $\hat{c}$ and ground truth c_t.

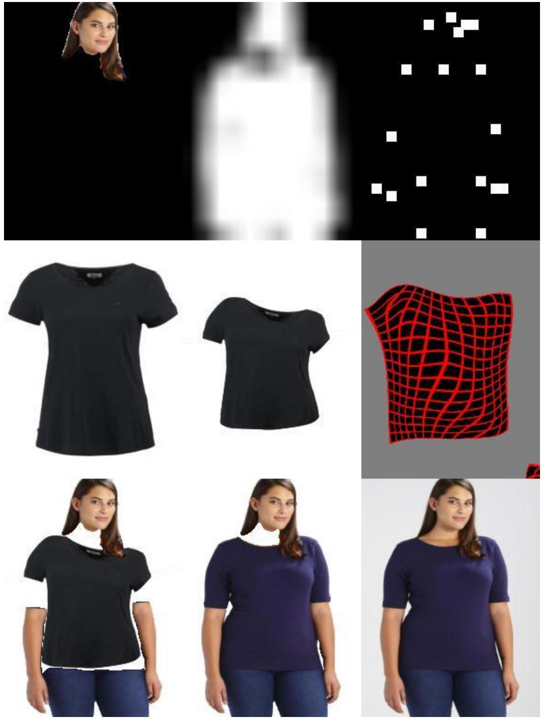
$$\mathcal{L}_{GMM}(\theta) = ||\hat{c} - c_t||_1 = ||T_\theta(c) - c_t||_1$$

- TOM is trained adversarially against the discriminator that uses the TOM result image I_o , input clothing image c, and person representation p as inputs and judges whether the result is real or fake.
- Optimizer used : Adam
- Final loss of generator on validation : 3.62001
- Final loss of discriminator on validation: 0.003821

VISUALISATION

Person

Cloth



GMM result

TOM result

VIRTUAL TRY-ON OF SUNGLASSES

- Uses Jeeliz library to detect the user's face and add to it the glasses frame in augmented reality (AR)
- Very low latency (almost in real-time)
- Can be easily extended for all other facial wearables like masks, jewellery, caps, etc.

REVIEW ANALYSIS

- Sample data was taken from an online shopping site which was then pre-processed using NLTK library to remove articles, prepositions etc
- An inbuilt tokenizer was employed to make sentences with relevant nouns which describe the consumer’s product experience.
- Subsequently, word count of each such relevant nouns was taken into consideration in order to calculate sentence score.
- Consequently, we receive the list of dominant sentences sorted by their sentence score which constitutes our review summary.

RECOMMENDER SYSTEM

- The module uses a sample data containing rating of products given by various users.
- The system works on collaborative filtering which employs Pearson Correlation Coefficient (PCC) to find similarity between a pair of users.
- The users having high PCC will be more closely related to that specific user and the corresponding products which were highly rated by them, will be recommended.

WORKING PROTOTYPE

Demo Video is available at :

https://www.youtube.com/watch?v=x_BFtcoaTks

Code along with setup instructions is available at:

https://github.com/cjchirag7/shopping-assistant

SCREENSHOTS

Available at:

https://github.com/cjchirag7/shopping-assistant#screenshots

TECH STACK

- **React.js** for website [It uses virtual DOM that is much faster and allows to create complex UI easily]
- **FastAPI** for web server [Fastest python web framework and easy to integrate Machine Learning model to the server]
- **PyTorch** for implementing virtual try-on [Due to its support for dynamic computational graph]
- **NLTK** for tasks related to Natural Language Processing. [As it has great pre trained models and corpus of data which makes text processing and analysis pretty quick and easy.]
- **Docker** and **Docker-compose** for containerizing the web application and the server [Ensures portability of the application]

EXTENT OF SCALABILITY

- Since, we have used docker to containerize our application. It just takes a few seconds to spin up a new container, to increase the capacity for our service. We can also use **Kubernetes** for **auto-scaling** the Docker containers across multiple hosts.

- Since, we have used data from JSON files. We can easily shift to a NoSQL database like MongoDB and create multiple replicas of the database and thus achieve **horizontal scaling**.

IMPACT

- Enhanced shopping experience
- Ensures Safety in the pandemic by reducing the need to go to offline stores.
- More Engagement on Platform
- Save retailers and customers from the expense of returns and exchanges
- Avoid immense installation cost of trial rooms, showrooms etc.
- Time-efficient shopping
- More Sales

FUTURE SCOPE

- Use of generative models in order to get more precise responses from the chatbot.
- Speech to text and Voice synthesis can be added to the chatbot for a better experience.
- A mobile app can also be developed, which will communicate to the same server.
- The script for storing the summary of reviews of a product can be run at regular intervals using a web worker.
- Multiple Language Support for the chatbot.
- GPUs can be used for further reducing the latency of virtual try-on.

THANK YOU