Mobile Recommendation System with Query-Driven Intelligence

Addanki Sanjeev Varma 210050003 | Akkapally Shasmith Krishna 210050008 | Gorle Krishna Chaitanya 210050057 | Vutukuri Vinay Mohan 210050167

Abstract Mobile phones have become vital in basic need of individual. The advances in technology, makes it necessary to satisfy different functional needs of users. Therefore, it is important to suggest mobiles for users according to personal preference and various constraints of mobile phones. A mobile recommendation system is proposed by integrating various requirements from users. This project involves the development of a recommendation system that utilizes server-side queries to analyze user input and subsequently suggests optimal mobile phones using recommendation alogrithms.

1 Introduction

Mobiles have become a basic need of individuals, as a communication and entertainment device across the globe. The endless increase in the options space left the customers in a dilemma of choosing the right mobile for them. The major factors that influence customers in selecting a mobile phone to use include brand, RAM, battery life, storage, camera, display, processor, and price. We used these features to develop a recommendation system. Apart from that, we also developed an additional feature in this recommendation system: We allow users to input a query describing the phone they want and this system also takes that into account for recommendations.()

2 Data Preprocessing

For every ML project, collecting datasets and preprocessing them is an important aspect. In this project, we mainly used 3 datasets.

The first dataset is of mobile devices, comprising 1020 samples, with a focus on key features such as Mobile Model Name, Price (INR), Average Rating, Sim Type, Processor, RAM, Battery, Display, Camera, and Operating System. Leveraging advanced Natural Language Processing (NLP) techniques, we meticulously transformed the original textual data into a rating-based format. Each feature is assigned a numerical value between 1 and 5, reflecting its specifications. These ratings are handcrafted which are inspired by diverse sources within the mobile technology domain. Notably, Card Compatibility, deemed extraneous to our analysis, has been systematically excluded from the dataset.

The second dataset is user-phone ratings. Unfortunately, we did not find any dataset that matches with our dataset of phones, so we have generated our own dataset using the average rating of phones from above dataset. We have assumed up to 1000 users for each phone, so each phone has been randomly assigned no of users 0 to 1000 and then used normal distribution with its mean as the average rating given. We then made appropriate changes to convert it into a 5 rating level.

The third dataset is a query to feature vector dataset. Again, we could not find a dataset regarding queries and this too was generated by us. We have collected different queries with the assistance of ChatGPT, and Bard and generated a dataset of queries to feature vector dataset comprising of n = 230. Each feature is rated from 1 to 5, we have used one-hot encoding to represent it. We have 9 features for each mobile phone, so we have 45 size feature vectors. These feature vectors are used for training the classifier model developed for query processing.

3 Model Overview

Let's walk through the model and dissect its individual components. Initially, users input their queries into our custom interface. The provided text undergoes NLP techniques, including tokenization and lemmatization, for effective processing. Subsequently, we extract a feature vector that corresponds to the user's query. This feature vector becomes an input for our recommendation algorithm. The recommendation algorithm generates a sorted list of mobile devices, prioritized based on their relevance to the user's query. To further enhance the precision of recommendations, a final filtering stage is implemented. This additional step considers specific cases to fine-tune the recommendations, ensuring a more refined and tailored output.

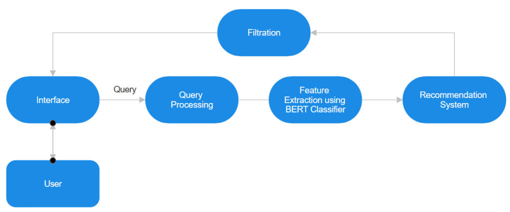

Figure 1: Model Overview

4 Query Processing

When users submit queries about gadgets, our chatbot employs various NLP techniques to enhance understanding. Initial steps involve tokenization and lemmatization to ensure consistency in processing user inputs. Specifically, we aim to extract crucial details such as brand names from these queries.

To tackle the challenge of plural forms, we've implemented a lemmatization step. This helps standardize the words in their singular form, ensuring a more robust comparison with the brand names stored in our "brands.csv" file. This preprocessing ensures that our chatbot can effectively recognize and respond to user queries.

Moving beyond brand extraction, users often specify price preferences in their queries. To address this, we've integrated the "word2number" Python package, enabling us to identify numerical values present in the queries. To refine this further, we've introduced a custom currency mapping that accommodates diverse representations such as 'K' for thousand and 'lakhs' for lakhs. By associating these variations with their respective values, including USD or INR, we can accurately calculate the final integral price in Indian Rupees.

4.1 Switching from BERT to Basic NLP Methods

We had some difficulties finding having all the data we needed for our project queries about different brands and prices in various formats— it. In our first try, we used a method called sentence encoding on a specific set of queries about brands and prices. It worked well for those, but when we tried it with new queries, the results weren't very accurate. So, we decided to switch to simpler Natural Language Processing (NLP) methods for better accuracy and adaptability.

5 Feature Extraction

A deep learning model is implemented with base model as **BERT-Base Classifier** and adding a fully connected layer. BERT-Base, or **Bidirectional Encoder Representations from Transformers** in its base form, is a pivotal natural language processing model developed by Google. It excels in understanding context bidirectionally, capturing nuances from both left and right directions in a sentence. It is built on the Transformer architecture and benefits from efficient parallel processing which has been pre-trained on extensive textual data, enabling it to learn comprehensive language representations. Bert outputs embeddings for different tokens parsed by its tokenizer and the overall summary of this contextualized embeddings of all the tokens in a sequence into a single vector representation known as **Classification token embedding** (often known as *pooler output*). This embedding generated is used for tasks like classification of text. Hence the *pooler-output* of the Bert-Base model is extracted and input into a fully connected layer with output size 5. An appropriate non-linear activation function is also added. **CrossEntropy** is used for training and evaluating

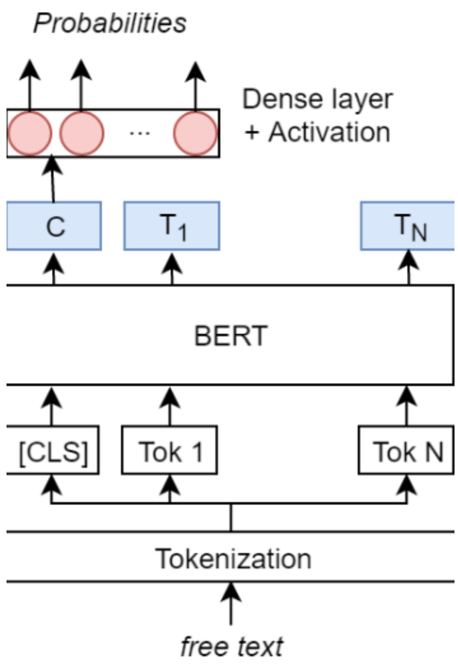

Figure 2: Bert Classifier

5.1 Attempts

In the first attempt of this idea, we used the above classifier model for each feature classifying each of them into 5 classes. We have got better results than using NLP techniques of finding synonyms using cosine similarities and assigning them to different ratings based on a base word (naive clustering idea without using any ML). However we have pointed out that this may not give the best results as each feature is being independently trained, but in reality, there can be dependencies between the features. We implemented only a single classifier with 45 classes and observed that it gave better results than before. We also implemented a classifier with two linear layers instead of 1, but we observed that it did not get good results. We suspect that the dataset is insufficient for this classifier (as a number of parameters would be increased) to learn and hence is not able to predict correctly. With a small dataset, we were not able to generate validation loss correctly and hence made our judgements based on solely training loss. The validation Dataset of our model is very small that validation loss abruptly changes and hence not included.

5.2 Results

Here are some results of the above classifier:

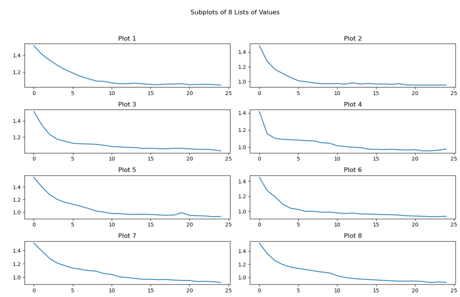

Figure 3: 9 Bert classifiers each of 5 classes for output

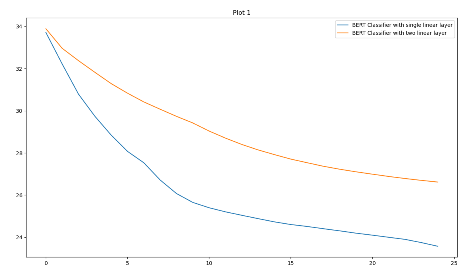

Figure 4: Bert classifier using 45 classes for output

6 Recommendation

Output from the feature extractor is used as the user's preferred features in a phone. In general, there are two types of recommendation algorithms: **Content-Based Filtering** and **Collaborative-Based Filtering**.

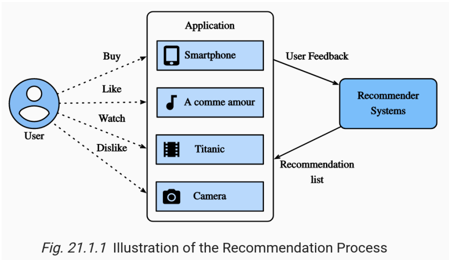

Fig. 21.1.1 Illustration of the Recommendation Process

Figure 5: Recommendation Process

6.1 Filtering Algorithms

In **collaborative filtering**, a matrix is created to represent the historical interactions between users and items. Entries in this matrix indicate user ratings, clicks, or other relevant interactions. It then calculates similarities either among users or among items. User-based methods find users with similar preferences, while item-based methods identify items with similar patterns of interaction.

In **content-based filtering** each user is associated with a profile that encapsulates their preferences, often derived from historical interactions or explicit feedback. Similarly, items are characterized by their attributes or features. In our recommendation system, features are price, processor, camera, storage, RAM, etc.

A matching algorithm is employed to assess the compatibility between user profiles and item features. Common methods involve vector representations and similarity measures, such as cosine similarity. Based on the matching scores, the system generates recommendations by selecting items with features that align well with the user's preferences.

In this recommendation system, we used content-based filtering as our main algorithm. In our project, collaborative filtering can not give true results as the data generated by us is random. We used **Matrix Factorization** algorithm for our recommendations.

6.2 Matrix Factorization

The model factorizes the user-item interaction matrix (e.g., rating matrix) into the product of two lower-rank matrices, capturing the low-rank structure of the user-item interactions.

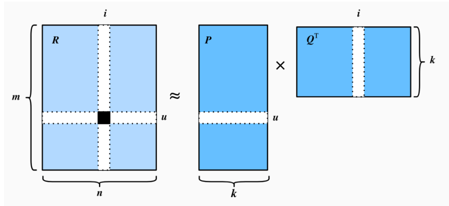

Figure 6: Illustration of Matrix Factorization model

Let $R \in \mathbb{R}^{m \times n}$ denote the interaction matrix with users and items, and the values of R represent explicit ratings. The user-item interaction will be factorized into a user latent matrix $P \in \mathbb{R}^{m \times k}$ and an item latent matrix $Q \in \mathbb{R}^{n \times k}$, where , $k << m, n$ is the latent factor size.The predicted ratings can be estimated by

$$\hat{R} = PQ^T \tag{1}$$

where the predicted matrix $\hat{R} \in \mathbb{R}^{m \times n}$ has same dimensions as original R matrix.One major problem of this prediction rule is that users/items biases can not be modeled. For example, some users tend to give higher ratings or some items always get lower ratings due to poorer quality.To capture these biases, user specific and item specific bias terms are introduced. Specifically, the predicted rating user u gives to item i is calculated by

$$\hat{R}_{ui} = p_u q_i^T + b_u + b_i \tag{2}$$

Objective Function using for training is mean squared error between predicted rating scores and real rating scores with regularization term as well.

$$\underset{\mathbf{P},\mathbf{Q},b}{\operatorname{argmin}} \sum_{(u,i) \in \mathcal{K}} \| \mathbf{R}_{ui} - \hat{\mathbf{R}}_{ui} \|^2 + \lambda (\| \mathbf{P} \|_F^2 + \| \mathbf{Q} \|_F^2 + b_u^2 + b_i^2)$$

For more information about recommendation algorithms and deep learning, follow the link (?)

6.3 Implementation

We have differentiated users as **new users** and **existing users**. For **new users**, we do not have any data about their ratings before, and hence feature vector from the query of the user is directly used for recommendation. We used **cosine similarity** between the feature vector of the user and the feature vector of the mobiles from data. We then extracted the scores, sorted them, and generated the list priority-wise for additional filtering.

For existing users too, we used the above method of generating a list sorted in priority. We also applied matrix factorization and generated **user-latent vectors** and **item-latent vectors**. This is then used for generating a list of user-based preferences by using of user-latent vector with all the item-latent vectors. Both these lists are combined by normalizing the scores in both lists and taking **weighted averages** of both scores. In this way, we are accounting for both the user's history or past behavior and the user's query or current behavior. This combined list is sent for an additional filtering process.

We can observe that Matrix Factorization does linear operations of dot product and hence it only captures linear relations. To increase complexity we can try using neural networks with ideas of autoencoder networks for non-linear relations between the user-item interactions. We did not do this in our project because the data was ranfdomly generated by us.

7 Filtration

Not every model is perfect. Likewise, some results may not be true by our recommender system. So for perfect and guaranteed results, we also have a filtering system that focuses on two main queries from users that are commonly given:

- Brand-based query- Best phones in Redmi
- Price-based query- Best phones under 40K

These types of queries are taken care of in this additional filtering where we extract the brand and price using the above-mentioned techniques. We use entity recognition for brand/phone names using word tokenization and price using word2num for recognizing the numbers given as text by the user in the query. The system extracts information of queries and filters in the synonyms to ensure the price range and features requested. There's also another stage of filtering in which the system categorizes the mobiles into **best**, **good** and **bad** based on their user ratings. It also categorizes words such as **lower/lesser** and **higher/more** and such synonyms for price range filtering which ensures mobile options are aligned in a specific range.

8 Interface

Finally, we have also developed an interface for this project which gives user an opportunity to post his query into the system. This interface was developed with the help of a library API known as **Streamlit**.

9 Conclusion

This model serves as a prototype for a recommendation engine with extensibility to various recommendation scenarios. The core elements of this project will be helpful in developing **personalized recommendation chatbots**, showcasing their potential utility. By employing diverse techniques, coupled with a robust dataset, the system's performance can be further enhanced. The focus of our implementation has been on mobile devices, recognizing their pivotal role in our lives. Our aim is to optimize user satisfaction by assisting them in making informed decisions when selecting the most suitable mobile device.

10 Related Work

The papers cited implemented recommendation system for mobile phones in different ways. Some ideas from these papers are used by us such as Content-Based Filtering (?), Data Processing

Results

cheap phones in samsung brand

1. samsung galaxy a33 5g: This is the best phone with a price of 25999 and an overall user rating of 4.1
2. samsung galaxy f42 5g (8gb ram + 128gb): This is the best phone with a price of 22999 and an overall user rating of 4.1
3. samsung galaxy a23 5g (8gb ram + 128gb): This is the best phone with a price of 24999 and an overall user rating of 4.05
4. samsung galaxy m32 (6gb ram + 128gb): This is the best phone with a price of 14490 and an overall user rating of 4.0
5. samsung galaxy a23 5g: This is a good phone with a price of 22999 and an overall user rating of 3.95
6. samsung galaxy f24 5g: This is a good phone with a price of 14999 and an overall user rating of 3.9
7. samsung galaxy a13 (6gb ram + 128gb): This is a good phone with a price of 16499 and an overall user rating of 3.9
8. samsung galaxy a24: This is a good phone with a price of 17999 and an overall user rating of 3.85
9. samsung galaxy m32: This is a decent phone with a price of 12364 and an overall user rating of 3.8
10. samsung galaxy m13: This is a decent phone with a price of 12999 and an overall user rating of 3.75 Please let me know if you want to know more about any of these phones

i want phones in apple under 80K

AI is thinking 🤔 ...

1. apple iphone 11 This is the best phone with a price of 38999 and an overall user rating of 3.65
2. apple iphone 14 mini This is the best phone with a price of 69990 and an overall user rating of 3.5
3. apple iphone se 2020 This is the best phone with a price of 39900 and an overall user rating of 3.15
4. apple iphone se 4 This is the best phone with a price of 49990 and an overall user rating of 3.0

Results from our Interface

```
queries = [
    "Looking for good cam phones",
    "Recommend phones with good battery life",
    "Suggest phones with both good camera and battery",
    "Smartphone with excellent camera performance",
    "Battery life comparison for latest phones",
    "Phones with top-notch camera technology",
    "Best battery backup phones",
    "Mobiles with superior camera quality",
    "Budget phone with ok battery life and average storage",
    "Phone with good ram and storage",
    "Need the best camera and top-notch battery life",
    "Affordable phone with no storage issues",
    "Looking for a phone with low as possible price and no problems",
    "Recommend best expensive phones",
    "Recommend best phones",
    "Phones with good display",
    "Best phone under 20K",
    "recommend phone with exceptional camera for capturing stunning photos",
]
```

	price	sim	processor	cam	ram	storage	battery	os	display
0	4	4	3	4	3	3	3	4	3
1	4	4	3	4	3	3	4	4	3
2	4	4	3	4	3	3	3	4	3
3	1	3	4	4	4	5	4	3	4
4	4	4	4	4	3	3	4	4	3
5	3	3	4	4	4	3	4	3	4
6	3	3	4	4	4	3	4	3	4
7	3	3	4	4	4	3	4	4	4
8	1	3	4	4	4	5	4	3	4
9	3	3	4	4	4	4	4	3	4
10	4	4	4	4	4	3	4	4	4
11	3	3	4	4	4	5	4	3	4
12	4	4	3	4	3	3	3	4	3
13	4	4	3	4	3	3	3	4	3
14	4	4	3	4	3	3	3	4	3
15	4	3	4	4	4	3	4	3	4
16	5	4	3	4	3	3	3	4	3
17	4	4	3	5	3	3	3	4	3
18	4	4	3	4	3	3	3	4	3
19	4	4	3	4	3	3	3	4	3

Feature vectors from BERT Classifier