Complete the Look Recommendation From Street Fashion Images

Submitted in partial fulfillment of
the requirements of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by
Bhaskar Bagchi
Roll No: 11CS10058

Under the guidance of
Prof. Pabitra Mitra

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India – 721 302

Spring–2015

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Certificate

This is to certify that this is a bonafide record of the project presented by Bhaskar Bagchi(*11CS10058*) during Spring 2015 in partial fulfilment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering.

Prof. Pabitra Mitra
(Project Guide)

Date:

Abstract

Recommendation Systems have now become ubiquitous to modern eCommerce. Most of the current research on recommendation systems focus on improving the speed and relevance of the recommendation. The aim of this project is to improve the relevance of recommended items with other items in the shopping cart. More specifically the work is on the problem which called *Complete the Look*, where a complete set of garments and accessories which complement the items in the users' shopping cart and can be used together as a complete look is recommended. The recommendations will be in accordance to the current fashion trends. Here recommendation is reduced to a *graph problem*. First an attribute vocabulary is built using annotated data, which is then used to train the co-occurrence graph of various clothes that can be worn together. The nearest neighbors of the items in the user's cart are recommended. Experiments are performed on a dataset of more than 500 user annotated street fashion images and the corresponding results are reported at the end of this paper.

Acknowledgments

This project would not have been possible without the guidance and patience of my guide Prof. Pabitra Mitra. Sir, you are an innovator of great thoughts and higher learning. You always ensured that I went above and beyond my potentials and showed me the value of thinking outside the bounds of all problems. The benefits of your advising went well beyond simply modelling and copying. I will forever benefit professionally and personally from your efforts during this project.

Thanks to my mother for her constant love and support and thanks to dad for being a wonderful role model. I would also take this opportunity to thank all the friends at IIT Kharagpur for being so wonderful. It was always a great relief to get out for lunch or a cup of coffee. I will never forget the memories and friendship developed here. I would specially thank *Mandakinee Singh, Smriti Agrawal, Samidha Paroha and Shiwangi Shah* for their immense help during the evalution phase.

Most importantly I thank god for his faithful provision throughout my life.

Bhaskar Bagchi

May, 2015
Indian Institute of Technology, Kharagpur

Contents

List of Figures

Chapter 1

Problem Definition

Given one or more in the shopping cart the problem statement is to suggest items complementary to it which may contain garments or accessories which makes a complete set as per current fashion.

Physically speaking if someone adds a top in her shopping cart, we try to suggest her matching/complementary jeans, denim, skirt, bag, sunglasses, shoes, etc. which go well with the product chosen and make a complete set of items that can be used together.

Figure 1.1: Problem Definition: Complete the look recommendation

Chapter 2

Introduction

Most of the malls and shops have scarcity of resources. They have limited number of shelves and space, thus can show the customers only a limited fraction of choices available. On the contrary to it online stores and e–shopping provides more options to the user to choose from. This abundance of choices and need of custom–tailoring the options to the needs of specific users make recommendation fairly important for e–commerce. It won't be wrong to say that it is the backbone of online e-commerce. We have a number of recommenders which use various techniques like content–based or collaborative filtering which recommend similar items based on user's previous choices and requirements. So we buy a Denim Jeans, we will be recommended with various Jeans ranging in various prices, companies, designs or other aspects. But we have rarely seen an online garment seller who recommends matching tops, t–shirts or accessories when we buy Denim jeans. Many times while shopping for garments online users find difficulty in finding clothes that match whatever is already in the cart. In such situations a recommendation can influence and increase the sales, and also enhances customer loyalty and engagement. The attractiveness of garments in this situation depend on other items bought by the user. Therefore it is beneficial to consider a whole set of recommendation instead of treat items individually. In this project we target this problem of online garment sellers, where given an item in the shopping cart we intend to suggest items complementary to it which may contain garments or accessories which make a complete set as per current fashion. We call it *Complete the Look problem.* This falls under the class of recommendation problem called the *Bundle Recommendation* problem.

2.1 Bundle Recommendation

Bundle is referred to as a set of items that the customer considers or buys together. Since we are specifically working with garments, a bundle in our case will be a set of clothes that are complementary to each other and can be worn together. In classical marketing research it is well known that there is a '$1 + 1 > 2$' effect in carefully designed product bundles[7]. This is because they generate context for the customer to buy more products. For example, if someone is buying a suit, it is irresistible to buy a tie that goes well with it. We will recommend a bundle of matching clothes which will lead to an increased sales probability.

2.2 Dataset and Ground-truth

Most progress in computer vision has been partly possible because of annotated public datasets. Obtaining large datasets with reasonably clean dataset is a daunting challenge. Specially annotated images of our requirements which express social, cultural and commercial importance are rarely available. We would like to collect fashion images which contain a full view of street fashion where model has several fashion accessories. The first large scale public scale dataset for street fashion was presented in [6]. It had 158235 images out of which only 685 were annotated. Another similar dataset was introduced in [**?**]. They called it the Fashion–136K. This consists of fashion images along with other metadata such as annotations of accessories, brands and demographic of the fashionistas who posted the images. We can use the fashion websites as source of annotated data for fashion recommendation. There are various websites where users and fashionstas upload their images in various clothes which in accordance to the current fashion trends, and this is a potential source of large fashion recommendation datasets. The data is easily available and is user annotated.

Our problem in hand i.e. recommending a bundle of complementary items has has a couple of computational challenges:

1. Representation of a fashion items as visual features is an open ended problem. Not much work has been done on it. Mostly color templates or HSV histograms are used.

2. Developing the notion of relative similarity is another challenge. It is subjective and we don't have concrete numerical measure for it. It depends on various factors like location, season and sometimes the occasion on which the clothing is to be used. This makes learning inherently complex.

Apart from them there are a number of practical challenges that are to be addressed like quality of input and training data, scalability issues in terms of memory and speed requirements. The main purpose of the project can be viewed in Figure 1.1.

We formulate the recommendation problem as follows: given an image i containing 'k' fashion items, referred to as 'part features'(eg. red top, black boots, etc.), we describe the image P_i as $P_i^T := [p_{i1}, p_{i2}, ..., p_{ik}]$ where each p_{ij} are textual part features which are two-tuples of part category and description. We learn a model from our dataset of fashion images, say $\mathbf{P}$, where $\mathbf{P} := [P_1, P_2, ...P_n]^T$. The task of our recommendation system is, given one or more apparel, and corresponding part features p's as input query, recommend garments which can be worn with it/them as a set.

Chapter 3

Literature Review

Research in visual fashion focuses on fashion parsing and retrieval of similar fashion images[6]. These methods either use mixture of parts based pose estimators or use poselet part detector after applying the deformable parts based detector for person detector. Vast amount of existing literature also deal with learning aesthetically pleasing colors which can be helpful in the problem in our hand, but there are a very few papers which directly deal with *complete the look*. Liu *et al.*[5] recommend clothes to wear for particular occasions. Here the user inputs the occasion like *X-mas, romantic date* or *family party* and clothes are suggested accordingly. They propose an SVM based formulation to handle both 'wear properly' and 'wear aesthetically' properties in the model. Iwata *el at.*[2] study a similar problem of matching clothes recommendation and propose a topic model based approach. Jagadeesh *et al.*[3] proposed using data driven approaches including Complementary Nearest Neighbour Consensus, Gaussian Mixture Models, Markov Chain-LDA and Texture Agnostic Retrieval. We propose a graph based approach to study the same problem and use textual tags as features instead of using visual features.

Chapter 4

Background

4.1 Bipartite Network Projection

Bipartite networks are a particular class of complex networks, whose nodes are divided into two sets X and Y and only edges between two nodes in different sets are allowed. For the convenience of directly showing the relation structure among a particular set of nodes, bipartite networks are usually compressed by one-mode projection. This means ensuring network contains nodes of only one of the two sets X or Y, and two X(or alternatively Y) nodes are connected if and only if they have at least one common neighbour Y(or alternatively X) node. Since bipartite networks with largely different structures can have one-mode representation, a lucid illustration of original network requires the use of some weighing method. Various weighting methods have been proposed and should be chosen with case.

In *simple weighing technique* the edges are weighted directly by the number of times the common association is repeated. This method works in wide range of settings where the impact of each relation in the bipartite network has equal impact.

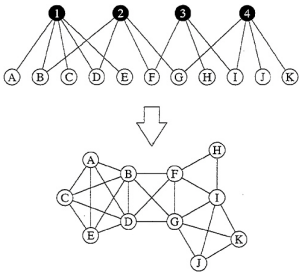

Figure 4.1: Bipartite network and its projection

4.2 Simrank

SimRank [**?**] is an algorithm for analysing the logical graphs derived from such datasets to compare similarity scores between nodes(objects) based on *structural context* in which they appear. The intuition behind this algorithm is that similar

objects are related to similar objects. Consider a small directed graph as shown in the Figure 4.2 below. The vertices represent websites and edges represent hyperlink i.e. one website is referred by the other. The basic assumption/intuition is that two objects are similar if they are referred to by some common object. So here we can say that websites *v3* and *v5* are similar i.e. contain similar content because both are referred by website *v2*.

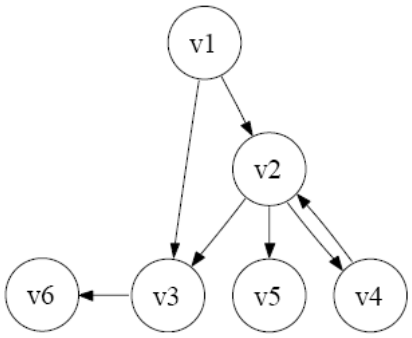

Figure 4.2: Directed graph representing websites and hyperlinks

Mathematically SimRank score is defined recursively. Let us denote similarity between objects a and b by s(a, b) $\in$ [0, 1].

$$S(a,b) = \begin{cases} 1 & \text{if } a = b; \\ \frac{C}{|I(a)||I(b)|} \sum_{i=0}^{|I(a)|} \sum_{j=0}^{|I(b)|} S\left(I_i\left(a\right), I_j\left(b\right)\right) & otherwise. \end{cases} \tag{4.1}$$

where, C is a constant between 0 and 1 which controls the amount of score propagation in recursive call, *I(a)* is the set of incoming edges to a and *I(b)* is the set of incoming edges to b.

4.3 Rank Aggregation

Rank aggregation is an old problem of taking into account the preferences of multiple competing agents fairly. Say we have n candidates and k voters giving complete or partial preference lists, we want a single list of ordering of candidates expressing all voters' preferences. In social choice theory, Arrow's impossibility theorem or Arrow's paradox states that when voters have three or more distinct alternatives, no rank order voting system can convert the ranked preferences to a community wide(complete and transitive) ranking which also meet a pre–satisfied set of criteria. These pre–satisfied set of criteria are unrestricted domain, non-dictatorship, *Patro* efficiency(if everyone prefers A to B then in the final result A should be preferred over B) and independence of irrelevant alternatives(given two inputs in which A and B are ranked identically by everyone the two outputs should order A and B the same). Much work on this topic is done in relation to the website ranking by various search engines and meta search engines. As we have posed our recommendation problem to a graph problem, if we have more than one item in the shopping list we will have more than one ranked list which are to be combined to a single list for recommendation. We use the concept of rank aggregation for that.

Mathematically,given a universe U, an ordered lists L_1, L_2, etc. three situations arise

1. If the individual lists contain all the elements in U then they are called complete lists. They are a total ordering of U.

2. There are situations where full lists are not convenient, and even not possible. In such cases the lists contain an ordered list of a subset of elements of U. $|L_i| < |U|$. Such lists are called partial lists.

3. A special case of the partial list problem is the top–K list. In this ranking we take the top K elements from each ordered list, and so we know that all the elements that are not present in the list are ranked below those which are present in the list. This are called the top–K lists.

Our problem falls in the third category, the top–K list problem.

The final aggregated list is a permutation of the elements of the universe. Many distance metrics have been proposed to find the distance of a ranked list from the final solution. One of the most popular metric is the Kendall's tua distance. The Kendall's distance between two ordered lists is the number of pairs (i, j) that the lists disagree on. Thus to find an aggregated list, we find the permutations of U and choose a permutation S_i such that Kendell's distances of the permutations from all the ranked lists is the minimum. But this algorithm is NP hard to compute for 4 or more elements. Thus the only remaining feasible option is the use of heuristics. Many heuristic algorithms and approximation algorithms have been proposed.

Chapter 5

Overview of Approach

We propose a graph based approach for the recommendation task. In this approach a weighted graph of features is used as the model M and given any query q, which contains some part features p we try to find those part features which are close to it. For this we first construct a vocabulary of part features we have including garments and accessories. Next we take a large database of user annotated fashion images and create a co-occurrence graph of various apparels and accessories that are used together in various images by fashionstas. Now given any query item, we pick the k–nearest neighbors of the node corresponding to the input item. Here *simrank*[4] is used as the similarity measure between nodes to calculate nearest neighbors. In case of more than one items in the input query we generate individual k–nearest neighbors lists and then merge the lists using the *Borda's* Rank aggregation algorithm[1]. The final aggregated list is recommend to the user. The steps are illustrated in Figure 5.1.

5.1 Scrapping Fashion Websites

As mentioned earlier there are only a handful of public fashion databases available, and finding an annotated fashion dataset is further difficult. But the large number of street fashion images present on the internet and the ease of availability shows immense potential to generate useful user annotated dataset, specially for experiments dealing with contemporary fashion trends. There are various fashion websites where fashionstas upload their images with currently trending apparels, and the images are user annotated i.e. each image is tagged with the type of garments. This makes exactly the kind of dataset we want to work with. We scraped data from a fashion website `www.chictopia.com` which is a large style community where bloggers share their style posts and it also has online clothing boutiques. We scraped nearly *500* images of female fashionstas with the tags corresponding to each image. These images cover an appreciable range of street fashion from corporate dressing sense to the most casual of the dresses.

Once the data was scraped we created a vocabulary of part features we had from the database. We used category-description pair as part feature in our work. We had to normalize the tags associated with each image manually because each user used different types of tags. As we are working with *textual* features instead of

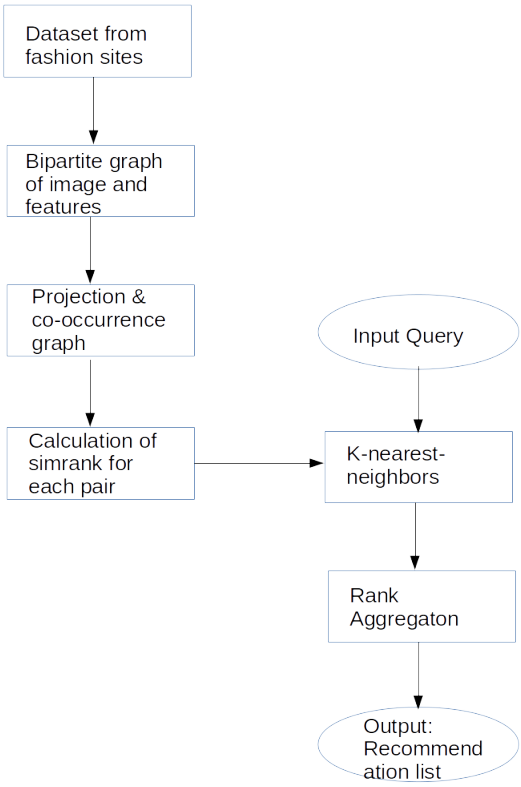

Figure 5.1: Flow chart of proposed algorithm.

visual features, this was one necessary step. For each image P_i, each part feature p_j has two tags, first is the type of clothing, like shirt, top, etc. and second is its one word description, mostly color. Once this was done we ended up with a codebook of total of *48* unique categories including garments like tops, jeans, etc. and accessories like watches, bracelets, etc. and *632* unique items i.e. category-description pair. For each image, each part is described by a 2–tuple {*Cloth – type:Cloth – description/color*}. For example {*Shirt:Blue*}, {*Skirt:Black*}, {*Watch:Gold*}, etc. are some valid features in our case.

5.2 Simplified Mathematical Formulation

5.2.1 Projection and Co-occurrence graph

In the dataset, each image P has some part features which is defined using two–tupes as described above. We now have a bipartite graph with dataset images P's as the first partite sets and part features p_i's as the second partite set. There exists an edge between ever part feature and the image in which it occurred. The bipartite graph is then projected to the set of part features. While projecting the graph, simple weighting technique was used, where the weight of an edge between two part features p_i and p_j is the number of common images where p_i and p_j occurred

together.

The projected graph so obtained is a weighted co–occurrence graph of the part features. Construction of this graph gives us the relation between different garments and accessories which can be used together and are complementary to each other. The edge weights signify the number of times the part features occurred together i.e. how compatible the garments are when they are worn together. Since the graph is constructed from street fashion images we can safely assume that the co–occurrences are as per the current fashion trends. These are also assumed to be aesthetically sound because they are actually used by models and fashionistas. This step helps us learn a correlation and inter-dependence between various part features from the dataset.

5.2.2 Similarity Measure and Nearest Neighbour Consensus

The co-occurrence graph falls in a domain where nodes represents the objects and edges represents the relations between them. Now we need a similirity measure between the nodes so that we can decide which part feature, or garment in a broader sense, is complementary to which other part features. SimRank[4] is an algorithm for analysing the logical graphs and comparing similarity scores between nodes(objects) based on *structural context* in which they appear. We first convert the co–occurrence graph into a directed graph where each edge between part features p_a and p_b in the original graph is replaced by two directed edges $p_a \rightarrow p_b$ and $p_b \rightarrow p_a$ both with weights equal to the weight of original edge. Now we compute *simrank* between each pair of nodes. For fast computation, we leverage the fact that simrank is defined recursively and use *memoization* technique to reduce computation time.

Now, given any part feature p (corresponding to the garment in user's shopping cart) as query we locate the node corresponding to that part feature in the co–occurrence graph. This can be easily done by storing the relation between textual part feature and corresponding node in co–occurrence graph in a hash table. After the node is located, we find out other nodes which are close to it, i.e. nodes which have highest simrank value with this node. Hence we find out the at most top k such nodes, the k–nearest–neighbours. The rationale behind this step is that since the graph had edges between part features that were used together by fashionistas and as the simrank values decrease with increase in node distances, the k–nearest–neighbors will be those part features which were frequently used with the selected item and are contemporary to it.

So, we get a list of k part features $p_1, p_2, ...p_k$ which are structurally close to the input feature and thus they can be recommended for the given query part feature.

5.2.3 Aggregating Ranked Item Recommendations

If there is only one item in the shopping cart, then the steps mentioned above will suffice. But if we have multiple items in the shopping cart, we need to perform one more task before the final recommendation. Say we have j part features $p_1, p_2, ...p_j$

as input query, we find out individual k–nearest–neighbors for each part feature. Now we have j ranked lists, each with k members, which are recommendation related to each input feature. Our final goal is to provide a single ranked list of recommendations. This is a problem of rank aggregation. There are various heuristic based rank aggregation algorithms in literature. We use *Broda's* aggregation algorithm[1] for our algorithm.

Broda's method is a proportional method in which it assigns a score corresponding to position in which a part feature appears within each ranked list. In our case, for each list i, ${p_a}^i$ is assigned a weight ${B_{p_a}}^i = k$ * fraction of part features in the list appearing below p_a. This is because in some cases the list does not have k elements but the maximum score that the top element should get is k. The *Broda* score of each element B_{p_a} is the the sum of *Broda* scores for that part feature in all the lists. Then the sorted list in non-increasing order is the final aggregated ranked list. The primary advantage of using this method in our case is that it is computationally very fast and can be implemented in linear time. Thus we get the aggregated ranked list in case of multiple input part features.

We can recomment the top k elements from this ranked list to the user.

5.2.4 Overall Algorithm

The overall algorithm for recommendation can be summarized as presented in Algorithm 1.

5.3 Shortcomings in this approach

There are a few shortcomings with the approach taken. The first and most important concern is that with the increase in number of nodes or features the size of the graph becomes large and difficult to analyze. Calculation of *SimRank* for the whole graph also becomes computationally expensive. Another important concern is that of out of vocabulary queries. Also the textual tags might not always be enough to represent a fashion accessory or garment. For the implementation purpose an optimized code is required to store the feature graph. *Simrank* should also be pre-calculated and stored for each couple of nodes, because calculation of *SimRank* is recursive and calculating it on each query will be very slow and expensive as well.Also the feature graph learned should be updated frequently with the change in trends in contemporary fashion. Creating a dataset for the learning task by manually tagging each image is a tiresome work. Another issue with this approach is network partitioning. If a node or a bunch of nodes is secluded from the graph, it is not possible to generate the recommendations for those part features.

Algorithm 1: COMPLETE THE LOOK

Input: A list of part features P containing part features p_1, p_2,...
Output: A list of recommended part features

```
Algorithm main()
    listofrecommendation                    // list of lists of features
    while p_i in P do
        list = kNearestNeighbors(p_i)
        listofrecommendation.add(list)
    end
    finallist = broda(listofrecommendation)
    return finallist
Procedure kNearestNeighbors(node)
    setofneighbors = BFS(node)
    for neighbor ∈ setofneighbors do
        calculateSimrank(node, neighbor)
    end
    return k nodes with highest simrank scores
Procedure broda(listoflists)
    max = listoflists.get(0).size()
    listFinal = nodes from each list in listoflists
    for list ∈ listoflists do
        for i < list.size() do
            assign broda score for node(list(i))= max - i
            update listFinal
        end
    end
    return 10 elements of listFinal with highest broda scores
```

Chapter 6

An Example

In this chapter we will discuss with an example the actual working of the recommender with a realistic example. We will take an input and begin the analysis step–by–step. Let the user has three items in the shopping cart, which are as follows:

- Light Brown Skirt
- Dark Green Shirt
- Brown Scarf

As a first step, we take the inputs as part feature in 2-tuple format
$p_1 = Skirt \rightarrow Light\ Brown$
$p_2 = Shirt \rightarrow Dark\ Green$
$p_3 = Scarf \rightarrow Brown$

6.1 Similarity

Now for each of the part feature, we find the neighbors and corresponding score of similarity measure. For this we do a BFS to the node of input part feature and for each neighbor calculate the simrank score. The Neighbors with corresponding simrank scores are presented in *Table*6.1, *Table*6.2 and *Table*6.3

6.2 Nearest Neighbor

Since the top 5 neighbors of each feature are returned, the following part features are obtained as recommendation for individual items. The results for our input is given in *Table* 6.4

6.3 Rank Aggregation

Once the ordered rank list for each feature is received, we follow the Broda's scoring technique and score each part feature in each list, the following is the broda score for

Table 6.1: Neighbors for Light Brown Skirt

Node No.	Feature	Simrank Score
34	*Top* → *Off White*	0.0024232180348514195
395	*Top* → *Peach*	0.0036795985834136963
495	*Bag* → *Handmade*	0.007307627187923889
18	*Blouse* → *White*	0.001810453655830971
494	*Glasses* → *Gray*	0.0073076271855647395
321	*Bag* → *Dark Gray*	0.002073684223076003
381	*Bag* → *Amethyst*	0.004012744082021973
24	*Boots* → *White*	7.157687783842247E-4
298	*Tights* → *Silver*	0.0016980449345485899
367	*Boots* → *Maroon*	0.003651443389923733
496	*Shoes* → *Flats*	0.0072933558373829074

Table 6.2: Neighbors for Dark Green

Node No.	Feature	Simrank Score
509	*Purse* → *Brown*	0.005153511903097642
510	*T-shirt* → *Eggshell*	0.005153511958756549
340	*Necklace* →	7.138473587161387E-4
511	*Hat* → *Neutral*	0.0022735552806618085
512	*Boots* → *Heather Gray*	0.0015633349481407172
504	*Loafer* → *Brown*	4.932201992766598E-4
6	*Jacket* → *Sky Blue*	0.0013077265455576177
351	*Skirt* → *Bubble Gum*	0.0018491842574126678
119	*Shoes* → *Black*	6.850209148704326E-5
16	*Dress* → *Navy*	9.234644641127203E-4
115	*Jacket* → *Crimson*	2.6413410667371073E-4
364	*Skirt* → *Maroon*	0.001843020532138039
366	*Earrings* → *Green*	0.001133841788133361
57	*Skirt* → *Black*	0.0014119021582344213
28	*Boots* → *Black*	0.0010103833088949928
61	*Hat* → *Black*	9.687164729080336E-4

Table 6.3: Neighbors for Brown Scarf

Node No.	Feature	Simrank Score
17	*Bag* → *White*	0.005678252182894807
84	*Shirt* → *Black*	0.0019481985857470335
50	*Top* → *Black*	0.0012562560345584332
57	*Skirt* → *Black*	0.0018072561827211103
167	*Hat* → *Fedora*	0.0030579584078584115
28	*Boots* → *Black*	0.0012206353776358363
61	*Hat* → *Black*	0.0013121156362252237
165	*Coat* → *Olive Green*	0.010810194265580369
164	*Bracelet* → *Silver*	4.368215287610927E-4

Table 6.4: Individual recommendation for each input

Light Brown Skirt	Dark Green Shirt	Brown Scarf
495	510	165
494	509	17
496	511	167
381	351	84
395	364	57

Table 6.5: Broda Score for each node

Node No.	Broda Score
509	14
510	15
511	13
381	12
351	12
167	13
496	13
165	15
395	11
17	14
84	12
495	15
494	14
364	11
57	11

each part–feature obtained. The Broda's scores for nodes in our case is presented in *Table* 6.5

The final recommendation is the list of 10 features with the highest Broda's score. Thus the final recommendation is as follows:
495: Bag → Handmade
165: Coat → Olive Green
510: T-shirt → Eggshell
494: Glasses → Gray
17: Bag → White
509: Purse → Brown
119: Shoes → Flats
167: Hat → Fedora
511: Hat → Neutral
84: Shirt → Black

Chapter 7

Experimental Results

7.1 Evaluation Methodology

Evaluating recommendation is subjective because user ratings vary according to each user. There are some standard approach where recommendation is evaluated using *precision, recall* and *f1–score*. We took 20 images as test set from our dataset. Since each image is user tagged, we have labelled ground truth for computing the required metrics. For each image we took, we used all its part features individually as one feature input. We also used various permutations of 2 part features and 3 part features as input to the recommender and compared the recommended part features with the ground truth i.e. the original part features in the image. Thus we calculated *precision, recall* and *f1* values for 158 sets of recommendations.

$$precision = \frac{no\ of\ matched\ recommendations}{no\ of\ recommendations}$$
$$recall = \frac{no\ of\ matched\ recommendation}{no\ of\ items\ in\ actual\ image}$$

Since the rating of recommendation varies from one user to other, only evaluating the precision and recall of annotated dataset is not enough. So, we also used manual evaluation where we asked users to choose some elements as shopping list and then asked them to rate the recommendations in a scale of 1 to 10 as per their satisfaction with the recommendation.

7.2 Results

Out of the 158 recommendation sets that we tested, 53 were 1 part feature input, 54 were 2 part feature input and 51 as 3 part feature input. For each generated recommendations we calculated the precision and recall. The results are given in the tables, Table 7.1, Table 7.2, Table 7.3, respectively.

The precision v/s recall graphs for the above experiments are are follows:

On the user evaluation we got 60 sets of recommendations evaluated by various users giving the recommender as a black box and the results were quite satisfactory. The results are summarized in Table 7.4.

Table 7.1: Precision

No. of inputs	Max Precision	Avg Precision
1	1	0.31
2	0.75	0.31
3	0.6	0.28

Table 7.2: Recall

No. of inputs	Max Recall	Avg Recall
1	0.8	0.23
2	1	0.44
3	1	0.48

Table 7.3: f1 score

No. of inputs	Max f1	Min f1
1	0.89	0.13
2	0.71	0.1
3	0.67	0.1

Table 7.4: User rating for recommendation

Rate(out of 10)	Frequency	Cumulative Freq.
10	1	1
9	2	3
8	9	12
7	9	21
6	5	26
5	11	37
4	11	48
3	6	54
2	4	58
1	2	60

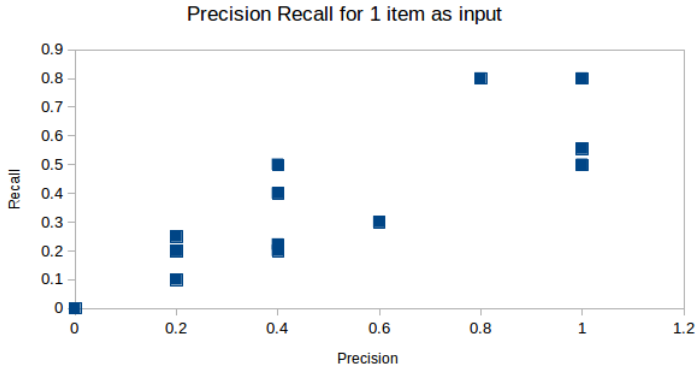

Figure 7.1: Precision-Recall for 1 item input

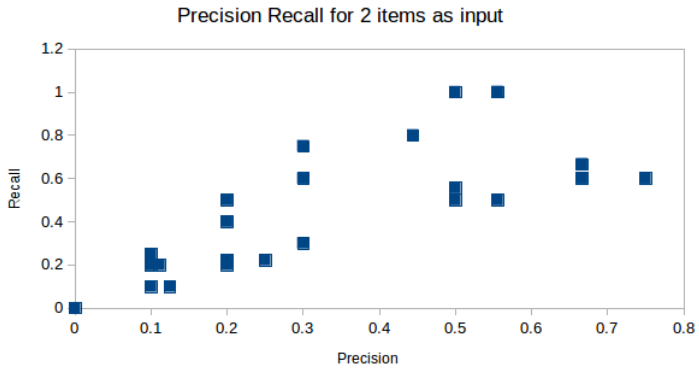

Figure 7.2: Precision-Recall for 2 item input

7.3 Analysis of Results

From the above results, it can be clearly observed that the recommendations generated by our algorithm produces good results when rated by real users. 26 recommendations i.e. 43% of the recommendations got a rating of 6 and above, whereas 37 recommendations i.e. nearly 62% of the recommendations got a rating of 5 and above. The reason of poor rating of some recommendation may be the lack of dataset. The input node may have very low similarity score with other nodes, thus giving poor recommendations.

The average precision values are highest for one or two items given as input whereas it is comparable for three items as input. But at the same time the highest average recall is obtained for three items as input. The recall value for two items as input is comparable to that of three items input. Also the f1–scores for two input recommendations are comparably satisfactory for two items input. We perform experiments only 1, 2 or 3 part features as inputs because it is reported that the average bundle size at Amazon.com is 1.5 and at Walmart.com is 2.3 [7].

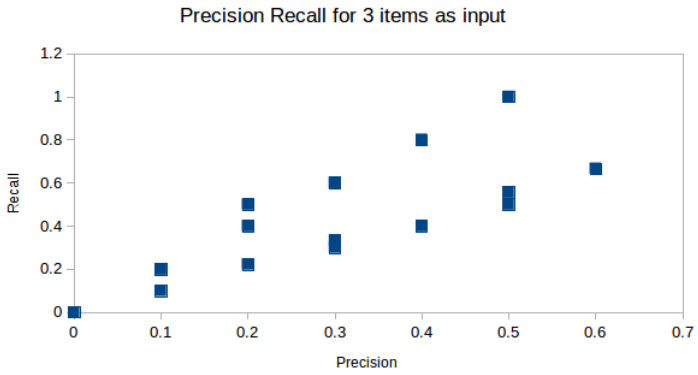

Figure 7.3: Precision-Recall for 3 item input

Chapter 8

Future Work

This is a barebone implementation and there are many advancements still to be done in the proposed technique. Testing and evaluation has revealed that the granularity of the features should be increased for better results. The size of database should also be large so as to include many types of appearels. Out of vocabulary errors are yet to be dealt with. There are many things that can be done as an add on to improve both performance and quality of results, some of them are listed as follows.

- Features for representation of parts are to be improved by incorporating visual features. Inclusion of visual features will also include the analysis of features like color, texture, etc. which is expected to improve the quality of evaluation.
- A feedback system can be added to the system as to increase edge weights to the features which are shopped together by users. This will be a self learning system and incorporate the changes in trending fashion all by itself.

Chapter 9

Conclusion

In this work we attempted to solve the Complete the Look Using the Street Fashion Images problem by reducing it to a graph problem and using structural similarity between objects instead of creating numeric feature vectors. It is very rarely observer that such problems are approached using textual features. The results obtained were satisfactory. Due to lack of evaluation methodology, only manual and subjective analysis of the results could be performed. Moreover our learning was that using textual features and structural similarity between objects can provide intuitive domain insights and perform satisfactorily for context based similarity and recommendation. The results obtained seemed to be best for 2 input queries in terms of precision and recall.

References

[1] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[2] T. Iwata, S. Watanabe, and H. Sawada. Fashion coordinates recommender system using photographs from fashion magazines. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2262–2267. AAAI Press, 2011.

[3] V. Jagadeesh, R. Piramuthu, A. Bhardwaj, W. Di, and N. Sundaresan. Large scale visual recommendations from street fashion images. *CoRR*, abs/1401.1778, 2014.

[4] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[5] S. Liu, J. Feng, Z. Song, T. Zhang, H. Lu, C. Xu, and S. Yan. Hi, magic closet, tell me what to wear! In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 619–628, New York, NY, USA, 2012. ACM.

[6] K. Yamaguchi, M. Kiapour, L. Ortiz, and T. Berg. Parsing clothing in fashion photographs. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3570–3577, June 2012.

[7] T. Zhu, P. Harrington, J. Li, and L. Tang. Bundle recommendation in ecommerce. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 657–666, New York, NY, USA, 2014. ACM.

Appendix A: Evaluation Results

Table 9.1: Evaluation of test set.

Begin of Table						
Set#	# input	# Reco	# Match	# Original	Precision	Recall
0	1	5	0	4	0	0
0	1	5	1	4	0.2	0.25
0	1	5	2	4	0.4	0.5
0	1	5	0	4	0	0
0	2	10	1	4	0.1	0.25
0	2	10	3	4	0.3	0.75
0	2	10	2	4	0.2	0.5
0	2	5	0	4	0	0
0	3	10	2	4	0.2	0.5
0	3	10	2	4	0.2	0.5
0	3	10	2	4	0.2	0.5
0	3	10	2	4	0.2	0.5
2	1	5	1	5	0.2	0.2
2	1	5	0	5	0	0
2	1	4	4	5	1	0.8
2	1	5	1	5	0.2	0.2
2	1	5	0	5	0	0
2	2	10	1	5	0.1	0.2
2	2	9	4	5	0.4444444444	0.8
2	2	9	5	5	0.5555555556	1
2	2	10	1	5	0.1	0.2
2	2	10	1	5	0.1	0.2
2	2	9	5	5	0.5555555556	1
2	2	9	1	5	0.1111111111	0.2
2	3	10	3	5	0.3	0.6
2	3	10	4	5	0.4	0.8
2	3	10	4	5	0.4	0.8
2	3	10	1	5	0.1	0.2
2	3	10	1	5	0.1	0.2
2	3	10	4	5	0.4	0.8
3	1	5	0	10	0	0
3	1	5	1	10	0.2	0.1

Continuation of Table 9.1						
Set#	# input	# Reco	# Match	# Original	Precision	Recall
3	1	5	1	10	0.2	0.1
3	1	5	5	10	1	0.5
3	1	5	2	10	0.4	0.2
3	1	5	1	10	0.2	0.1
3	1	5	0	10	0	0
3	1	5	5	10	1	0.5
3	1	5	3	10	0.6	0.3
3	1	5	0	10	0	0
3	2	10	1	10	0.1	0.1
3	2	8	1	10	0.125	0.1
3	2	9	5	10	0.5555555556	0.5
3	2	9	6	10	0.6666666667	0.6
3	2	10	2	10	0.2	0.2
3	2	10	1	10	0.1	0.1
3	2	10	5	10	0.5	0.5
3	2	8	6	10	0.75	0.6
3	2	10	3	10	0.3	0.3
3	3	10	1	10	0.1	0.1
3	3	10	3	10	0.3	0.3
3	3	10	3	10	0.3	0.3
3	3	10	4	10	0.4	0.4
3	3	10	1	10	0.1	0.1
3	3	10	5	10	0.5	0.5
3	3	10	5	10	0.5	0.5
3	3	10	5	10	0.5	0.5
4	1	5	2	9	0.4	0.2222222222
4	1	5	5	9	1	0.5555555556
4	1	5	0	9	0	0
4	1	5	5	9	1	0.5555555556
4	1	5	0	9	0	0
4	1	5	2	9	0.4	0.2222222222
4	1	5	2	9	0.4	0.2222222222
4	1	5	0	9	0	0
4	1	5	0	9	0	0
4	2	9	6	9	0.6666666667	0.6666666667
4	2	10	5	9	0.5	0.5555555556
4	2	10	5	9	0.5	0.5555555556
4	2	10	5	9	0.5	0.5555555556
4	2	10	2	9	0.2	0.2222222222
4	2	8	2	9	0.25	0.2222222222
4	2	10	2	9	0.2	0.2222222222
4	2	10	0	9	0	0
4	2	10	2	9	0.2	0.2222222222
4	3	10	5	9	0.5	0.5555555556

Continuation of Table 9.1						
Set#	# input	# Reco	# Match	# Original	Precision	Recall
4	3	10	6	9	0.6	0.6666666667
4	3	10	3	9	0.3	0.3333333333
4	3	10	3	9	0.3	0.3333333333
4	3	10	2	9	0.2	0.2222222222
4	3	10	2	9	0.2	0.2222222222
4	3	10	2	9	0.2	0.2222222222
4	3	10	2	9	0.2	0.2222222222
5	1	5	1	5	0.2	0.2
5	1	5	0	5	0	0
5	1	5	1	5	0.2	0.2
5	1	4	4	5	1	0.8
5	1	5	1	5	0.2	0.2
5	2	10	1	5	0.1	0.2
5	2	10	1	5	0.1	0.2
5	2	10	5	5	0.5	1
5	2	9	5	5	0.5555555556	1
5	2	9	1	5	0.1111111111	0.2
5	3	10	1	5	0.1	0.2
5	3	10	4	5	0.4	0.8
5	3	10	4	5	0.4	0.8
5	3	10	4	5	0.4	0.8
5	3	10	1	5	0.1	0.2
6	1	5	0	5	0	0
6	1	5	1	5	0.2	0.2
6	1	5	4	5	0.8	0.8
6	1	5	1	5	0.2	0.2
6	1	4	4	5	1	0.8
6	2	10	1	5	0.1	0.2
6	2	9	4	5	0.4444444444	0.8
6	2	9	4	5	0.4444444444	0.8
6	2	9	5	5	0.5555555556	1
6	2	9	4	5	0.4444444444	0.8
6	3	10	3	5	0.3	0.6
6	3	10	3	5	0.3	0.6
6	3	10	5	5	0.5	1
6	3	10	5	5	0.5	1
6	3	10	3	5	0.3	0.6
7	1	5	0	5	0	0
7	1	5	1	5	0.2	0.2
7	1	4	4	5	1	0.8
7	1	5	1	5	0.2	0.2
7	1	5	0	5	0	0
7	2	10	0	5	0	0
7	2	9	5	5	0.5555555556	1

Continuation of Table 9.1						
Set#	# input	# Reco	# Match	# Original	Precision	Recall
7	2	9	5	5	0.5555555556	1
7	2	10	1	5	0.1	0.2
7	2	10	0	5	0	0
7	3	10	4	5	0.4	0.8
7	3	10	5	5	0.5	1
7	3	10	4	5	0.4	0.8
7	3	10	1	5	0.1	0.2
7	3	10	1	5	0.1	0.2
8	1	5	0	5	0	0
8	1	5	1	5	0.2	0.2
8	1	5	0	5	0	0
8	1	5	2	5	0.4	0.4
8	1	5	1	5	0.2	0.2
8	2	10	1	5	0.1	0.2
8	2	10	1	5	0.1	0.2
8	2	10	2	5	0.2	0.4
8	2	10	3	5	0.3	0.6
8	2	10	1	5	0.1	0.2
8	3	10	0	5	0	0
8	3	10	1	5	0.1	0.2
8	3	10	2	5	0.2	0.4
8	3	10	2	5	0.2	0.4
8	3	10	1	5	0.1	0.2
9	1	5	0	5	0	0
9	1	5	2	5	0.4	0.4
9	1	5	0	5	0	0
9	1	4	4	5	1	0.8
9	1	5	0	5	0	0
9	2	10	2	5	0.2	0.4
9	2	10	2	5	0.2	0.4
9	2	9	4	5	0.4444444444	0.8
9	2	9	4	5	0.4444444444	0.8
9	2	10	0	5	0	0
9	3	10	0	5	0	0
9	3	10	5	5	0.5	1
9	3	10	4	5	0.4	0.8
9	3	10	4	5	0.4	0.8
9	3	10	1	5	0.1	0.2
End of Table						