Paper AA05-2015

Need Help Finding Something? Why Not Ask Your (Virtual) Community! A Collaborative Filtering Based Recommendation Engine in SAS®

Ben Elbert, Alliance Data, Columbus, OH

ABSTRACT

People have likely been recommending everything from where to eat to whom to meet since the dawn of man, but in the last twenty years we have seen the rise of automated, data driven, recommendation engines (also known as recommender systems, expert systems, etc.). One popular approach to recommending products to potential shoppers is collaborative filtering (CF), which relies on weighting the quantity, propensity, or rating of other shoppers by the similarity of the potential shopper's transactions to those of the community of shoppers (traditionally at an individual level). This traditional implementation of a CF-System has the drawback that new or early tenure shoppers may not have enough informative transactional data to make valuable recommendations. In this paper we present a model that, when used with information available to banks and creditors such as geographic, demographic, and credit bureau data, overcomes the cold-start problem and make valuable recommendations to new or low tenure cardholders in addition to multi-transaction persons. We also demonstrate how to implement a model that leverages large amounts of data in SAS® and optimizes run time by providing tips for testing the model to determine optimal parameters. Finally, we discuss how to extend our CF-System to areas of interest beyond the recommendation challenge.

INTRODUCTION

Collaborative Filtering is a popular recommendation engine technique that is, in fact, only about as old as most college graduates these days. In their 1992 paper [1] introducing the automated librarian system, Tapestry, Goldberg et al. birthed collaborative filtering (as an algorithm) and employed it to assist persons with subscribing to meaningful email lists with ease. We make the point to clarify that this is the first algorithmic instance of collaborative filtering because the idea of relying on trusted advisors has existed for, likely, as long as humans have. Collaborative filtering is an appealing algorithm for many reasons. First, it is intuitive: would you rather listen to a group of close friends with similar interests or to ten random persons? Additionally, the algorithm itself focuses on the *grouping* of persons, not any particular recommendations or use, thus the groups can be used for many purposes like Test/Control selection, Behavioral Analysis, and other instances where look-alike modeling is useful. Finally, it is used by companies like Amazon [5] for item-to-item recommendations, Last.fm [8] for recommending new music to users based on their 'scrobbled' songs, and StumbleUpon [6] to provide a social news feed to users–a large number of successful information-based companies always makes a new technique much easier to sell up the chain of command.

In retail and credit marketing, one of the most sought after successes is to *know your customer*, and well placed, well chosen, well timed product recommendations rank near to Holy Grail status along with geo-fenced offers and just-in-time basket growing offers. For most retailers, customer information requires either multiple transactions and additional voluntary steps or comes at a steep cost through purchasing files and implementing matching algorithms to incorporate said data. Creditors have one advantage in the data acquisition arms race, namely a large volume of data assets made available during the application process that allows for new cardholders to be marketed to quickly by leveraging the knowledge gained from longer tenure

accounts. Because collaborative filtering requires intense matrix and vector operations, both during development and implementation, SAS® is not always thought of as a go-to tool for CF-systems, but in this paper we will provide the theoretical background, helpful tips, and SAS® code and macros to make a collaborative filter based recommendation system a reality for any motivated SAS® user.

Collaborative Filtering

The two guiding principles of Collaborative Filtering are the Law of Large Numbers (LLN) and the Wisdom of Crowds (WofC) [6]. We find that collaborative filtering is stable and useful because the LLN guarantees that our recommendations are close approximations of the *average* of a like shopper as our virtual community grows. Moreover, a *stringent* CF-system that takes only the recommendations from the N most similar persons mimics a WofC principle, the Delphi Method, to rely on the combined expertise (hence the stringency) of the informed (via the mimicking of appealing fashion) panel (the community).

More formally, let us define and examine the CF-system model to better understand the operations that need to be carried out in order to implement. Suppose we have a set of shoppers, S, with cardinality N, and a set of items that can be purchased, K with cardinality k. Additionally, assume we have a new shopper, $\alpha \notin S$, and have a relation for computing the similarity between two shoppers i and j, s_{ij}, such that s_{ij} is at least a semi-metric. Then, for the $n \leq N$ most similar shoppers to α and their corresponding matrix of quantity of items purchased, $\hat{K} \subseteq K$ with entries q_{ij} corresponding to the quantity of item j purchased by shopper i, we have the following matrix equation for deriving the vector recommended product quantities, V, to α:

$$V_{1xk} = \Delta_{1xn}\hat{K}_{nxk} = \begin{pmatrix} s_{1\alpha} & s_{2\alpha} & \dots & s_{n\alpha} \end{pmatrix} * \begin{pmatrix} q_{11} & \dots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{n1} & \dots & q_{nk} \end{pmatrix} \tag{1}$$

where Δ is the length n vector of similarities between the n shoppers and α. Typically one chooses a similarity function that normalizes to the $[0, 1]$ interval such as Pearson's R^2 or Cosine Similarity, but the algorithm provided in this paper uses the Gower Similarity Coefficient [2] which is also normalized to the $[0, 1]$ interval but is specifically designed to handle heterogeneous variable types efficiently. Thus, implementing collaborative filtering can be laid out in three steps:

1. Determine the universe of persons or objects that will form the virtual community and the items that they will recommend by having purchased/interacted with before
2. Apply a similarity function to the cross-product of the virtual community (further apply thresholding if *experting* is desired)
3. Derive the item recommendations by multiplying the distance matrix resulting from Step 2 with the person-item matrix from Step 1.

Before providing code and analytic results from improvements to the general collaborative filter algorithm, let us introduce the Gower Similarity Coefficient, which can be calculated in SAS® using PROC DISTANCE.

Gower's Similarity Coefficient

In 1971, J.C. Gower introduced a similarity coefficient that could handle heterogeneous variable types easily [2]. In the event that there are no missing values in the similarity matrix, Gower shows that the resulting similarity matrix is positive semi-definite, which ensures that the square root of the dissimilarity (i.e. $(1 - s_{ij})^{\frac{1}{2}}$) is a Euclidean measure. For this reason, we recommend omitting or imputing missing features to provide for this convenient result.

In his paper, Gower describes three classes of variables that his function operates on, and we provide them here with example:

1. *Dichotomous or Binary Variables* – Variables that have a 0 or 1 state indicating the presence or absence of a given feature (e.g. The variable *Is_Female* which is a 1 if the shopper is female and 0 otherwise)
2. *Qualitative/Categorical Variables* – Variables that have many, discrete possible values but do not have a transform mapping to an ordinal scale (e.g. The variable *Home_Style* which takes on values such as Ranch, Condo, Apartment, Multi-Story, Studio, etc.)
3. *Quantitative Variables* – Ordinal, Continuous, or Ratio variables that imply a scale and range of values (e.g. The variable *Sales_Last_Year* that provides the total sales in dollars of a shopper last year; The variable *Sales_Rank* that gives the rank (0-9,1-5,etc.) of the sales total such that ordering is possible and preserved; The variable *Sales_to_Max_Ratio* that provides the ratio of the individual shopper's sales to the maximum sales seen in the period)

As one can see, Gower designed his similarity coefficient to be able to handle multiple variable types. It is because of this that we use the coefficient so as to allow for adaptive incorporation of future data without extensive normalization, hygiene, and additional variable mapping.

The Gower Coefficient, s_{ij}, measuring the similarity between two entities, i and j, each with k features is calculated in the following way:

$$s_{ij} = \frac{\sum_{\hat{k}=1}^{k} s_{ij\hat{k}} w_{\hat{k}} \delta_{ij\hat{k}}}{\sum_{\hat{k}=1}^{k} w_{\hat{k}} \delta_{ij\hat{k}}}$$

where

- w_k is a weight applied to the similarity at feature k;
- δ_{ijk} is a signal variable that is 1 if either i_k or j_k are present and 0 if both are missing; and,
- s_{ijk} is 1 if feature k is of variable Type 1 or Type 2 and $i_k = j_k$, 0 if feature k is of variable Type 1 or Type 2 and $i_k \neq j_k$, and equal to $1 - (|i_k - j_k|/R_k)$ if feature k is a Type 3 variable where R_k is the range (or measure) of feature k determined either by the sample or the universal population.

Thus, s_{ij} is determined as a weighted (allowing $w_k := 1$ for all k, possibly) average ranging between 0 (complete disagreement) and 1 (complete agreement). Before moving into the more applied sections where we will discuss ways to optimize the implementation of a CF-System, provide analytic results, and discuss code, let us provide a brief example to help the reader see how the algorithm works from start to finish.

Example: *Suppose we have three shoppers, $< p^1, p^2, p^3 >$, with a system of features given by (with row i corresponding to p^i)*

$$\begin{pmatrix} 1 & 3.4 & F & Ohio & 12.1 \\ 1 & 4.1 & M & Utah & 8.4 \\ 0 & 3.9 & M & Texas & 8.6 \end{pmatrix},$$

and further suppose the three shoppers have the following quantity purchased matrix for five items given by

$$\begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 3 & 7 & 5 & 3 & 0 \\ 4 & 1 & 3 & 3 & 5 \end{pmatrix}.$$

Now, suppose our new shopper, α, has a feature encoding of $(0, 4.0, F, Texas, 10.4)$. What are the top two products that should be recommended to our new shopper?

Solution Walkthrough: With limited time-series transactional data to construct a correlation-based product affinity model (which we will discuss later why our CF-system model is more appropriate for product recommendations than affinity models) and the new shopper, α, not being distinctly enough like any one of the three shoppers considered to apply descriptive profiling, we elect to measure the similarity between α and our three shoppers with the unweighted Gower Similarity. Examining the limited sample of data, we decide that Features 1 and 3 are Type 1 (binary) variables, Features 2 and 5 are Type 3 (Quantitative), and Feature 4 is a Type 2 variable. Because we do not know the ranges for Features 2 and 5, we will make the assumption that the interval extends from 0 to the ceiling of the maximum observed value. Additionally, as we do not even know what each feature corresponds to (even though we can infer at least two definitions), we are unable to create weights to adjust the similarity score based upon feature importance.

If we carry out the calculations for the Gower Similarity Coefficient with unitary weights, we obtain the following similarity vector:

$$\Delta = (0.5498, 0.3652, 0.7668).$$

Substituting our Δ vector and the item quantity matrix belonging to the three shoppers into Equation 1, we see that the recommended product 'quantities' for our new shopper α are:

$$V = (4.163, 3.873, 5.226, 3.396, 3.834).$$

We place the term quantities in quotes above because these are not in fact the recommended quantities, but rather the votes submitted through this recommendation system. The benefit to using this approach instead of a binary PURCHASED flag is that we fully account for the popularity of an item based upon how frequently it is purchased. For this reason, after ranking the votes, we would say that our top recommendation for shopper α is Product 3 and our second recommendation would be Product 1. One interesting point about this type of recommendation system is the way that everyone contributes versus looking for maximal points. If we would

have just used the maximums for the most similar shopper, p^3, we would have recommended Product 5 first and then Product 1–the use of singular rows reduces the information gain from the others that resulted in Product 3 being the top choice.

Additionally, if we had further business rules such as sales goals or profit margins, we could add post filtering to our recommendations to recommend the products that would be most appealing given the criteria. In the next section we demonstrate how to compute the Gower Similarity Coefficient in SAS® using PROC DISTANCE, and then we will move on to optimization strategies and implementation.

Calculating Gower's Similarity Coefficient in SAS® with PROC DISTANCE

PROC DISTANCE is a useful SAS® procedure provided in SAS/STAT that easily allows for multiple types of comparisons (distance, dissimilarity, and similarity) depending on the data types present in the comparative measure. Contrary to the procedure name, when we set the METHOD parameter in the procedure statement to Gower, we're calculating the *similarity* and thus the closer the similarity is to 1 the more alike the two elements are. One important callout about the DISTANCE procedure is that it requires the ID variable provided in the ID statement to be character, so one will typically need to create a new variable with a TRIM(LEFT()) operation, or some other numeric to character conversion process.
Below we demonstrate the calculation of the Gower similarity coefficient in SAS® with PROC DISTANCE, where we assume that we have a dataset named ID_FEATURES which has a character ID variable named *char_id*, a Type 1 Variable named *sex_mf*, a Type 2 Variable named *shopping_chan_pref*, and a Type 3 Variable named *sales_LY*. Additionally, because we desire the full matrix rather than the lower, triangular half, we set the SHAPE option to Square.

```
PROC DISTANCE data=ID_FEATURES method=GOWER
shape=SQUARE out=ID_FEATURES_SimMatrix;
id CHAR_ID;
var anominal (SEX_MF SHOPPING_CHAN_PREF);
var interval (SALES_LY);
run;
```

As PROC DISTANCE creates a complete (dis)similarity matrix and does not simply add a new variable to each observation, it is important to provide a new dataset name in the OUT statement rather than providing the input dataset (which would result in that dataset being overwritten). The most difficult aspects of using PROC DISTANCE for the Gower similarity (and really when calulating similarity with any metric) is the task of separating variables into their *types* and the task of assigning weights (if desired).

In the appendix we provide sample code for a macro that is embedded in our primary macro which allows for users to have their variables automatically split into three separate types and have them stored as space-delimited lists in macro variables for ease of use, but determining variable weight is not as easy. In the next section we will discuss ways to determine variable weight, and then, in the section following, we will demonstrate how to apply the Gower Similarity to a recommendation engine using SAS/IML® (a brief overview of SAS/IML® will be provided as well).

Weighting Schemes for Use in the Gower Similarity Coefficient Formula

Determining feature weights is a difficult problem and can actually be an NP-hard problem if one desires to find the optimal point in the search space of all possible weights, we recommend a few approaches here that have been used before in our research. The following suggestions should not be regarded as hardfast rules, nor should one expect them to always work in every scenario. We provide the ideas, a few sources to read for more information, and also considerations for why the approach may not be appropriate in some cases, but we do not provide details or specifics about our previous research findings in this area.

The first method for determining feature weights in the Gower equation is to leverage **regular regression analysis** to predict a separation variable such as difference in annual spend, difference in annual trips, or another behaviorally descriptive variable. Once the regression coefficients have been determined, we recommend first splitting the variable set into two classes based upon whether or not the Variance Inflation Factor (VIF) is greater than or less than a given threshold, Θ (typically 4 to 5). The variables with $VIF \geq \Theta$ should have their weight set to 0 in the similarity equation, while the others should be given a weight equal to their ranking (in descending order) based upon their corresponding regression coefficient. The regression approach does have the drawback that any Type 2 variables need to be coded properly for the regression analysis, in addition to the fact that the analysis needs to be repeated multiple times with a Leave One Out approach in order to account for the separation variable.

A second approach that also relies on the Leave One Out principle is to repeatedly, randomly draw an individual from a universe of active persons (meaning they have made at least one, but preferably multiple purchases during a time window), correlate the LOO individual's item quantity matrix and to each other person's item quantity matrix and flag as a 0/1 if the correlation is higher than a given threshold. From here, **logistic regression** can be utilized to model to the binary variable, and the same process as before of ranking coefficients and converting those ranks to weights can be implemented.

The next two approaches are based on applying biological mechanisms (digitally, of course) to solve problems. Neither attempt is guaranteed to converge to the global optimum, and can fall victim to local maxima that do not correspond to the most optimal solution–one method for attempting to overcome this limitation is to scatter starting points far from each other and allow for longer runs to see where they converge. This is still not a guaranteed approach, but it requires less start up work once an algorithm is coded. The first algorithm is the **parallel genetic algorithm**, which Li et al. describe in great detail and provide code for in their paper from the 2014 MWSUG [4], is a process of choosing a starting population (a vector of feature weights or signals) and allowing the fittest (and possibly a small number of randomly selected sub-fit individuals) to crossbreed (with mutation) and form new individuals. This process is repeated until a convergence criteria is established. The second algorithm is **particle swarm optimization** which can be implemented in many ways (ant colony optimization, pollination optimization, etc.) but always entails the same principles that swarm insects apply to food gathering. By allowing for thousands or tens of thousands of randomly started entities to wander the search space and allowing the gain or loss of ground towards a better solution to be the food, and by allowing the entities to communicate by pulling the vectors of nearby entities toward more desirable solutions, we mimic nature's solution to finding food and sustaining insect communities.

These weighting methods are by no means the only ones, nor are these likely to be the best in all scenarios. What we have tried to do here is present two sets of methods which will re-

sound and be easily explained to senior leadership if they are implemented. Our next section describes the details of implementing a Collaborative Filtering based Recommendation Engine in SAS/IML.

Implementing a Collaborative Filtering Based Recommendation Engine in SAS/IML

Because a CF-System is an implementation of a matrix-based algorithm, we have elected to build our recommendation engine with SAS/IML. SAS/IML® (**I**nteractive **M**atrix **L**anguage) is a powerful matrix engine in the SAS® toolkit that enables users to convert complex mathematical formulas into SAS® code, run sophisticated numerical algorithms and simulations, and execute R code (provided the proper R version is installed). One hurdle for the SAS® programmer that is looking to extend their skill set into IML, is the difference in language syntax and semantics.

The SAS/IML® language differs from the traditional SAS® language in that it is closer to an Object Oriented Language (OOL) where objects must be opened, closed, and handled properly and explicitly. For instance, to convert a SAS® dataset, *DATA*, to an IML matrix object, *MATOBJ*, one would do the following:

```
proc iml;
use work.data;
read all into MatObj[colname=varNames];
close work.data;
quit;
```

The final statement, **CLOSE**, must be included since we are actually opening the dataset for read-access and must close it to remove the temporary lock. For an in-depth overview of the IML language and the many uses of it, please refer to Rick Wicklin's book [7], his incredibly helpful blog (The DO Loop), or the SAS/IML® User's Guide [3].

Next we provide two macros, **Make_Gower_Vars** and **CF_System**. The first is a handy macro to have in general because it creates a meta-dataset for the dataset you choose and it creates macro variables that list Type I, II, and III variables. This becomes useful in modeling, summarizing, and selecting data.

The second is the CF-System, but it does not have any weighting functionality presently due to the level of specificity and customization required for weighting variables. Still, this current CF-System is powerful and useful, though threshold tuning may still be needed on a case by case basis. Before providing the macros, let us show the code for invoking the CF-system and discuss the macro variables.

```
%cf_system(
universe = All_Persons,
items = Items_Purchased,
id_var = ID,
excl_vars = ('ID','NEW'),
comp_indicator = New,
library = WORK ,
theta = 0.6 ,
out_dsn = Product_Recs
```

```
);
```

Looking at the variables, we have: **Universe**, which is the name of the dataset of your people and their attributes; **Items**, which is the dataset of products purchased (note, this must include the ID variable, and have one row for each person and the quantities of items purchased as columns); **ID_Var**, which is the unique identifier variable; **Excl_Var**, which is a list of variables to be excluded from the Gower formula (note, this list must be in all-caps, single quoted, comma separated, and wrapped in parentheses due to the SQL code used); **Comp_Indicator**, which is the name of the 0/1 variable used to separate the persons recommending from those receiving (note, one can set all persons to 1 and then everyone with more than Θ similarity will recommend for everyone else); **Library**, which is the name of the library where the datasets given in **Universe** and **Items** are stored; **Theta**, which is the threshold for filtering out dissimilar persons (higher values imply more strict filtering, and 0 imposes no filtering at all); and, finally, **Out_DSN**, which is the name you want to give to the final product recommendation dataset.

A key thing to note is that the returned dataset, named in the **Out_DSN** variable, will have all of the original fields from the **Universe** dataset, but also will include the raw votes for each product and the rankings (1 implies most recommended, 2 is next most, etc.). Thus, if you had 8 matching attributes and 12 products you would have a final dataset of 34 (2 non-matching, 8 matching, 12 product vote variables, and 12 product rank variables).

In the final section below contains the macros that can be used to implement a CF-System in SAS, but the penultimate section discusses some additional uses for CF-Systems beyond the recommendation challenge.

Extensions of the CF-System

In marketing, behavioral analytics, and research, we are commonly faced with scenarios where we must compare the behavior of Group A, a known participant group, with the behavior of some other group, Group B, that may or may not be identified. When Group B has not been identified, it is possible to use the CF-System to identify suitable persons to match Group A. Moreover, by applying additional weight to either the characteristic or behavioral variables, one can obtain groups that should either be expected to behave alike (behavioral weighting), or that may provide key predictive and profiling power for the future (characteristic weighting).

Macros

```
/* Macro to determine the variables for the Gower similarity score */
%macro make_gower_vars(dsn, excl_vars,library=WORK);
%symdel type1_vars type2_vars type3_vars;
ods output nlevels=nlevelsds;
proc freq data=&library..&dsn. nlevels;
tables _all_/noprint;
run;

%global type1_vars type2_vars type3_vars;

proc sql noprint;
create table meta as
```

```
select name,type,nlevels
from dictionary.columns,nlevelsds
where libname=upcase("&library")and memname=upcase("&dsn")and name=tablevar;
/* Select Binary Variables (Type 1 Variables) */
select name into :type1_vars separated by ' '
from meta
where nlevels = 2
and upper(name) not in &excl_vars.;

/* Select Characteristic Variables (Type 2 Variables) */
select name into :type2_vars separated by ' '
from meta
where nlevels > 2 and type NE 'num'
and upper(name) not in &excl_vars.;

/* Select Interval/Ordinal Variables (Type 3 Variables) */
select name into :type3_vars separated by ' '
from meta
where nlevels > 2 and type EQ 'num'
and upper(name) not in &excl_vars.;
quit;
%mend;

%macro cf_system(
universe /* Single Level DSN of all persons in Universe */ ,
items /* Single Level DSN of Items Purchased by persons in Universe */ ,
id_var /* Variable (Character or Numeric) which is a unique record identifier */ ,
excl_vars /* Must be comma separated, quoted, list wrapped in parentheses */ ,
comp_indicator /* 0/1 Var...0's Recommend to 1's */ ,
library = WORK /* Library where datasets are stored */ ,
theta = 0.6 /* Threshold Variable to create trusted advisor groups */,
out_dsn = Product_Recs /* DSN for final product recommendation votes and ranks */
);

%make_gower_vars(&universe.,%str(&excl_vars.));

%put &type1_vars;
%put &type2_vars;
%put &type3_vars;

data &library..&universe.;
set &library..&universe.;
char_id = trim(left(&id_var.));
run;

proc sort data=&library..&universe.; by char_id; run;
proc sort data=&items.; by &id_var.; run;

proc distance data=&library..&universe. method=GOWER
        absent=0 shape=SQUARE out=&library..&universe._sim_matrix;
```

```
id char_id;
var anominal (&type1_vars. &type2_vars.);
var interval (&type3_vars.);
run;

proc sql;
create table &library..&universe._sim_matrix as
select b.&comp_indicator., b.&id_var.,
                        a.*
from &library..&universe._sim_matrix a
inner join
          &library..&universe. b
on a.char_id=b.char_id;
quit;

proc iml;
use &library..&universe._sim_matrix;
read all var _num_ into s;

use_ind = s[,1];

rows = nrow(s);
cols = ncol(s);
do i=1 to rows;
        do j=3 to cols;
                if (j-2=i | s[i,j] < &theta. | use_ind[j-2,1]=1) then s[i,j]=0;
        end;
end;

create work.sim_matrix_thresh from s;
append from s;
close work.iml_test;
close &library..&universe._sim_matrix;
close;

use work.sim_matrix_thresh;
read all var _all_ into Delta_mxn where(COL1=1);;
close work.sim_matrix_thresh;

Delta_mxn = remove(Delta_mxn,{1 2});

use &library..&items.;
read all into K_nxk[colname=varNames];
close &library..&items.;

cols = ncol(K_nxk);
rows = nrow(K_nxk);
K = K_nxk[,2:cols];
```

```
V_mxk = Delta_mxn * K;

cols = ncol(V_mxk);
rows = nrow(V_mxk);
r = j(rows,cols,0);

do i=1 to rows;
        r[i,] = rank(V_mxk[i,]);
        do j=1 to cols;
                r[i,j] = (cols+1)-r[i,j];
        end;
end;

rRows = nrow(r);
rCols = ncol(r);

print r;
print rRows;
print rCols;

call symputx("NumItems", ncol(V_mxk));

end_rn = "Item_Raw_Votes_" + strip(char(ncol(V_mxk),4));
raw_names = "Item_Raw_Votes_1":end_rn;

end_rnkn = "Item_Ranks_" + strip(char(ncol(V_mxk),4));
rank_names = "Item_Ranks_1":end_rnkn;

create work.prod_recs from V_mxk [colname=raw_names];
append from V_mxk;
close work.prod_recs;

create work.prod_recs_ranks from r [colname=rank_names];
append from r;
close work.prod_recs_ranks;

close;
quit;

%put &numItems;

data Prod_Recs;
length n 4.;
set Prod_Recs;
n = _N_;
run;

data Prod_Recs_Ranks;
length n 4.;
set Prod_Recs_Ranks;
```

```
n = _N_;
run;

data &universe._recs;
set &universe.;
if &comp_indicator.=1;
run;

data &universe._recs;
length n 4.;
set &universe._recs;
n = _N_;
run;

proc sql;
create table &out_dsn. as
select a.*, b.*, c.*
from &universe._recs a
inner join
          Prod_Recs b
on a.n=b.n
inner join
          Prod_Recs_Ranks c
on a.n=c.n
order by a.&id_var.;
quit;

%mend;
```

References

[1] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[2] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):pp. 857–871, 1971.

[3] The SAS Institute. Sas/iml 9.2 user's guide. *SAS User Guides*, 2008.

[4] Jinqiao Li, Yi Cao, and Yanping Shen. A sas macro using parallel genetic algorithm to automate variable selection. *Proceedings of the 2014 MWSUG*, 2014.

[5] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations (item-to-item collaborative filtering). *IEEE INTERNET COMPUTING*, 7(1):76–80, January 2003.

[6] Muhammad Saleem. Collaborative filtering: Lifeblood of the social web, 2011.

[7] Rick Wicklin. *Statistical Programming with SAS/IML Software*. SAS Publishing, 2010.

[8] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation - A probabilistic generative model approach. *CoRR*, abs/1109.0758, 2011.