

Our Team

AMISHA LALWANI
(202251013)

TUSHAR SHUKLA
(202351149)

DEVRAJ SONI
(202451050)


LAKSHAY YADAV
(20225106)

TRACK – ML

Problem statement

To Aid small businesses

ML Fashion Recommendation System



**Similar Item
Generation**

**Recommendation
Generation**

Similar Item Generation

STEP 1: DATASET- polyvore outfit dataset

STEP 2: Relevant data extraction

We filter our image dataset into 4 categories tops,

```
valid_categories = {  
    "tops", "blouses", "shirts", "t-shirts",  
    "pants", "skirts", "shorts", "jeans", "trousers", "bottoms",  
    "shoes", "boots", "heels", "sneakers", "sandals",  
    "bags", "handbags", "purses", "clutches"  
}
```

STEP 3: Feature Extraction

We use pretrained ResNet50 neural network

Input : .jpg image files

Output: 2048x1 vector

STEP 4: Similarity Mapping

cosine similarity.

measures angles between vectors not magnitud

+1->A&B in same direction

0-orthogonal completely unrelated

-1->Pointing in opposite directions

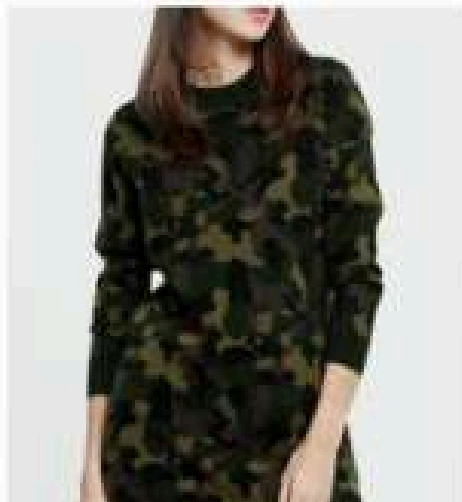
Input : Query image ID

Output : K most similar items

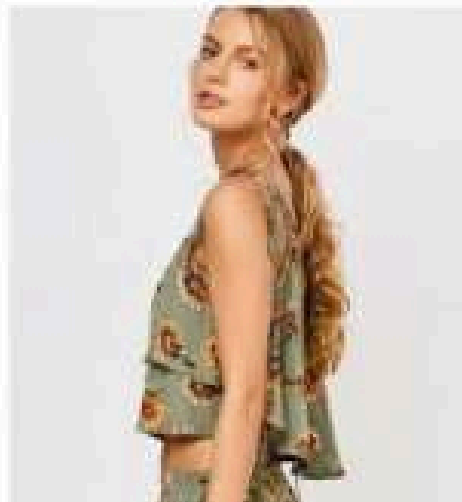
Query Image



Score: 0.76



Score: 0.74



Score: 0.72



Score: 0.72



Score: 0.72



Recommendation Generation

STEP 1: DATASET- polyvore outfit dataset - Nondisjoint - test.json train.json

STEP 2: Feature Extraction

We use pretrained ResNet50 neural network

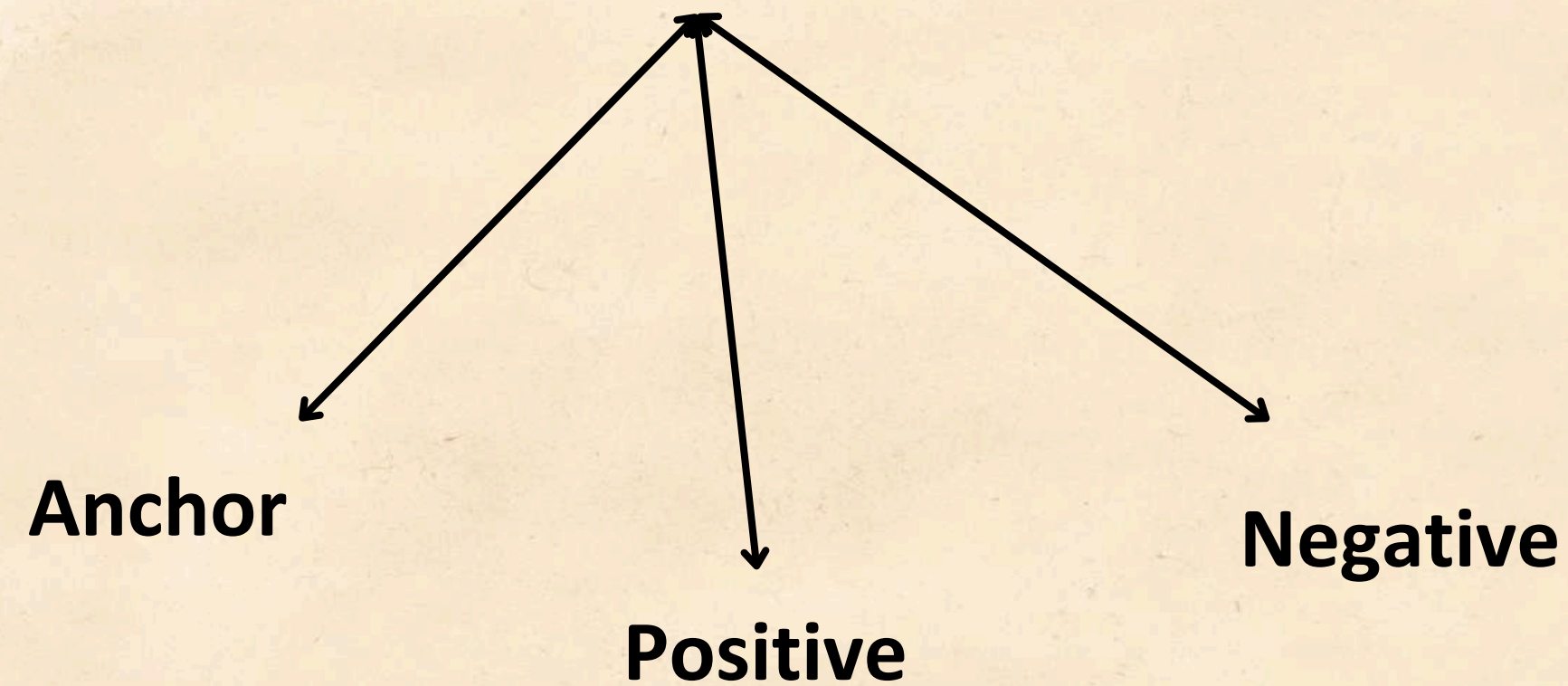
Input : .jpg image files

Output: 2048x1 vector

STEP 3: Data extraction from train.json

```
{ 'items': [{ 'item_id': '154249722', 'index': 1 }, { 'item_id': '188425631', 'index': 2 }, { 'item_id': '183214727', 'index': 3 } ], 'set_id': '210750761' }  
[ { 'item_id': '154249722', 'index': 1 }, { 'item_id': '188425631', 'index': 2 }, { 'item_id': '183214727', 'index': 3 } ]  
{ 'item_id': '154249722', 'index': 1 }
```


STEP 4: Generate triplets



```
('154249722', '188425631', '170296108')
('154249722', '183214727', '172046976')
('188425631', '183214727', '59939642')
('201813350', '195213892', '70616529')
('201813350', '169020872', '154223887')
```


STEP 5: Compatibility MLP

```
class CompatibilityMLP(nn.Module):
    def __init__(self, embed_dim=2048, hidden_dim=512):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(embed_dim * 2, hidden_dim), # 2048 * 2 = 4096
            nn.ReLU(),
            nn.Linear(hidden_dim, 1),
            nn.Sigmoid()
        )

    def forward(self, x1, x2):
        x = torch.cat([x1, x2], dim=1)
        return self.fc(x)
```

STEP 6: Generate Compatibility datasets

```
class CompatibilityDataset(Dataset):
    def __init__(self, triplets, features):
        self.pairs = []
        self.labels = []
        for a, p, n in triplets:
            self.pairs.append((a, p)) # compatible
            self.labels.append(1)
            self.pairs.append((a, n)) # incompatible
            self.labels.append(0)
        self.features = features

    def __len__(self):
        return len(self.pairs)

    def __getitem__(self, idx):
        id1, id2 = self.pairs[idx]
        x1 = self.features[id1].to("cuda")
        x2 = self.features[id2].to("cuda")
        label = torch.tensor(self.labels[idx], dtype=torch.float32).to("cuda")
        return x1, x2, label
```


STEP 7: Train The model

```
compat_model = CompatibilityMLP().to("cuda")
optimizer = torch.optim.Adam(compat_model.parameters(), lr=1e-4)
loss_fn = nn.BCELoss()

comp_dataset = CompatibilityDataset(triplets, image_features)
comp_loader = DataLoader(comp_dataset, batch_size=1024, shuffle=True)

for epoch in range(10):
    total_loss = 0
    for x1, x2, label in tqdm(comp_loader, desc=f"Compat Epoch {epoch+1}"):
        out = compat_model(x1, x2).squeeze()
        loss = loss_fn(out, label)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        total_loss += loss.item()

    print(f"Epoch {epoch+1}: Loss = {total_loss / len(comp_loader):.4f}")
```

```
Compat Epoch 1: 100%|██████████| 1342/1342 [00:47<00:00, 28.45it/s]
Epoch 1: Loss = 0.5897
Compat Epoch 2: 100%|██████████| 1342/1342 [00:47<00:00, 28.10it/s]
Epoch 2: Loss = 0.5530
Compat Epoch 3: 100%|██████████| 1342/1342 [00:48<00:00, 27.73it/s]
Epoch 3: Loss = 0.5397
Compat Epoch 4: 100%|██████████| 1342/1342 [00:48<00:00, 27.89it/s]
Epoch 4: Loss = 0.5315
Compat Epoch 5: 100%|██████████| 1342/1342 [00:47<00:00, 28.49it/s]
Epoch 5: Loss = 0.5250
Compat Epoch 6: 100%|██████████| 1342/1342 [00:48<00:00, 27.87it/s]
Epoch 6: Loss = 0.5195
Compat Epoch 7: 100%|██████████| 1342/1342 [00:47<00:00, 28.07it/s]
Epoch 7: Loss = 0.5143
Compat Epoch 8: 100%|██████████| 1342/1342 [00:47<00:00, 28.16it/s]
Epoch 8: Loss = 0.5098
Compat Epoch 9: 100%|██████████| 1342/1342 [00:47<00:00, 27.98it/s]
Epoch 9: Loss = 0.5054
Compat Epoch 10: 100%|██████████| 1342/1342 [00:48<00:00, 27.89it/s]
Epoch 10: Loss = 0.5013
```


STEP 8: Recomend items based on query

Query Items: ['165668728', '171568733']

- Suggested: 208760467, Score: 0.9579
- Suggested: 170046349, Score: 0.9567
- Suggested: 202535931, Score: 0.9510
- Suggested: 206476589, Score: 0.9497
- Suggested: 213658084, Score: 0.9491

165668728



Query Items

171568733



Top Suggestions

208760467



170046349



202535931



206476589



213658084



Query Items: ['214359927', '209892020']

- Suggested: 212788254, Score: 0.9351
- Suggested: 198303067, Score: 0.9262
- Suggested: 213658084, Score: 0.9107
- Suggested: 198236017, Score: 0.9068
- Suggested: 145536679, Score: 0.9046

214359927



Query Items

209892020



Top Suggestions

212788254



198303067



213658084



198236017



145536679



