A Survey of Content-based Recommender Systems and Proposal of Possible Extensions to Visual Recommender Systems

Prabhav Agrawal
Department of Computer Science
University of California San Diego
prabhav@eng.ucsd.edu

Abstract

Recommender systems broadly aim to model users' preferences and guide users to available options in a personalized way, for example, online retailer suggestions based on purchase and search histories. Content-based recommender systems recommend items similar to the ones the user has liked in the past. We survey the state-of-the-art in content-based recommendation systems and discuss current trends and future research.

We also propose extensions to a visual and relational recommender system proposed by McAuley et al [1]. A visual and relational recommender system aims to model human visual preferences based on characteristics of images associated with the product and can be used for recommending items to the user based on his previous product searches/purchases.

1 Introduction

The growth of online markets has a significant impact on the behavior of consumers, providing them access to a variety of products and information on them. It has made more difficult for the consumers to select the products which are most suitable to their needs. Recommender systems aim at providing solutions to this problem of information overload for consumers by guiding them in a personalized way towards available options. They have a wide variety of applications in recommending movies, book, news, and web pages to consumers.

Recommender systems has been a hot research area since mid-1990s. There has been much work done both in the industry and academia to develop better recommender systems over the past few decades. In spite of the advances in the area, further improvements are required to make recommender systems more effective and applicable to a broader range of real-life applications. The improvements include better methods to represent user and item information, advanced modeling methods to learn user profiles, and methods to incorporate information regarding the context into the recommendation process.

Two main paradigms have emerged for recommending items in the recent past. Content-based recommender systems try to recommend items similar to the ones a given user has liked in the past, whereas collaborative recommendation systems find users with preferences similar to those of the given user and recommender items based on their interests. In this paper, we will first discuss the basic concepts of content-based recommenders. Section 2.1 discuss the advantages and drawbacks of the content-based approach. Section 2.2 discussed each of the components of a content-based recommender in detail and describes the current techniques used for each of them. Section 3 describes our proposed extensions to a visual and relational recommender developed by McAuley et al [1]. Visual recommender systems aim to model human visual preferences based on characteristics of images associated with the product. They are closely related to content-based recommender systems and can be used for recommending items to the user based on his previous product searches/purchases.

2 Content-based Recommender Systems

Content-based recommender systems suggest items to users based on the items he has liked previously. Such systems analyze the set of items rated by a user and build a profile of his interests based on the features/attributes of the items he has rated. [16]. The recommendations process consists of matching the attributes of a user profile which contains interests of that user, with the attributes of an item profile which contains features of the item, to recommend new interesting items to the user. The key challenge is to build a user profile which accurately reflects user preferences.

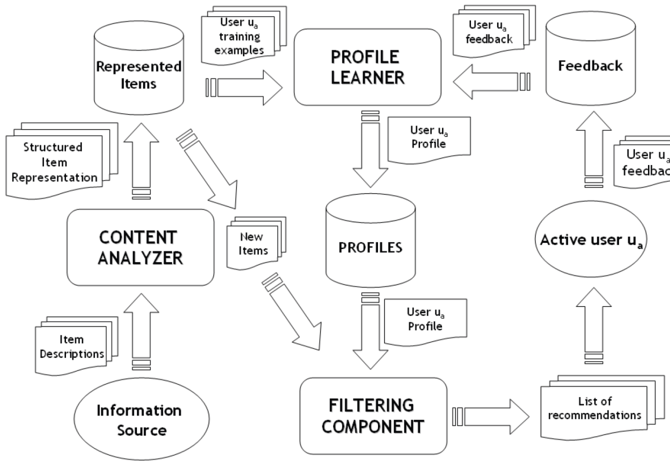


Figure 1: High level architecture for a content-based recommender system (source: Lops et al [5])

Figure 1 shows the high level architecture for a content based recommendation system [5]. The keys steps components are as follows:

- **Content Analyzer** - In case of unstructured information sources (e.g. documents, Web pages, news articles), a preprocessing step is required to extract relevant information from the content of items. Feature extraction techniques are used to shift item representation from the original space to the new target space (e.g. from corpus to word vectors in case of text). This task of preprocessing the content to create item representation is performed by a Content Analyzer whose output subsequently goes to Profile Learner and Filtering Component.
- **Profile Learner** - This module gets the preprocessed item representations only for the set of items which a particular user has rated in the past. It uses machine learning techniques to learn the profile of a user based on the items liked or disliked in the past. If users have provided feedback to the items presented, it together with the item profile, can be used for learning the relevance of an item to a user.
- **Filtering Component** - This component matches the information available in a user profile against the set of items which can be recommended. The output of the Filtering Component can be binary (recommend or not recommend) or continuous (using similarity metrics between user profile and item profile [18]) which results in a ranked list for the items which can be of interest to the user.

2.1 Advantages and Drawbacks of Content-based Systems

Content-based recommendation paradigm has several advantages when compared to collaborative paradigm:

- **User independence** - Content-based systems depend only on the ratings provided by the active user to build her profile. They don't depend on the ratings of other users with similar tastes to the active user. On the other hand, collaborative filtering required ratings from other users in order to find the most similar users of the active user.

- **Transparency** - Justifications for recommending an item to a user can be provided based on the description of that item and its similarity to the ones rated highly by the user in the past. The only reason collaborative systems can provide is that there are unknown users with similar tastes who have liked that item.
- **New item** - Content-based systems can provide recommendations for items that are not yet rated by any user. Collaborative systems can't provide recommendations for a new item because they depend on the preferences of users' who have liked an item.

Content-based systems have the following drawbacks as well:

- **Limited Content Analysis** - The effectiveness of content based systems is limited by the features associated with the items to be recommended. No content-based system will provide good recommendations to a user if the features do not contains enough information to distinguish between the items which user likes from the ones she does not like. Information retrieval techniques work well for the extraction of text based features, but for multimedia content such as images, videos and audios, feature extraction techniques are harder to apply. Even in text-based feature extraction, documents are generally represented by a set of important keywords, content-based systems are unable to differentiate between an informative article and a poorly written one if they represented by the same keywords [19].
- **Overspecialization** - Content based recommender are not able to provide recommendation that are serendipitous or unexpected. They can only recommend items that have a higher similarity when compared to user's profile, and the user is limited to recommendation of items which are similar to those she has rated [2]. For example, if a user has seen only action movies, she won't be recommended a comedy movie.
- **New user** - The user should provide ratings for a sufficient number of items before the system can understand user's preferences and provide relevant recommendations. Therefore, if few ratings are available for a user, the system would not be able to provide accurate recommendations.

2.2 Components of a Content-based System

Items which are recommended to a user can be represented as a set of features, which are also known as 'attributes' or 'properties' of that item. Structured data usually has smaller number of attributes, each item is represented by the same set of attributes which can take values from a known finite set. For example, in a movie recommender system, the features for representing a movie can be genres, actors, directors, language etc. Pazzani et al. [20] describes several algorithms which can be used to learn user profile from structured data. Typically, they are classification algorithms learning on a set of feature vectors.

In most content-based systems, information about items need to be extracted from unstructured data present in form of text on web pages, news articles or product descriptions. The natural language text can have polysemous words (same word with different meanings) and synonyms (different word with same meaning), which make it difficult for keyword based representation to capture the attributes of an item accurately. User profiles from such representations do not capture the semantics of user interests because they string match a set of keywords present in the user profile against the item description and considering it relevant if a match occurs. Semantic analysis is proposed in literature to solve some of these problems by adopting knowledge bases, such as lexicons or ontologies, for capturing the semantics of user interests as well. Section 2.2.1 describes basic keyword based approached and subsequently, we describe approaches based on semantic analysis in Section .

2.2.1 Keyword-based Item Representations

Most content-based systems use traditional keyword based approaches to represent the unstructured text. Let $D = d_1, d_2, \ldots d_N$ denotes the set of documents (or item descriptions) and $T = \{t_1, t_2, \ldots, t_n\}$ be the vocabulary of keywords selected from the corpus. Each document d_j is represented as a n-dimensional vector as $d_j = \{w_{1j}, w_{2j}, \ldots, w_{nj}\}$ where w_{kj} represents the weight of term t_k in document d_j. Empirically, TF-IDF (Term Frequency-Inverse Document Frequency) works well for text-based feature vectors. It is based on the intuition that more frequent terms (terms with high TF values) in a document which occur rarely in other documents tend to be more relevant for representing the document. Additionally, normalizing the TF-IDF weight vectors prevents documents from being considered as more relevant just because of their high length. The TF-IDF weight is defined as follows [5]:

$$\mathtt{TF}-\mathtt{IDF}(t_k,d_j) = \mathtt{TF}(t_k,d_j)\log\frac{N}{n_k} \tag{1}$$

$$\mathtt{TF}(t_k,d_j) = \frac{f_{k,j}}{max_z f_{z,j}} \tag{2}$$

Here N denotes total number of documents, n_k denotes the number of documents in which the term t_k appears at least once and $max_z f_{z,j}$ denotes the maximum frequency of a word in document d_j. The weights are usually normalized as:

$$w_{k,j} = \frac{\mathtt{TF}-\mathtt{IDF}(t_k,d_j)}{\sqrt{\sum_{s=1}^{|T|}\mathtt{TF}-\mathtt{IDF}(t_s,d_j)^2}} \tag{3}$$

For measuring closeness between two documents, a similarity measure such as cosine similarity is used.

$$sim(d_i,d_j) = \frac{\sum_k w_{ki}w_{kj}}{\sqrt{\sum_k w_{ki}^2}\sqrt{\sum_k w_{kj}^2}} \tag{4}$$

In case explicit user feedback regarding items is not present, user profile can be represented as weighted term vectors averaged over the items bought by the user. User's interest in a particular items can be measured by computing the cosine similarity.

2.2.2 Item Representations Based on Semantic Analysis

Semantic analysis helps in the constructing more accurate profiles by using the knowledge present in external lexical databases. One of the lexical database widely used for this purpose is WordNet.

WordNet [1] [21] is one of the largest lexical databases that has been used in learning sense-based representations of documents. The fundamental block in WordNet is the synset which is an unordered set of synonyms belonging to same POS (Parts-of-Speech). There are multiple relations present between synsets such as antonymy (opposites such as {wet} and {dry}), hyperonymy (also called hyponymy or IS-A relation). IS-A relation is the most frequently encoded and it links more general synsets like {furniture, piece_of_furniture} to increasingly specific ones like {bed} and {bunkbed}. It is a hierarchical relation with all hierarchies ultimately ending up at a root node e.g. noun hierarchies end up at root node {entity}.

ITR (Item Recommender) [23] is system which can provide recommendations in several domains such as movies, music and books with the constraint that item descriptions are in the form of text. It uses a Word Sense Disambiguation (WSD)[2] based approach to get a sense-based representation of document. In *ITR*, items are represented as bag-of-synsets (BOS) which is an extension to the traditional bag-of-words approach. The vocabulary consists of all the synsets present in the corpus. Each monosemous word w in a document is matched to the corresponding WordNet synset. For polysemous words, a Word Sense Disambiguation (as shown in Figure 2) procedure is applied to identify the correct synset. The feature vector for a document is a vector of the TF-IDF score of the synsets present in the document .

SEWeP (Semantic Enhancement for Web personalization) [24] is a web personalization system which exploits the semantic information present on a web page for personalization. First, a domain specific taxonomy of categories is manually created and is used to semantically annotate the web pages automatically. Each web page is represented by the set of categories present in it. If a word on the web page belongs to the taxonomy, it is included as it is. If it not, then the closest keyword to it present in the taxonomy is included. The closeness is determined using a word similarity measure based on WordNet.

[1] http://wordnet.princeton.edu

[2] As per Wikipedia, "WSD is identifying which sense or meaning of a word is used in a sentence, when the word has multiple meanings."

Algorithm 1 The WordNet-based WSD algorithm

```
1: procedure WSD($w, d$)    ▷ finds the proper synset of a polysemous word $w$ in document $d$
2:     $C \leftarrow \{w_1, \ldots, w_n\}$    ▷ $C$ is the context of $w$. For example, $C = \{w_1, w_2, w_3, w_4\}$ is a window
           with radius=2, if the sequence of words $\{w_1, w_2, w, w_3, w_4\}$ appears in $d$
3:     $X \leftarrow \{s_1, \ldots s_k\}$ ▷ $X$ is sense inventory for $w$, that is the set of all candidate synsets for $w$
           returned by WordNet
4:     $s \leftarrow null$    ▷ $s$ is the synset to be returned
5:     $score \leftarrow 0$    ▷ $score$ is the similarity score assigned to $s$ with respect to the context C
6:     $T \leftarrow \emptyset$ ▷ $T$ is the set of all candidate synsets for all words in $C$
7:     for all $w_j \in C$ do
8:         if $POS(w_j) = POS(w)$ then    ▷ $POS(y)$ is the part-of-speech of $y$
9:             $X_j \leftarrow \{s_{j1}, \ldots s_{jm}\}$    ▷ $X_j$ is the set of $m$ possible senses for $w_j$
10:            $T \leftarrow T \cup X_j$
11:        end if
12:    end for
13:    for all $s_i \in X$ do
14:        for all $s_h \in T$ do
15:            $score_{ih} \leftarrow$ SinSim($s_i, s_h$)    ▷ computing similarity scores between $s_i$ and every synset
                   $s_h \in T$
16:            if $score_{ih} \geq score$ then
17:                $score \leftarrow score_{ih}$
18:                $s \leftarrow s_i$    ▷ $s$ is the synset $s_i \in X$ having the highest similarity score with respect to
                       the synsets in $T$
19:            end if
20:        end for
21:    end for
22:    return $s$
23: end procedure
24: function SinSim($a, b$)    ▷ The similarity of the synsets $a$ and $b$
25:    $N_p \leftarrow$the number of nodes in path $p$ from $a$ to $b$
26:    $D \leftarrow$maximum depth of the taxonomy    ▷ $D = 16$ in WordNet 1.7.1
27:    $r \leftarrow -log(N_p/2D)$
28:    return $r$
29: end function
```

Figure 2: Word Sense Disambiguation algorithm (source: Degemmis et al [23])

It has been observed that the features which use both external knowledge from lexical databases and knowledge from documents perform better than the ones which use only the information extracted from documents. Many resources of external knowledge such as Wikipedia, Yahoo Answers have become available in recent years. Explicit Semantic Analysis [25] is example of a technique which is able to provide a representation of text documents in a space of concepts (wikipedia articles, e.g. *clustering*, *data mining*, *algorithms*) derived from Wikipedia. Mihalcea et al. [26] propose a system called Wikify! which exploits Wikipedia knowledge for selecting important information from a text (keyword extraction) and disambiguating the links (word sense disambiguation). The concept of using semantic information from Encyclopedia knowledge bases has not been used for learning user profiles in content-based recommender systems and it seems a promising research area.

2.2.3 Learning User Profiles

User profile are models which represent the preferences of user towards a set of items. In scenarios, when user feedback is not known regarding what the user likes and dislikes, user profile represents an averaged feature representation of the items' profile she has bought. User is recommended items based on a similarity measure (e.g. cosine distance) between user and item feature vectors. Random-hyperplane and locality-sensitive-hashing (LSH) techniques could also be used to place item profiles in buckets. For comparing item profile with user profile, we can determine the same techniques to find out the item buckets which are at lower cosine distance from the user [22].

If information about "items the user likes" and "items the user doesn't like" is present or the user has rated the items, then learning user profiles becomes a form of classification task and methods such as decision trees, Nearest neighbor classifier, or probabilistic method such as Naive Bayes are used.

2.3 Conclusion and Future Trends

We surveyed the field of content-based systems and provided an overview of the important aspects involved in the design. We discussed the key components involved in a typical content-based system. We also talked about the issues related to item representation, starting from simple feature vector based representations to more complex representations based on semantic analysis for unstructured data. A promising research area of interest is to use semantic knowledge from Encyclopedia sources for learning user profiles.

Content-based systems suffer from the problem of over-specialization, the system cannot recommend completely different items from what user has seen before. Some of the recommender systems such as Daily-Learner [27] do not recommend items to users if they are too similar what user has already rated. According to Gup's theory [28], content-based systems cannot naturally generate serendipitous recommendations. Iaquinta et al. [29] propose some heuristics to provide serendipitous recommendations. The key idea is that if the user lower the chances that user knows an item, higher will be the probability of it being serendipitous.

We believe that one of the interesting challenges is to provide serendipitous but relevant recommendations. Content-based systems also suffer from the 'New User' problem and are unable to provide good recommendations unless enough data about user's preferences has been collected. In the next section, we describe the work done on extending a visual recommender system which can be helpful in addressing some of the limitations of content-based recommender systems.

3 Visual Recommender System

We are interested in modeling the human notion of visual appearance, in particular what objects go well with each other (*compliments*) and which can be seen as possible alternatives for an object (*substitutes*). We are building on the work done by McAuley et al. [1] which had the following contributions: (1) The authors propose a method which is capable of modeling a variety of visual relationship beyond visual similarity. They show that human notion of visual relationship can be modeled using a suitably large dataset, even if it is tangentially related to the quantity we are trying to model; (2) The authors provide a dataset (referred to as *Styles and Substitutes* dataset) of relationships between objects from Amazon web store (refer Section 3.1).

We observe that the distance metric proposed by McAuley et al [1] is not able to capture the notion of relationship when the two products are visually different but belong to the same style (e.g. a white shirt and a black jeans). In this work, we provide extensions to the work by providing multiple similarity scores which are able to capture the notion of visual relatedness in a better way. We also include other information, such as the category information, present in the dataset as our features so that the model does not have to explicit learn it. (e.g. product-pair belonging to categories $boys -> badminton\ shoes -> Nike$ and $boys -> badminton\ shoes -> Adidas$ tend to be related and they have multiple categories in common). We also show that application of clustering to learn multiple distance metrics helps in improving the performance.

3.1 Dataset description

We use a portion of *Styles and Substitutes* dataset [1] for our analysis. The original dataset contains over 180 million relationships among 6 million objects from the Amazon web store. It was obtained by visiting the web store and recording the product recommendations given by Amazon. The relationships in the dataset describe two notions of interest: *compliments and* substitutes. Compliment products are the ones which might be purchased together (e.g. a matching pair of trouser and shoes) while substitute products are the ones which are an alternate for each other (e.g. a pair of Nike sports shoes and a pair of Adidas sports shoes).

In terms of product pairs (X,Y) there are 4 types of directed relationships recorded in the dataset: 1) users who viewed X and also viewed Y (65M edges) - "also viewed"; 2) users who viewed X eventually bought Y (7.3M edges) - "buy after viewing"; 3) users who bought X also bought Y (104M edges) - "also bought"; 4) users who bought X and Y simultaneously (3.4M edges) - "bought-together". Categories 1 and 2 can be approximated as *substitutes while 3 and 4 as* complements].In addition to these relationship graphs, each product is associated with a product-id, product-image and a set of hierarchical category labels. Further, a non-complete set of meta-data information such as reviews, brand, and price is also included. Amazon's tech reports mentions that recommendations are made based on the cosine similarity of the set of users that bought/viewed the product.

3.2 Relevant Literature

The closest systems to the visual recommender systems we propose are content-based recommender systems [5] which try to recommend items similar to those a given user has liked in the past by using the meta-data available with the items. We describe the drawbacks of content-based systems in Section 2.1. Visual recommender systems could also be effective in address 'new user' and over-specialization problems.

The most relevant systems to ours is the visual recommender system proposed by McAuley et al. [1] which tries to generalize the idea of a visual distance measure. The authors make an attempt to model human preferences for visual appearances rather than modeling visual similarity between objects.

We extend this idea by using different distance functions (in Section 3.3), so that the model generalizes and captures the relationships between objects that are visually different (e.g. white shirt and black pants are visually different but often co-purchased together.). We include the "category" meta-data in our models and compare its performance with baseline model. We then design models which combine both types of information (images and category data) and then evaluate their performance. We also propose a clustering based extension so that different clusters are able to learn multiple notions of 'similarity'. It is discussed in more details in Section 3.3.

3.3 Features and Model Description

In this section, we present a description of the features and the model that we use. As mentioned in Section 3.1, we use the *Styles and Substitutes* dataset provided by McAuley et al [1]. The dataset consists of a feature vector of length $F = 4096$ for each product image which are calculated from the original images using Caffe deep learning framework [6]. As per [1], "A Caffe reference model (*caffe.berkeleyvision.org*) with 5 convolutional layers was used followed by 3 fully-connected layers, which has been pre-trained on 1.2 millon ImageNet (ILSVRC2010) images. The output of FC7, the second fully-connected layer results in feature vector of length $F = 4096$".

Besides image features, we also use category features in some of the distance functions proposed in the extension. We have hierarchical category information with each product. We find all the unique categories present in the dataset and then prune them based on the counts of products associated with each categories. We use cat_{ij}, number of categories in between object i and j intersect, as the category-based feature.

Our notation is defined in Table 1. We choose relatively simple linear models because they need to scale to the volume of data. For each object, an $F-$ dimensional feature vector $x \in \mathbb{R}^F$ representing

Notation	Description
x_i	feature vector for object image i
F	dimension of image feature vector x_i
r_{ij}	relationship between objects i and j
R	set of relationship between all objects
$d_\theta(x_i, x_j)$	parametrized distance between x_i and x_j
M	$F \times F$ Mahalanobis transform matrix
U	a $F \times K$ matrix for approximating M
V	a $F \times K$ matrix for approximating M
cat_{ij}	Number of categories which intersect between object i and j
γ	Parameter learnt for cat_{ij}
N_{cat}	Number of categories present after pruning
C	Number of clusters (distance functions of each type) used
$\sigma_l(.)$	shifted sigmoid function with parameter l

Table 1: Notation description

the object image, is obtained from the *Styles and Substitutes* dataset. The category feature cat_{ij} and calculated using the category information present as part of the meta-data. The dataset also contains a set R of relationships where each $r_{ij} \in R$ denotes that objects i and j are related. We learn a parametrized distance function $d(x_i, x_j)$ such that feature vectors x_i, x_j for objects which are related ($r_{ij} \in R$) are assigned a lower distance than the ones which are unrelated ($r_{ij} \notin R$)]. Or in other words,

$$P(r_{ij} \in R) \propto -d_\theta(x_i, x_j, cat_{ij}) \tag{5}$$

We use a shifted sigmoid function for mapping distance to a probability value.

$$P(r_{ij} \in R) = \sigma_l(-d_\theta(x_i, x_j, cat_{ij})) = \frac{1}{1 + exp(d_\theta(x_i, x_j, cat_{ij}) - l)} \tag{6}$$

The intuition behind this is for two items i and j: (1) If $d_\theta(x_i, x_j, cat_{ij}) = l$, probability i and j are related $= 0.5$; (2) If $d_\theta(x_i, x_j, cat_{ij}) > l$, probability i and j are related < 0.5; (3) If $d_\theta(x_i, x_j, cat_{ij}) < l$, probability i and j are related > 0.5 The parameter l is also learnt by the model to maximize the log-likelihood. We also refer relationships as positive edges and non-relationships as negative edges. We will now discuss the distance functions that we used in our models:

3.3.1 Distance Functions

Mahalanobis transorm (MT)
We use this distance function (proposed by McAuley et al. [1])as a baseline to compare against the distance functions proposed by us. The authors use Mahalanobis transform (as opposed to Weighted nearest neighbor (WNN[3]) because it captures information about how different feature dimensions co-relate with each other. The distance function is of the form:

$$d_M(x_i, x_j) = (x_i - x_j)\mathbf{M}(x_i - x_j)^T \tag{7}$$

A full-rank positive semidefinite matrix M requires approximately 8 million parameters to fit ($F = 4096$), therefore a low rank approximation U of dimension $F \times K$ is used by McAuley et al. [1]), such that $M \simeq UU^T$.

$$\begin{aligned} d_U(x_i, x_j) &= (x_i - x_j)\mathbf{UU^T}(x_i - x_j)^T \\ &= ||(x_i - x_j)U||_2^2 \end{aligned} \tag{8}$$

For K=1, the above model becomes a weighted nearest neighbor (WNN) model.

Visual Relationship Score (VRS)
MT distance function is able to capture the relationship between products having similar visual appearance (e.g. black pants related to black shoes) but it might not be able to capture the relationship between products which are visually different (e.g. a white shirt matches with blue jeans). In other words, MT distance function gives higher weights to the atrributes which are similar in related products and ignores (gives lesser weight) to the attributes which are dissimilar in them. It only learns the similarities between the items. VRS, on the other hand, is able to capture the similarity in some attributes (using the first term $||(x_i - x_j)\mathbf{U}||_2^2$) and dissimilarity in other (using the second term $||(x_i - x_j)\mathbf{V}||_2^2$). Based on this intuition, the distance function for VRS is:

$$d_{U,V}(x_i, x_j) = ||(x_i - x_j)\mathbf{U}||_2^2 - ||(x_i - x_j)\mathbf{V}||_2^2 \tag{9}$$

An important point to note that is $d_{U,V}(x_i, x_j)$ does not follow the properties of a metric i.e. $d_{U,V}(x_i, x_j)$ can take negative values as well. Another way to think about VRS is that we have relaxed the condition on matrix M to be positive semi definite because it is approximated as $(UU^T - VV^T)$ now in the hope that it is able to provide a generalized classifier.

Category Parameter (CP)
Product category is an important aspect in describing relationships and MT distance function tries to learn it implicitly from the data using visual features. Our intuition is that if we encode the category information explicity, the model need not learn it from the visual data, which, in turn can result in better performance.

[3]A weighted nearest neighbor can be one of the simplest method to learn which feature dimensions are importance for a particular relationship. It is of the form: $d_\theta(x_i, x_j) = ||\theta^T(x_i - x_j)||$

While performing exploratory analysis, we find that related products intersect in more categories as compared to unrelated products. Therefore, we choose c_{ij}, number of categories items in i and j intersect, as a feature. We consider only the categories with a product count greater than a certain threshold. The distance function for CP is defined as:

$$d_\gamma(cat_{ij}) = \gamma cat_{ij} \tag{10}$$

Again, $d_\gamma(cat_{ij})$ need not be a metric.

We also propose models which combine both the image features and category features. The motivation behind this is that we should be able to obtain a better model by combining both types of information.

MT + CP
MT+CP distance function is defined as:

$$d_{U,\gamma}(x_i, x_j, cat_{ij}) = ||(x_i - x_j)\mathbf{U}||_2^2 + \gamma c_{ij} \tag{11}$$

VRS + CP
VRS+CP distance function is defined as:

$$d_{U,V,\gamma}(x_i, x_j, cat_{ij}) = ||(x_i - x_j)\mathbf{U}||_2^2 - ||(x_i - x_j)\mathbf{V}||_2^2 + \gamma cat_{ij} \tag{12}$$

MT + Clustering
We propose a clustering based extension for MT distance function. The idea is that there could be multiple different notions of 'similarity' at play simultaneously (e.g. a black shoe pair and a white shoe pair are similar in 'shape' but different in 'color' while a black jeans and a black jacket are similar in 'color' but different in 'shape'). If we use multiple clusters and each cluster learns a different kind of similarity measure for the set of positive edges and negative edges, they will be able to capture different notions of similarity.

The clustering algorithm if formally written in the box below. For Step 5 in the algorithm, we have two options: (1) cluster for $(x_i, x_j) = \mathtt{argmin}_c d_{U_c}(x_i, x_j)$ and (2) cluster for $(x_i, x_j) = \mathtt{argmax}_c d_{U_c}(x_i, x_j)$. Step 6 is the learning phase in which each cluster is trying to bring the related objects closer in space and the unrelated ones farther in space for the set of edges assigned to it. If we look at an overall picture, option 1 ('argmin') tries to minimize the distance for related objects in at least one cluster and maximize the minimum distance for unrelated objects across all the clusters. On the other hand, Option 2 ('argmax') tries to minimize the maximum distance for related objects across all the clusters and maximize distance for negative edges in at least one cluster.

We further explain both the options with the help of a small example. Let's say there are two clusters c_1 and c_2, and c_1 learns similarity in 'color' while c_2 in 'shape'; (1) If related objects are similar in either 'color' or 'shape' in the dataset, then in option 1 ('argmin'), the positive edge for similarity in color will be assigned to c_1 and the one for similarity in 'shape' will be assigned to c_2. Option 2 ('argmax') would not be able to correctly assign a suitable cluster for the positive edges. (2) If the related objects are similar in both 'color' and 'shape' in the dataset, in Option 1 ('argmin'), c_1 will learn similarity in 'color' from a subset of positive edges and c_2 will learn similarity in 'shape' from another subset of them. On the other hand, in Option 2 ('argmax'), c_1 will learn 'color' similarity and c_2 'shape' similarity from all the related objects and it can learn those attributes in a better way.

We conducted experiment for multiple dimensions, K and clusters, C on different datasets and Option 2 ('argmax') came out to be unanimously better than Option 1 ('argmin') every time. We, therefore, decided to move forward with Option 2 ('argmax') for comparison against baseline model in Section 3.4.

Algorithm 1: Clustering Algorithm

```
1 Parameters: $U_c$ and $l_c$ for each cluster c = 1 ... C
2 Hyper-parameters: Number of clusters ($C$) and number of iterations ($N$)
3 for $itr$ = 1 ... $N$ do
4     for all edge pairs $(x_i, x_j)$ in the training set do
5         cluster for $(x_i, x_j) = \mathtt{argmax}_c / \mathtt{argmin}_c d_{U_c}(x_i, x_j)$;
6     for $c$ = 1 ... $C$ do
7         Learn $U_c$ and $l_c$ by maximizing log-likelihood;
```

3.3.2 Training

The original dataset provides only the set of positive relationships R i.e. pairs of items which are related. We randomly select a negative set $Q = \{r_{ij} | r_{ij} \notin R\}$ such that $|Q| = |R|$. We have associated a probability for the presence (absence) of each relationship, therefore we can estimate the parameters by maximizing the likelihood of complete set of relationships and non-relationships as:

$$ll(\mathbf{U}, \mathbf{V}, \gamma, l) = \sum_{r_{ij} \in R} \log\left(\sigma_l(-d_\theta(x_i, x_j))\right) + \sum_{r_{ij} \in Q} \log\left(1 - \sigma_l(-d_\theta(x_i, x_j))\right) \tag{13}$$

For maximizing $ll(\mathbf{U}, \mathbf{V}, \gamma, l)$, we use gradient ascent method. We use hybrid L-LBFGS method [15] for this optimization problem. The gradient computations can be easily parallelized over the pairs $r_{ij} \in R \cup Q$.

3.4 Experimental Results

From the *Styles and Substitutes* dataset, we perform our experiments on products belonging to "Clothing, Shoes, and Jewelry" category. Due to limited compute resources, we choose only "Boys" and "Girls" subcategories of "Clothing, Shoes, and Jewelry" category for our experiments. We consider two types of relationship graphs - "also-viewed" (approximated as *Substitutes*) and "also-bought" (approximated as *Compliments*). *Table* 2 summarizes the number of items and positive relationships present in each dataset. We added the same number of negative relationships as positive in each dataset by selecting an edge randomly between pairs of products which do not contain a positive edge.

Subcategory	**Edge-type**	**# items**	**# edges (positive)**
Boys	Also-bought Also-viewed	54677	350004 375564
Girls	Also-bought Also-viewed	71068	418530 503038

Table 2: Description of the datasets considered for experiments

Distance Function	**Dimension**	**Also-bought (Compliments)** Test	**Also-viewed (Substitutes)** Test

MT
(baseline)
	K=1
	K=2
		74.5%
		80.2%
			81.4%
			85.9%

VRS	K=1
	K=2
		79.1%
		83.8%
			82.2%
			86.3%

CP	NA	76.0%	81.2%

MT + CP	K=1
	K=2
		78.6%
		81.4%
			87.6%
			88.7%

VRS + CP	K=1
	K=2
		83.5%
		87.0%
			88.4%
			89.8%

Table 3: Accuracies of link prediction on subcategory 'Boys' of 'Clothing' category considering 25% of total edges

Table 3 and 4 contain the link-prediction accuracies for 25% of the edges present in 'Boys' and 'Girls' sub-categories respectively. The further reduction in the number of edges considered is due to limited compute resources.

We compare the performance of proposed distance functions on the test set. For all of them, it is observed that the accuracy on test set increases from $K = 1$ to $K = 2$ because $K = 2$ is a better approximation for matrix M (refer Section 3.3.1). The performance for "also-viewed" edges is better than "also-bought" probably because "also-viewed" edges mostly consists of visually similar product-pairs belonging to same category while "also-bought" mostly contains visually different product-pairs belonging to different categories. The negative edges are randomly selected and therefore are more likely to contain product-pairs from different categories. As a result, the models are able to differentiate easily between negative-edges (dissimilar in many attributes) and "also-viewed" positive edges (similar in many attributes). We see that all distance functions (except CP) outperforms the baseline MT distance function for both edge types and both sub-categories.

For category based feature, we have set the pruning count to be 100 i.e. (categories with product count > 100 are only considered). We did not observe significant variation in validation accuracy on changing the pruning count. One interesting aspect to note that CP distance function is able to give comparable performance to MT distance with $K = 1$ even though CP contains a single feature. This shows the importance of category information in identifying the relationships.

Overall, we see that VRS+CP model performs the best among all the distance functions. Intuitively, this is expected because it is combining both the idea of including a generalized distance function and exploiting category information.

Distance Function	**Dimension**	**Also-bought (Compliments)** Test	**Also-viewed (Substitutes)** Test
MT (baseline)	K=1	75.9%	81.3%
	K=2	81.3%	85.9%
VRS	K=1	79.3%	82.1%
	K=2	84.7%	86.3%
CP	NA	78.6%	86.3%
MT + CP	K=1	80.2%	88.2%
	K=2	82.6%	89.5%
VRS + CP	K=1	84.6%	89.2%
	K=2	87.7%	90.6%

Table 4: Accuracies of link prediction on subcategory 'Girls' of 'Clothing' category considering 25% of total edges

Distance Function	**Dimension**	**Also-bought (Compliments)** Test	**Also-viewed (Substitutes)** Test
MT (baseline)	K=2, C=1	80.2%	85.9%
MT + Clustering	K=1, C=2	79.0%	84.6%
MT + Clustering	K=1, C=3	81.3%	85.6%

Table 5: Accuracies of link prediction on subcategory 'Boys' of 'Clothing' category considering 25% of total edges

We present our results of extension based on clustering in Table 5 and 6. We see that the performance on test set improves for all datasets as we increase the number of clusters, C. We also compare our results against for $K = 1, C = 2$ with MT distance for $K = 2, C = 1$ (baseline) because both these distance functions have same number of parameters. The performance on test set shows that it is better to increase K as opposed to number of clusters, C for both edges types and both sub-categories as clustering is not providing any extra improvement.

Distance Function	**Dimension**	**Also-bought (Compliments)** Test	**Also-viewed (Substitutes)** Test
MT (baseline)	K=2, C=1	81.3%	85.9%
MT + Clustering	K=1, C=2	80.2%	84.9%
MT + Clustering	K=1, C=3	82.4%	85.1%

Table 6: Accuracies of link prediction on subcategory 'Girls' of 'Clothing' category considering 25% of total edges

We also conducted experiments on the full dataset for 'Girls' subcategory for "also-bought" edges and see if clustering is giving improvement as compared to increasing K. We found that MT + clustering (Full - row 1 and row 2 in Table 7) gives a slightly worse performance than MT distance. We then further removed the edges between two products of the same category from the dataset so that "also-bought" edges represents a pair of *compliment* products more accurately. For the 'pruned' case (Pruned - row 3 and row 4 in Table 7), we again find that clustering does not give performance improvement over MT for the test set.

Dataset	**Distance Function**	**Dimension,**	**Also-bought (Compliments)**		
		# clusters	Train	Validation	Test
Full	MT (baseline)	K=3, C=1	86.2%	85.1%	85.0%
Full	MT + Clustering	K=1, C=3	86.8%	85.0%	84.8%
Pruned	MT	K=3, C=1	95.6%	91.4%	91.1%
Pruned	MT + Clustering	K=1, C=3	98.5%	90.3%	89.4%

Table 7: Accuracies of link prediction on subcategory 'Girls' of 'Clothing' category for "also-bought" edges

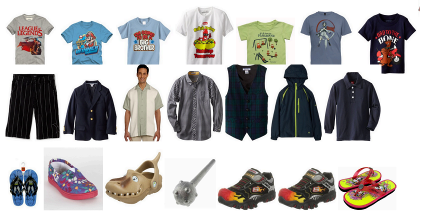
3.4.1 Visualization

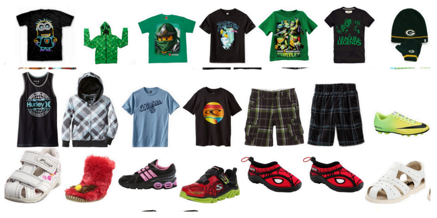
McAuley et al. [1] mention that low-rank Mahalanobis transform (eq. 8) can be seen as euclidean distance between transformed feature vector s_i ($s_i = x_i U$) and s_j ($s_j = x_j U$) and the equation also be written as:

$$d_U(x_i, x_j) = ||s_i - s_j||_2^2 \tag{14}$$

s_i and s_j can be seen as low-dimensional embedding of x_i and x_j. The authors refer to low-dimensional embedding as product's representation in 'style-space'.

We can extend the same idea for visualization for MT+Clustering distance transform, considering a 'style-space' for each cluster. We perform principal component analysis over the set of products present in each cluster and display some of the products with highest principal components in each dimension (1..K) in Figure 3 The items with similar shapes and color belong to a dimension in a cluster. Also, subtle characteristics of style such as, jackets separated from t-shirts and shoes separated from boots, are also captured.


(a)


(b)

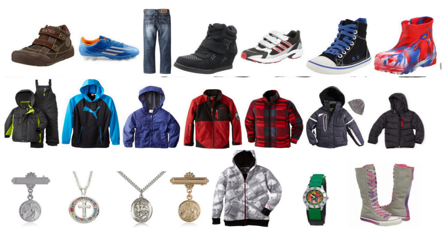

(c)

Figure 3: Examples of items in style space for 3 clusters ($C = 3$) and 3 dimensions ($K = 3$) (Girls' Clothing "also-bought" dataset). Each subfigure denotes a cluster

3.5 Generating Recommendations

Given a query item or a product which user is currently browsing or has purchased recently, we want to recommend him a set of other items which might be of interest to him. All the proposed extensions can be used to generate recommendations following the methodology used by McAuley et al.[1]. For example, if a user is browsing products from "clothing, shoes and jewelry" category, we can recommend him/her items from the subcategories such as shoes, school uniforms, watches etc. Specifically, for a query item x_q, for each sub-category C, we provide recommended items as:

$$\texttt{argmax}_{j \in C} P(r_{qj} \in R) \tag{15}$$

i.e. the item with the minimum distance measure in each C.

3.6 Conclusions and Future Work

We gained performance improvement over the baseline model from using VRS distance function because it is able to capture the notion of similarity in some attributes but different in others. Category information turned out to be very important for learning relationships and the distance functions using category information in addition to visual features performed better. Clustering did not perform well as expected. Increasing dimensions (K) came out to be better as opposed to increasing number of clusters (C).

In future, we plan to run the experiments on higher K values and also for data belonging to other categories for the distance functions proposed. We would also try to evaluate the quality of our recommendation and compare with baseline recommendation methods using suitable metrics. We would also possibly include some domain adaption to the convolutional neural network based features extracted using Caffe framework. We also plan to compare the performance with other forms of metric learning such as max-margin optimization.

Acknowledgement

I would like to thank my advisor, Prof. Julian McAuley, for his continued mentorship and help throughout project. I would also like to thank my chair, Prof. Henrik Jensen and committee members, Prof. Lawrence Saul and Prof. Kamalika Chaudhuri for their patience and co-operation throughout the process.

References

[1] J. J. McAuley, C. Targett, Q. Shi and A. van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDD*, 2005.

[3] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*. Springer US, 2011.

[4] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*. Springer, 2011.

[5] P. Lops, M. de Gemmis and G. Semeraro. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*. Springer, 2011.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[7] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Trans. on PAMI*, 2007.

[8] W. Di, C. Wah, A. Bhardwaj, R. Piramuthu, and N. Sundaresan. Style finder: Fine-grained clothing style detection and retrieval. In *CVPR Workshops*, 2013.

[9] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

[10] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, 2011.

[11] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes paris look like paris? In *SIGGRAPH*, 2012.

[12] P. Peng, L. Shou, K. Chen, G. Chen, and S. Wu. The knowing camera 2: Recognizing and annotating places-of-interest in smartphone photos. In *SIGIR*, 2014.

[13] K. Xu, H. Li, H. Zhang, D. Cohen-Or, Y. Xiong, and Z.-Q. Cheng. Style-content separation by anisotropic part scales. *ACM Trans. on Graphics*, 2010.

[14] K. Xu, H. Li, H. Zhang, D. Cohen-Or, Y. Xiong, and Z.-Q. Cheng. Style-content separation by anisotropic part scales. *ACM Trans. on Graphics*, 2010.

[15] Y. Liu, W. Wang, B. Levy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal Voronoi tessellation energy smoothness and fast computation. *ACM Trans. on Graphics*, 2009.

[16] D. Mladenic. Text-learning and Related Intelligent Agents: A Survey. *IEEE Intelligent Systems 14(4), 4454*, 1999.

[17] J. Rocchio. Relevance Feedback Information Retrieval. In *G. Salton (ed.) The SMART retrieval system - experiments in automated document processing, pp. 313323. Prentice-Hall, Englewood Cliffs, NJ*, 1971.

[18] L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems 22(1), 553*, 2004

[19] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating Word of Mouth. In *Conf. Human Factors in Computing Systems*, 1995.

[20] M.J. Pazzani, D. Billsus. Content-Based Recommendation Systems. In P. Brusilovsky, A. Kobsa, W. Nejdl (eds.) The Adaptive Web, Lecture Notes in Computer Science, vol.4321, pp. 325341, 2007

[21] George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM Vol. 38, No. 11: 39-41*, 1995

[22] A. Rajaraman and J.D. Ullman. Mining of massive datasets. *Cambridge University Press*, 2011.

[23] M. Degemmis, P. Lops, G. Semeraro. A Content-collaborative Recommender that exploits WordNet-based User Profiles for Neighborhood Formation. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research (UMUAI) 17(3), 217255. Springer Science + Business Media B.V*, 2007

[24] M. Eirinaki, M. Vazirgiannis, I. Varlamis. SEWeP: Using Site Semantics and a Taxonomy to enhance the Web Personalization Process. In *SIGKDD, pp. 99108.* 2003

[25] E. Gabrilovich, S. Markovitch. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *M.M. Veloso (ed.) Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007

[26] R. Mihalcea, A. Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the sixteenth ACM conference on Conference on Information and Knowledge Management, pp. 233242.*, 2007

[27] D. Billsus, M.J. Pazzani. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, 2000

[28] T. Gup. Technology and the End of Serendipity. *The Chronicle of Higher Education*, 1997

[29] L. Iaquinta, M. de Gemmis, P. Lops, G. Semeraro, M. Filannino and P. Molino. Introducing Serendipity in a Content-based Recommender System. In *Proceedings of the Eighth International Conference on Hybrid Intelligent Systems*, 2008