

Recommending Functions in Spreadsheets from the Fuse Corpus

Shaown Sarker, Emerson Murphy-Hill
Department of Computer Science
NC State University
Raleigh, North Carolina
Email: {ssarker, emurph3}@ncsu.edu

Abstract—The most common form of end-user programming software is spreadsheets. Despite spreadsheets having a large array of functions built-in, most of these functions are underused. To address this, we investigate recommender system technology and present a collaborative filtering based function recommender system for spreadsheets. Our evaluation shows that the recommender system performs significantly better in comparison with a most popular algorithm based system. In this paper, we discuss the implementation of the system and the evaluation process, and finally discuss the results of the evaluation.

I. INTRODUCTION

By definition, end-user programmers are people who are not professional software developers, but makes use of tools and processes that lets them perform tasks similar to programming [1]. Spreadsheets are the most common form of end-user programming software. According to a study in 2005 [2], nearly 23 million Americans use spreadsheets, constituting 30% of the workforce. The use of spreadsheets is also very common in industry for analytical purposes. Winston [3] estimates that around 90% of all analysts in industry perform calculations in spreadsheets. From a spreadsheet user's perspective, formulas can be viewed as program fragments as formulas can contain programming constructs such as variables, conditional statements, and references to other parts of the spreadsheet. It is important that these end-user programmers using spreadsheet makes efficient use of the functionality available to them [Citation needed?].

In Microsoft Excel, there are almost 350 unique functions, (472 in Microsoft Excel 2013 including the compatibility functions¹). But, most of these are rarely used [Citation needed?]. To facilitate these very large number of spreadsheet analysts to use these functions, it is viable that we explore effective function recommendation in spreadsheets.

Researchers have been assembling and preserving spreadsheet corpora for the purpose of better understanding of end-user activities and designing tools to assist them [4], [5]. In this paper, we look into Fuse [6], the largest reproducible spreadsheet corpus known to date. We applied a slightly modified version of the collaborative filtering algorithm on the spreadsheets in Fuse to get function recommendations for an input spreadsheet. We evaluated the performance of our system

against another commonly used recommender algorithm and found out that our collaborative filtering-based recommendation system recommends functions in spreadsheet by insert future calculated percentage here.

The remainder of the paper is structured as follows. We first review previous work on recommendation systems in softwares and related work on making spreadsheet formula usage efficient (Section II). We then provide details of the spreadsheet corpus used for our system and the details of the algorithm used (Section III) before describing our evaluation measures (Section IV) and their outcome (Section V). We discuss the various aspects of our system (Section VI) along with the limitations and future works (Section VII) based on our contribution in this paper afterwards.

II. RELATED WORK

Recommender systems are used generally to produce a list of predictions based on the preference of an user and the similarity of preferences of other users. Recommender systems assist users by recommending products and services to her based on her past preference in a scenario where the number of available options are extremely large. In recent years, recommender systems have become quite popular and have been applied successfully in multiple sectors of business and academia [7]–[9].

Collaborative filtering is one of the most prominent recommending technologies available. The core concept of collaborative filtering is to apply a nearest neighbor method between a user's preferences and the preference of a large user community, and provide the user with recommendations by extrapolating based on how her selection or preference relate to that of the community. Collaborative filtering has been applied to recommend a fringe of products and services, ranging from movies to school courses [8], [10]–[15].

However, there has been little effort to use recommendation systems to help users of a large software system with vast set of functionality to learn these functionalities. One of the notable attempts to use recommender systems to recommend command is the OWL [16], which makes command recommendations to Microsoft Word users. OWL recommends commands to an individual if she is not using certain commands at all but the community is using them on a frequent basis.

¹<https://support.office.com/en-us/article/Excel-functions-by-category-5f91f4e9-7b42-46d2-9bd1-63f26a86c0eb>

Matejka and colleagues developed a collaborative filtering-based approach to recommend commands in AutoCAD, a computer aided drafting software, called CommunityCommands [17]. The researchers of CommunityCommands showed that the collaborative filtering derived command recommendations were preferred by the AutoCAD users over the history based recommendation approach used in OWL.

In this paper, we build on these prior works by applying the concept of recommender systems to recommend functions in spreadsheets. We use the Fuse spreadsheet corpus as a source of function usage preference of the spreadsheet users' community and recommend functions contextualized to an individual user by applying collaborative filtering algorithm.

III. METHODOLOGY

According to the formal definition, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating)². Although collaborative filtering can be applied in many ways, most approaches falling under the category of user-based collaborative filtering are composed of mainly two distinct steps - find the users with similar preference patterns to that of the input user and then use the preference of the similar users to obtain predictions for the input user.

To apply this technique, we need to measure the similarity between two users. In order to do so, we define a vector for each user's spreadsheets and compare the vectors to find the similar preferences. Our function vector is defined such that the vector V for an individual user consists of cells, where each cell V_i represents the usage frequency of the function f_i .

To generate the input user's vector we extract function usage frequencies from multiple files created by the user. The more spreadsheets created by the same user is given as input the better representative vector for the user we get. As for the potential pool of similar user vectors, we decided to use Fuse, a spreadsheet corpus containing nearly 250 thousand unique spreadsheets extracted from the Common Crawl³ index [6]. The authors of Fuse also extracted meta information such as created by and function usage frequencies. We classified the spreadsheet meta information and discarded any spreadsheet that did not use any function. This reduced the number of spreadsheets under consideration to less than 13 thousand. These spreadsheets were then grouped by the user name (There are a large number of spreadsheets where this information is not present, how do we refer to them in the paper?). We proceeded to extract the function vector from each of these groups in the similar way the input user vector was extracted. The final number of function vectors in this pool of potential similar user vectors was `insert actual number from modified code here`.

To measure the similarity between users, we used the cosine similarity function, which measures the cosine of the angle

between the user's vectors as described above. Given the function vector for two users a and b as V_a and V_b respectively, the similarity function is:

$$similarity(a, b) = \cos(\theta_{V_a V_b}) = \frac{V_a \cdot V_b}{||V_a|| ||V_b||} = \frac{\sum_{k=1}^n V_{a_k} V_{b_k}}{\sqrt{\sum_{k=1}^n V_{a_k}^2} \sqrt{\sum_{k=1}^n V_{b_k}^2}}$$

When the similarity function evaluates closer to 0, the vectors are substantially orthogonal to each other, which indicates that the users are dissimilar. A similarity function value of closer to 1, indicates that the vectors are nearly collinear and the users are quite similar. For ease of our implementation we used the complement of the cosine similarity: $1 - similarity(i, j)$, which is often commonly referred to as *cosine distance*.

To find the similar user vectors for the given vector, we calculated the similarity function value between each of the vector in the pool and the input vector, after this the vectors in the pool were sorted based on their cosine distance in ascending order. We selected n most similar vectors from this ordered list, where n is a tuning parameter, which in our case was between 3 and 5.

We compare the frequencies of each individual function between each of the similar vector and the input vector, and add a function to the recommendation set only if the function had a frequency value in one of the similar vectors but not in the input vector. The set of recommended functions is then ordered by the cumulative frequency of the functions in the similar vectors, this way the functions in the recommendation list appear in order of their usage in the similar vectors.

IV. EVALUATION

For evaluating the system implemented based on the algorithm in the previous section, we used a cross validation [18] technique, which enabled us to evaluate our system using the existing feature vectors extracted from the Fuse corpus. Cross validation is a model evaluation technique, where the input data is partitioned into two complementary subsets - one subset is used to perform the analysis (training set) and the other is used to validate the analysis on the training set (testing set). Generally, multiple passes of cross validation are performed using different partitions to cope with variance in the dataset and the results are averaged.

We used a variation of cross validation, called Leave p -out (LPO) cross validation [19]. According to LPO, all possible training/testing partitions are generated by removing p samples from the complete set. The testing set consists of the p samples and the training set contains the rest $n - p$ samples, where n is the size of the original dataset. Thus, a dataset of length n will have $\binom{n}{p}$ possible partitions in LPO cross validation.

We used the function vectors extracted from Fuse to perform LPO cross validation. In LPO cross validation, such a function vector V_i will be partitioned into the training set $S_{training}$ and the testing set $S_{testing}$, where the size of $S_{testing}$ is p .

²https://en.wikipedia.org/wiki/Collaborative_filtering

³<http://www.commoncrawl.org>

The $S_{training}$ set with the rest of the function frequency set to 0, is used as an input vector for the collaborative filtering-based recommender system, which produces the recommendation set R_S . We define the correct number of recommendations for this partition as the number of functions that are both in set R_S and $S_{testing}$ and calculate the result of the LPO evaluation for this function vector h_i as:

$$h_i = \frac{\sum_{k=1}^n |R_{S_k} \cap S_{testing_k}|}{n}$$

Where k presents each distinct partition and n is $\binom{|V_i|}{p}$. We calculated h_i for every vector V_i in the function vector pool with a p value of 3.

V. RESULTS

VI. DISCUSSION

VII. LIMITATIONS & FUTURE WORK

ACKNOWLEDGMENT

REFERENCES

- [1] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers *et al.*, "The state of the art in end-user software engineering," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 21, 2011.
- [2] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE, 2005, pp. 207–214.
- [3] W. Winston, "Executive education opportunities millions of analysts need training in spreadsheet modeling, optimization, monte carlo simulation and data analysis," *OR MS TODAY*, vol. 28, no. 4, pp. 36–39, 2001.
- [4] M. Fisher and G. Rothermel, "The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, 2005, pp. 1–5.
- [5] F. Hermans and E. Murphy-Hill, "Enrons spreadsheets and related emails: A dataset and analysis," Delft University of Technology, Software Engineering Research Group, Tech. Rep., 2014.
- [6] T. Barik, K. Lubick, J. Smith, J. Slankas, and E. Murphy-Hill, "Fuse: A reproducible, extendable, internet-scale corpus of spreadsheets," 2015.
- [7] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 194–201.
- [8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994, pp. 175–186.
- [9] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.
- [10] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl, "Movielens unplugged: experiences with an occasionally connected recommender system," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 263–266.
- [11] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76–80, 2003.
- [12] J. Liu, C. K. Wong, and K. K. Hui, "An adaptive user interface based on personalized learning," *IEEE Intelligent Systems*, no. 2, pp. 52–57, 2003.
- [13] S. M. McNee, N. Kapoor, and J. A. Konstan, "Don't look stupid: avoiding pitfalls when recommending research papers," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. ACM, 2006, pp. 171–180.
- [14] R. Farzan and P. Brusilovsky, "Social navigation support in a course recommendation system," in *Adaptive hypermedia and adaptive web-based systems*. Springer, 2006, pp. 91–100.
- [15] M.-H. Hsu, "A personalized english learning recommender system for esl students," *Expert Systems with Applications*, vol. 34, no. 1, pp. 683–688, 2008.
- [16] F. Linton, D. Joy, H.-P. Schaefer, and A. Charron, "Owl: A recommender system for organization-wide learning," *Educational Technology & Society*, vol. 3, no. 1, pp. 62–76, 2000.
- [17] J. Matejka, W. Li, T. Grossman, and G. Fitzmaurice, "Community-commands: command recommendations for software applications," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, 2009, pp. 193–202.
- [18] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection." Morgan Kaufmann, 1995, pp. 1137–1143.
- [19] S. Arlot, A. Celisse *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.