Page Rank in Recommendation Systems

Aravind Sankar (CS11B033)
Adit Krishnan (CS11B063)
Team 4

Table of Contents

1 Introduction

Our project is aimed at a study and implementation of conversational Recommender Systems. As opposed to conventional One-Shot Recommenders that aim to use historical accounts of user data, pre-encoded knowledge or collaborative means to provide a set of recommendations, the goal of conversation is to better understand the user's requirements through an iterative feedback based system.

In our system, we use the mechanism of preference based feedback where the user provides feedback in the form of accepting one product over the others. This feedback is then used to specialize the remaining items to better fit the user's needs. Note that similarity to user's chosen product may not be the only criterion here. There are many other issues to consider such as feature wise utility, serendipity and drift. However these are used as control factors over and above the primary process, which is to iteratively understand the user's needs.

In this project, we are specifically inspecting the effects of a wide variety of changes in an already existing recommendation framework which is based on application of Simrank and Pagerank to solve the conversational recommendation problem. We are particularly focused on 2 metrics here. The first one is the number of iterations required to converge to the chosen case. This is a metric that measures how fast the system converges to the required case. The second metric we are looking at is diversity. Diversity here will measure the spectrum of products that have been displayed to the user in the process of converging to the required product. This is also an important measure since we do not want to overfit to the user's initial requirements alone when displaying products. We wish to permit the user to change his mind (though our experimentation procedure does not consider this). With these goals in mind, we have built a hybrid conversational system borrowing ideas from multiple existing works as described in the following sections.

2 Literature Survey

Our initial literature survey was focused on ideas pertaining to collaborative filtering, knowledge/content based recommenders and other one shot recommendation strategies, illustrated in [1]. Itemrank [3] focuses on the idea of correlation through commonly rated items (in their case a movie dataset) and uses these as the weights to build a similarity matrix. A random walk is then performed on this matrix, to score items according to user preferences. This is a very basic collaborative filtering idea. [2], on the other hand, talks about similarity metrics using the MLT(More like this) idea to promote the users interests in product selection. In this system, a user expresses his preferences by selecting a product over a few other products shown during each interaction with the system. In the next interaction, the user is shown products most similar to the one he chose based on the Weighted Similarity model. As this model weights each feature equally, the results shown in the further interactions may not be relevant, if the user has given preference to some specific features over the others. To overcome this issue, WMLT was a technique introduced to weight attributes differently by looking at the deviation in the value of chosen product and the rejected products. [4] improves upon this by considering simrank based similarity measures and domain specific dominance relations among attributes (such as MIB and LIB). In particular, [4] uses simrank to estimate product and attribute value similarities by constructing a bipartite graph between products and their attribute values. They also dynamically estimate the product utilities by applying pagerank on a dominance graph. One significant drawback of [4] is that it does not include a diversity factor in it's recommended products. In our model, we focus on incorporating diversity into the recommendation process, in addition to improving the performance in terms of number of cycles using various changes in the way pagerank and simrank are used.

3 Basic Methodology

To start with we adopted ideas from some of the works mentioned in the Literature Review section. The most important ideas were those of application of Simrank and Pagerank in computation of utilities (idea proposed by [4]). While this was the primary theme, we made quite a few changes to the formulation and tried out a wide variety of significant changes to certain components and formulations in the algorithm. These are discussed in the later sections.

A brief overview of the idea in [4] is as follows. To start with, we have the database of products and their corresponding attribute values for each attribute. The set of attributes however, is fixed for all the products.

Since we are building a conversational system, it is expected that the objectives satisfied by the recommended products at each iteration depend on the user's inputs over the previous iterations of conversation. However, There is a need for a rigid mechanism of similarity evaluation that does not change over iterations of conversation. This is a set of facts that the system follows irrespective of other changes on top of how they are employed and where they are used. This mechanism must ensure invariance of similarities for a given dataset. Since we require not only product - product similarities but also attribute-value - attribute-value similarities in our computations, Simrank can be employed here. It permits simultaneous computation of both sets of similarities. Also since there are no dynamic parameters whatsoever, this can be precomputed hence entailing no computation at runtime.

Similarity, used by the system as described above is invariant over iterations and stays fixed for a given pair of products as well as a given pair of attribute values over the whole process. This acts as our basic similiarity model over the whole process. On the other hand the user's choices vary over iterations and must reflect in the computation of utility for products to show him in the next iteration. The utilities of a product for being shown in the k^{th} round of conversation depend on two things.

1. Basic/Invariant Similarity - Simrank Based

First, Similarity of attribute values of the product to those of the products it is being compared with.

Typically it will be compared to either the product chosen by the user in the $k-1^{th}$ round or other products that are deemed to be relevant by some other means, such as proximity to the chosen product.

2. Conversation Model

This the effect of propagated or induced effects. Propagated effects include things like weighted similarity, based on products chosen in previous rounds of conversation with decaying weights. Induced effects are based on applying a model to the process of conversation such as a preference dominance graph. This results in the induction of corresponding rules and parameters in the process of utility computation.

The following subsections detail both the parts involved.

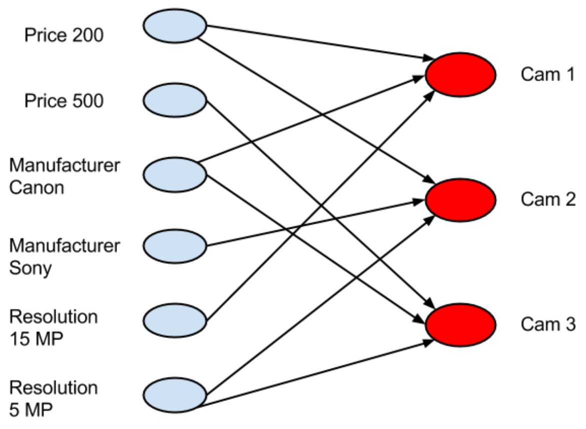
3.1 Application of Simrank - Basic/Underlying similarity

Simrank is a node proximity algorithm used to compute node similarities in a graph based on the idea of similarity propagation. If we consider a graph G, and look at tuples of nodes of G, then the source of similarity are the tuples of the form <a,a>, where a is a node of G, since the similarity of every node to itself is 1.

The flow of similarity is then recursively defined as follows: Nodes a and b of G are similar if the nodes that point to them (i.e.their in neighbours) are similar. The fundamental equation corresponding to this is:

$$\mathbf{S(A,B)} = \sum_{i=1}^{|I(A)|} \sum_{j=1}^{|I(B)|} sim(I_i(A), I_j(B))$$

Where I(A) and I(B) denote the in neighbours of A and B. However in case of a bipartite graph, like our product - attribute-value graph, we use the out neighbour set of nodes in A, the set of products, and the in neighbour set for nodes in B, the attribute value set, instead of using in neighbour sets for both. Below is an illustration of the product - attribute-value graph.

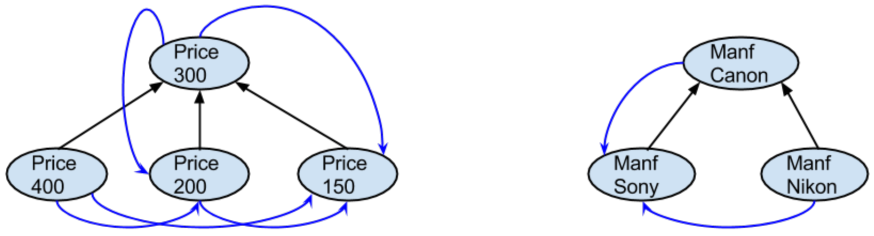


Once simrank is run on this bipartite graph, we obtain and store the similarities of every pair of products as well as every pair of attribute values of every attribute.

3.2 Graph based Conversation model

Now that we have the similarity model in place, we need to model the effect of conversations. This is performed by first modelling the result of the interaction in the form of a set of graphs. Each of these graphs is defined over the set of values of a particular attribute, that appear in the recommended top-K products.

There are two sets of edges added to these graphs. The first are called preference dominance edges, that point from all other values of that attribute to the value that appears in the product chosen by the user. The second set of edges are called local dominance edges. These are edges that point from a less desirable to a more desirable value of the attribute. i.e. if values a_1 and a_2 appear in the graph, there is an edge from a_1 to a_2 if a_2 is the more desirable value (could be more or less than a_1). The choice of more and less desirable attribute values are based on static MIB (more is better) or LIB (less is better) assigned to each of the attributes. If an attribute is MIB, a larger value is considered dominant over a smaller value of the attribute attribute and in the opposite case (LIB) a smaller value locally dominates a larger value. This is illustrated in the sample graphs below.



Black edges represent preference dominance, while blue edges indicate local dominance.

Thus in a given conversation round, we obtain these graphs for every attribute. The next step is to infer the relative importances of attribute values for each attribute. The relative importances must be based on the structure and edges of the graph. The attributes that have incoming edges of high weight should be considered more important. Pagerank algorithm captures all of the required aspects here. The weights of the edges can be normalized to provide the transition probabilities.

3.3 Utility computation

When a round of conversation is complete, we obtain the set of Pagerank importances P(a_{att}) for every attribute value a_{att} of every attribute att. We also have the chosen product of this round, say P_c.

Now for every product in the database:

$$\text{Utility(P)} = \sum_{att \in ATT} \text{pr(att(P)) x simrank(att(P),att(}P_c\text{)}$$

This clearly, the two factors receiving importance here are the Pagerank score (pr) of the attribute value, which depends on the user's choice and the edge weights of the attribute graph, and second the simrank similarity of the attribute values of P and the chosen product (P_c). This is summed up over the set of all attributes(ATT).

The top K highest utility products form the next recommended set of K items and the process repeats.

4 Extensions

4.1 Preference Dominance

Preference dominance edges of the attribute graphs are edges that begin at attribute values other than that of the chosen product and end at that attribute value. They are called preference dominance edges since they capture the prefernce of the chosen product over the other recommended products.

What could be a good value to assign to the preference dominance? (Value here referring to the weight of the preference dominance edges). Note that it is an indication of how strong the preference was for that specific product among the remaining recommended ones.

The metric that we used was as follows - First we computed the average similarity over all pairs of cases in the recommended K products, Let us call this μ_S.

Then 1-μ_S is one simple metric that can be used. This is a measure of the overall dissimilarity within the K products. Higher the dissimilarity, by the WMLT idea, more significant the user choice becomes.

A secondary idea here is to change the value of preference dominance attribute wise. For each attribute, we find the average deviation for the chosen product from the same attribute for the rest of the products. If this is high, it indicates that this is a strong choice since it has been chosen among lots of other choices that are diverse from it.

This is given by $\sum\limits_{p \in topK} 1 - sim(attr(chosen), attr(p))/(K-1)$ (Note that 1-sim denotes deviation)

This can be normalized by computing this over all attributes and dividing by the sum.

Pref.Dominance$_{att}$ =

$$(\sum_{p \in topK} 1 - sim(att(chosen), att(p)) / \sum_{att \in A} \sum_{p \in topK} 1 - sim(att(chosen), att(p)))$$

4.2 Local Dominance

In all the works mentioned thus far, it is assumed that the direction of edges incase of Local Dominance is determined based on static MIB/LIB values. This means that every attribute is statically assigned a more is better/ less is better behaviour. There are 2 problems with this.

- Users tend to prefer mid range rather than extremes. Thus the local dominance works against them when they try to move towards the center.
- A static assignment always attaches a constant weight with the local dominance edge. However we want to might want to see the degree of dominance. If the degree of dominance is low, a lower edge weight should be added.

We came up with the following solutions to the above problems:

a) To avoid the problem of static behaviour determination, we use the notion of membership to MIB and LIB which is dynamically computed at each round of conversation.

   MF(att) (more is better factor) $\propto$ number of products among the recommended set with less of the attribute att than the chosen product, extent to which they are less than the chosen product

   LF(att) (less is better factor) $\propto$ number of products among the recommended set with more of the attribute att than the chosen product, extent to which they are more than the chosen product

   The exact computations of these memberships are illustrated in [5].

b) Since the user's choice reflects whether he wishes to reduce a particular attribute or move higher, and to what degree, thus the edge weight is set accordingly for local dominance edges when carrying out Pagerank over the values of that attribute.

4.3 Utility Computation with wMLT

The normal utility score computation illustrated in section 3.3 considers all attribute picks to be equally important. It adds them all up to get the overall utility of that product. However, the utility contribution of attributes that most certainly are of the user's preference should be the highest since that would lead to the best products for the user, having the highest utility.

To include this, we weight each attribute with the number of unique values under that attribute which were present in the K recommendations of this round. This helps us ensure that the more discriminating attributes (i.e.those for which the choice available was more) are weighed better. The score then changes to:

$$\text{Utility(P)} = \sum_{att \in ATT} \text{pr(att(P)) x simrank(att(P),att}(P_c)\text{ x wMLT-score(att)}$$

4.4 Personalized Pagerank

Personalized Pagerank changes the random teleport vector in the attribute graphs to a deterministic vector which take all restarts back to a single node or a small fixed set of nodes that we can choose. This helps prevent too much drift, Keeps most of the importance uniformly distributed around the central points and is more predictable and consistent. This is particularly useful when attributes follow a normal distribution., i.e. A large number of users prefer low deviation from the initial set of values and wish to explore its neighbourhood.

Main Problem Here - The convergence of Personalized Pagerank is much slower than the random teleport version. Due to this limitation we have not extensively experimented with Personalized Pagerank. Depending on accuracy constraints, It may be infeasible when considering online transactions.

4.5 Recommended Product Diversification

This is an additional goal that does not enhance fast convergence necessarily. The goal of product diversification is to show the user as wide and diverse a set of products as possible at each round of conversation. This boils down to just one simple modification in the computation of the utility score of the onject.

Utility(P) = α x $\sum_{att \in ATT}$ pr(att(P)) x simrank(att(P),att(P_c) x wMLT-score(att) + (1 - *alpha*) x diversity-score

This is a simple weighted sum formulation. Diversity here was set to two different things. The first is how diverse the K recommended items are from each other, i.e.mutual diversity, and the second is how diverse the new case is from the history of chosen cases over the completed rounds of iteration. A combination of these two things could also be used.

The same is also applied to the starting product set. The user provides a set of attributes as his query. The starting set of recommendations is then built as follows:

While size < K, insert the node which maximizes $\alpha \times$ weighted-similarity to query + (1 - α) $\times$ mutual diversity of this node with others in the set

Again note that this exercise was to try and improve the quality of recommendations, It does not help the system converge faster in our case, but in practice should be an important objective as well.

5 Experiments and Results

5.1 Dataset Used

The dataset used by us was the Camera Dataset available at `http://josquin.cs.depaul.edu/~rburke/research/downloads/camera.zip`. It has a total of 210 different Cameras, each having 10 attributes (4 nominal and 6 numerical attributes).

5.2 Experiments

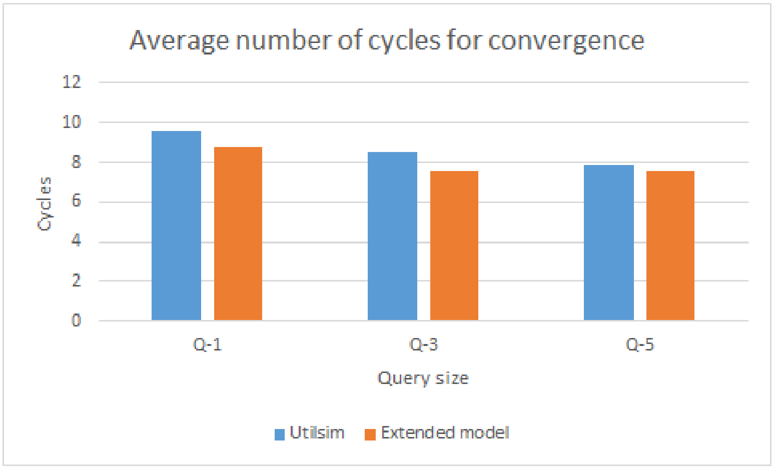
For experimentation it is necessary to have an agent which simulates a user. In our case this agent uses a very rudimentary weighted similarity model with equal weights for all attributes. A case is removed from the case base and assigned as the agent's target. The agent then randomly chooses as many attributes from this case as instructed and provides these values to the recommender. Then over each round of iteration it uses the weighted similarity score to make a choice among the K products recommended, with preference noise added.

Note that our user agent here is static. Its preferences do not change over time.

It is a rigid and poor approximation of user behaviour, but it serves the purpose to give us an idea of the convergence behaviour of the system.

Our convergence goal is the most similar product, to the product removed from the database when forming the query. We will stop when this most simiar product is present among the K recommended products in a round of conversation.

The results are somewhat sensitive to the choice of parameters in our equations, most importantly the value of the diversity factor α. For the following set of results α was fixed at 0.1. We compare the number of rounds of conversations for convergence of the system, in absence of the extensions and in their presence. The number of products shown per iteration is 4 (K=4).



Note that query size here denotes the number of attributes of the chosen product used to form the query.

On an average, we notice around 10% improvement in convergence. We can make it slightly faster if we remove the diversity metric, however considering the quality of recommendations in each round of conversation, it is important to maintain.

For measuring the effect of our diversity metric, we looked at the diversity within the recommended products in rounds of conversation. The average diversity metric that we defined was 1 - avg. sim recommended products.

Where, avg. sim $= \sum_{i=1}^{K} \sum_{j=1}^{K} \text{sim}(P_i, P_j)$,
sim used here being the simrank similarity of the two products.

When we used α factor as 0.1 instead of 0, there was a 2% decrease in the average similarity of products lying within the recommendations over all iterations. While this is small in absolute terms, relative to the earlier average similarity of about 30%, this works out to about 8% reduction.

References

1. Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." Knowledge and Data Engineering, IEEE Transactions on 17.6 (2005): 734-749.
2. Mc Ginty, Lorraine, and Barry Smyth. "Comparison-based recommendation." Advances in Case-Based Reasoning. Springer Berlin Heidelberg, 2002. 575-589.
3. Gori, Marco, Augusto Pucci, and V. Roma. "ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines." IJCAI. Vol. 7. 2007.
4. Gupta, Saurabh, and Sutanu Chakraborti. "Utilsim: Iteratively helping users discover their preferences." E-Commerce and Web Technologies. Springer Berlin Heidelberg, 2013. 113-124.
5. Gupta, Saurabh, and Sutanu Chakraborti. "Flexible and dynamic compromises for effective recommendations." Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013.