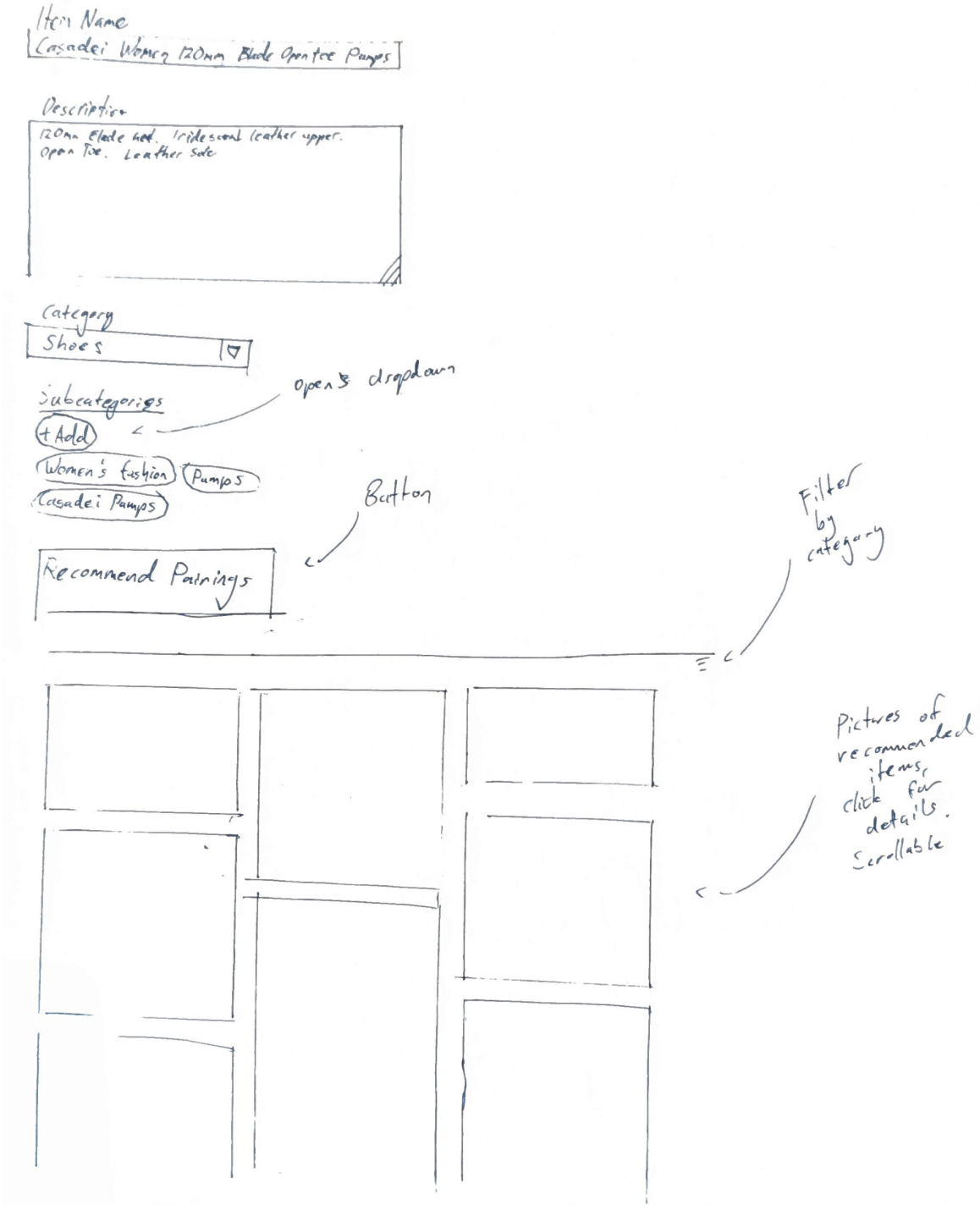
Project Design Document

Oscar Parrish – [z5476376]

Scope

A problem often faced by modern shoppers is hesitancy in exploring fashion and style. The clothing market is enormous, and through online retailer's consumers have access to a greater number and breadth of fashion items than ever before. But consumers can struggle to take advantage of this vast landscape of fashion because they may find it difficult to find good pairings and put together cohesive outfits. My proposed recommender system would take a description and details of a clothing item as input and return a number of proposed pairings. The user would be required to input the item's name, a brief description, and its category, which would be selected from a drop down menu. Optionally, the user can also choose to add zero or more subcategories to their item, also selected from a drop down menu, to help refine the search. The following page has a sketch of my design for the User Interface. Note that the images will not be generated until the 'Recommend Pairings' button has been pressed, and the button will not be clickable until the 'Item Name', 'Description' and 'Category' entries have been filled in. In order to simulate user interactions for the project, I can take the name, description, category and subcategories of their items from them directly and test them on the codebase. I can then show them the output images and details of the recommendations in order to gather feedback on the model's efficacy. This model will not face cold start problems, as it is drawn from a dataset scraped from the defunct website Polyvore. This site catalogued clothing items and outfit pairings, so the relevant data is already available. This does, however, present a problem of relevancy. Fashion is often fast moving, and this data, gathered from 2018, may be out of touch with modern trends. Hence, I further propose as the model is utilized by the public, we take the new items users input and weight their selections of the provided recommendations to generate new items and pairings for the model. This could be a financially viable system through affiliate sales links. By working with fashion retailers like H&M, the recommended items could include affiliate sales links to purchase the item through our partnered retailer. If this purchase was completed, we would then receive a portion of the profit as a commission for the sale.

User Interface



Dataset

This project will be drawn from the Polyvore Outfits dataset, scraped from the Polyvore website in 2018 by Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar and David Forsyth. This dataset can be found online on Kaggle at:

https://www.kaggle.com/datasets/dnepozitek/polyvore-outfits

This dataset comprises of 68,306 outfits and their metadata. The dataset is separated into a Json file containing outfits, a Json file containing items and a folder of images for each item. Each outfit object is comprised of one or more item objects, and an index which relates to a title and URL string for the outfit. Each item object is comprised of seven strings: 'url_name', 'description', 'categories', 'title', 'related', 'category_id' and 'semantic_category'. It should be noted 'categories' is spelt 'catgeories' throughout the dataset for an unknown reason. 'semantic_category' is a singular super category, selected from one of 206 unique classes, such as purse, sports bra and tank top. The 'categories' item is a list of more granular identifying terms, such as 'Women's fashion', 'Pumps' and 'Calypso St. Barth mini skirts.' The 'url_name' is broadly the same as the item's 'title' and will be ignored in this project. Each item also includes an index which matches to an image and can be displayed to the user. Below is the metadata for a typical item.


```
"134454029" : { 7 items
  "url_name" : string "calypso st barth metai lace"
  "description" :
  string "Balance your tough, utilitarian outerwear with charming lace. Figure enhancing gores surround this short linen skirt with structure. Wide elastic waistband and cotton voile lining make outfitting a breeze. Allover cotton embroidery thread and crochet bauble trim at zig zag sweep hemline lend unmistakable Calypso St. Barth charisma."
  "catgeories" : [ 4 items
    0 : string "Women's Fashion"
    1 : string "Skirts"
    2 : string "Mini Skirts"
    3 : string "Calypso St. Barth mini skirts"
  ]
  "title" : string "CALYPSO St. Barth Metai Lace Linen Skirt"
  "related" : [ 6 items
    0 : string "Lace skirts"
    1 : string "Short skirt"
    2 : string "White skirt"
    3 : string "Linen skirt"
    4 : string "Embroidered skirt"
    5 : string "Crochet skirts"
  ]
  "category_id" : string "8"
  "semantic_category" : string "bottoms"
}
```


This dataset is of a high quality, as it is a significantly large set of data scraped from a commercial website, where users could submit and vote on outfit pairings. As style is a

largely subjective topic, the best place to build a model to give style advice is on a website which catalogues public sentiment like Polyvore. When recommending pairings for an item, the system will perform two jobs. First, it will find items which are significantly similar to the input item, using the categories, semantic category, description and title provided by the user. Second, the system will find outfits in which these generated similar items are present and present the user with prospective pairings. This data is not without drawbacks, however. While a significant number of items are complete objects, a large number are missing key information, such as ‘description’ and ‘categories’. The only enforced values in the dataset are ‘url_name’, which can be used as a title, and ‘semantic_category’. This can make it difficult to gather enough detail on certain items to make informed decisions on similarity against our input. As each item is paired with an image of it, this problem could be mitigated by running an image classification model on the data to generate simple descriptions and subcategories for each item. However, this is out of the scope of this project.

Methods

To tackle the fashion pairing recommendation problem, several recommendation paradigms are considered: content-based, collaborative filtering and hybrid systems. Given the nature of the Polyvore dataset—focused on user-curated outfits rather than direct user-item interaction data—the primary recommendation strategy will be content-based filtering, augmented with collaborative signals from outfit groupings. A hybrid recommendation approach will be utilised, incorporating variants of both content-based and item-based collaborative filtering methods to address data sparsity and improve robustness.

1. Content-Based Filtering

The central mechanism of the system will be a content-based filtering model. This method uses the metadata associated with a clothing item—its title, description, category (semantic category), and subcategories (from the 'categories' field)—to compute similarity between items. The intuition is that similar items are likely to appear in similar contexts (i.e., outfits), so identifying items with closely matching metadata allows us to infer which other outfit items would pair well.

To implement this, each item will be represented as a weighted feature vector constructed from the four key attributes:

- **Semantic category (e.g. 'tank top', 'sports bra')** – a strong signal for garment type.
- **Categories (e.g. 'Women's Fashion', 'Mini Skirts')** – a mid-level descriptor useful for capturing styling context.
- **Title and description** – transformed into vector space using TF-IDF or similar embedding techniques (e.g. BERT sentence embeddings for richer semantics).

These features are then combined with predefined weights (e.g. semantic category: 0.4, categories: 0.3, title: 0.2, description: 0.1) to compute similarity scores using cosine similarity or Euclidean distance. The system then retrieves the *k* most similar items, identifies outfits that contain these similar items, and returns the non-overlapping items from those outfits as pairing recommendations.

This method is suitable because:

- It operates on the available item metadata, which is rich in fashion semantics.
- It requires no user preference history, aligning with the project's cold-start-immune nature.
- It can be efficiently computed and updated as new items are added.

2. Item-Based Collaborative Filtering via Outfit Co-Occurrence

While traditional collaborative filtering relies on user-item interactions, this project adapts the idea to item-item collaborative filtering using co-occurrence data from complete outfits. Items that appear together across many outfits are considered complementary. The core assumption is: if item A often co-occurs with item B in outfits curated by Polyvore users, then B is likely to be a good pairing for A.

A co-occurrence matrix is computed where each entry M_{ij} reflects how often item *i* and item *j* appear together in the same outfit. This matrix can be normalised to account for item popularity biases by Jaccard Similarity. Given an input item, we retrieve items with high co-occurrence scores and recommend them directly or prioritise them among candidate pairings.

This method leverages implicit human-curated signals in the dataset, improving the relevance and diversity of pairings. It also mitigates overfitting to metadata and expands beyond near-duplicate matches.

3. Hybrid Recommender System

A hybrid recommender system will combine content-based similarity with item co-occurrence signals for more robust recommendations. This will be done via Score-level fusion. This means the similarity scores from both methods (content-based and co-occurrence-based) are linearly combined.

This hybrid design improves coverage and compensates for weaknesses in each method alone: content-based filtering handles cold-starts well and offers diversity; co-occurrence handles subtle style affinities that may not be evident in text.

Evaluation

A wide variety of metrics will be utilized in evaluating the design, which is important for understanding the weaknesses and strengths of the implementation. They are listed below.

Precision

Precision measures the proportion of true positives against predicted positives in a machine learning approach. In this model, it can determine what proportion of the top K recommended items are valid pairings, based on the ground truth from the outfit dataset.

Recall

Recall measures the proportion of true positives against the total true labels. In this model, it measures what proportion of valid pairings the model served to the user. This is a helpful metric in determining coverage and ensuring diversity in output.

F1-Score

The F1-score is the harmonic mean of the precision and recall. While not providing any new information in and of itself, it is a useful metric at a glance for balancing precision and recall within a model.

Normalized Discounted Cumulative Gain

Measures the ranking quality of the ordered output items served to the user. Maximising this metric will drive the system to rank more coherent items, those that appear in the most similar ground truth outfits, higher. This means the user is served the best candidates first, and candidates become weaker as they scroll further through the recommendations.

Diversity

Measures the dissimilarity between recommended items. This is a highly important metric in providing a balanced recommendation to the user. For instance, imagine the system decides that the best outfit pairing for the user input is a black jacket. In the dataset we have dozens of black jackets, and the system would readily serve rows of very similar items. To account for this, we measure and control for diversity, which ensures that the user is recommended a wide array of potential pairings across a number of categories.

Coverage

Coverage measures what proportion of the item catalogue is recommended over time. This is another metric which aids in determining the diversity of the system. Over time,

we should ideally see a large proportion of our total catalogue recommended. We should also see the most popular items gradually shift over time as new input items are taken in. If we are not seeing this behaviour, it is likely the system is overfitting on popular items and recommending them even when they are not the most ideal pairing.

Click-Through Rate

The click-through rate measures what proportion of the recommended items the users select to view the details of. This is a useful way of gathering user feedback without prompting users with annoying pop-up surveys. This metric tells us the degree to which users are engaging with the recommendations, which in turn gives us our most clear of the recommendation system's efficacy.

Adaptivity

Adaptivity measures how the system shifts over time as new items are added by users. This is important for determining how effective our model is at integrating fresh data into the existing dataset. We should begin to see new items recommended as the item catalogue is expanded and new pairings are solidified.

Computational Requirements

As we are using a kNN approach to generate recommendations, we have a problem with processing. kNN is not a pre-trained method and dynamically places items within the existing data set. This means that the model is easily expandable, and the new user generated items can be added instantly as they are input. The trade-off is that the generation of recommendations may be slow and will only grow slower as the number of items balloons. Users may have to wait multiple seconds after the button is pressed for the recommendations to generate. This is already on the cusp of acceptability. The time it takes to generate recommendations must be a carefully controlled metric. The longer the user is forced to wait, the greater the risk of them simply clicking off the website.

User Study Design

As part of the user study, users will submit online questionnaires, which have been filled with the item name, description, and categories. We will then run the model on these items individually and present the recommendations in a list for user feedback. We will take note of the user's engagement with the recommendations, the proportion they consider accurate, the proportion they would consider purchasing and their overall sentiment. With a sufficient amount of user feedback, we can determine not only the efficacy of the overall system, but also the specific items which may cause problems.

By examining the submitted items which correspond to the lowest user satisfaction scores, we can attempt to account for the specific weaknesses in the model, such as class imbalance.

Timeline

Week 6

The first week of the project will comprise the preliminary exploration of the dataset and see work begin on the kNN model.

Week 7

The kNN model will be in a stage where it can be prototyped at this stage. The system should be able to take in a new item as an object and return a number of similar items. We are only generating similar items at this stage, rather than recommending item pairings.

Week 8

At this stage the model will be able to find similar items and recommend pairings from the ground truth. When given a new item, it will be able to return a selection of recommended pairings and the relevant details to the user.

Week 9

The system will now be able to take items used as input into its dataset for future use. It is able to dynamically expand as more items are added by users and generate new pairings. The model is now in its completed state, and we will perform a user study in the latter half of week 9.

Week 10

The project report will be written as relevant in the preceding weeks, but week 10 will see it completed with the final metrics from out user survey and all the relevant data from our completed model.