Recommending Functions in Spreadsheets from the Fuse Corpus

Shaown Sarker, Matthew Neal, Nisarg Vinchhi
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
{ssarker,meneal,nvinchh}@ncsu.edu

Abstract

Spreadsheets are the most widely used form of end-user programming. Although spreadsheets have a large array of functions built-in, the majority of spreadsheet users often do not utilize them to perform their tasks. To address this lack of function usage, we investigate recommender system technologies and consider two distinct approaches to a function recommender system for spreadsheets. In our work, we apply two variations of collaborative filtering algorithm to produce personalized function recommendations to spreadsheet users by applying these algorithms on the Fuse spreadsheet corpus. We compare the performance of the algorithms used with a baseline of most popular algorithm, also known as Linton's algorithm. In this paper, we present the feature extraction process from the spreadsheet corpus, the algorithms used and their comparative performance. Finally, we also outline the roadmap to use our findings to create an Excel plugin for increasing functional awareness.

1 Introduction

Ko et al. define end-user programmers as people who are not professional software developers, but make use of tools and processes that lets them perform tasks similar to programming [1]. According to a study from 2005 [2], nearly 23 million Americans use spreadsheets, constituting 30% of the entire workforce. Spreadsheets are used not only for home and small businesses, but they are also very commonly used in industry. Almost 90% of all analysts use spreadsheets to perform their calculations [3]. Based on their numbers, spreadsheet users form the largest demographic within end-user programmers.

Spreadsheet software come with lots of functions built-in. Consider Microsoft Excel, the most widely used spreadsheet application, which has more than 300 functions[1]. But there are many situations where the user neglects using them. Let us consider the case of a spreadsheet user, Titus. Titus is a fourth grade teacher in an elementary school. At the end of the school year, he wants to calculate the class grades in Excel from all the test scores entered into his spreadsheet. Instead of using the arithmetic functions in Excel, he reaches for his hand-held calculator, and uses it to calculate and enter the grade for each student. This type of usage, not taking advantage of the functions in spreadsheets, is very common with users where they lack awareness of functionality [4] of the end-user programming tool being used.

Functionality awareness is important to accomplish new tasks as well as achieving efficient completion of existing tasks. Systems to recommend commands have been used successfully to improve

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

functionality awareness in large and complex software applications [5, 6]. 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. In recent years, recommender systems have become quite popular and have been applied successfully in multiple sectors of business [7] and academia [8, 9].

Our contribution in this paper – we recommended functions in spreadsheets by applying two distinct variations of collaborative filtering algorithm and compare the effectiveness of the recommendations using a cross validation based automated evaluation. We also compared their performance for recommending functions against another algorithm priorly used for recommending commands in large software systems, the most popular algorithm [10]. And we conclude by discussing how the results of our work can be used to create an Microsoft Excel plugin that can assist users to increase functional awareness by recommending functions they can use.

2 Related Work

Researchers have been studying and analyzing spreadsheets to better comprehend the activity of the users and design better tools to assist them. Previous research has focused on extracting structured domain information from spreadsheets [11] and visualizing spreadsheet using dataflow diagrams [12] to enhance understandability of spreadsheets. For improving the maintainability and quality of the spreadsheet formulas, systems have been implemented to detect code smells in formulas and refactor them to resolve the detected smells [13]. In contrast, our work attempts to increase the functionality awareness in spreadsheets by suggesting functions.

Moreover, some research has used recommender systems to help users of a large software system with vast set of functionality to learn these functionalities. One of the early notable attempts is the OWL system [10] developed by Linton et al., which suggests commands to Microsoft Word users based on the most popular algorithm. In OWL, commands are recommended to an individual if she is not using certain commands at all but the community is using them on a highly frequent basis. The underlying assumption of this system is that all users of a software system have a similar command usage distribution. Recommender systems were developed to improve upon the system delineated by Linton.

Matejka and colleagues developed a collaborative filtering-based approach to recommend commands in AutoCAD, a computer aided drafting software, called CommunityCommands [5]. Similarly based on the users' usage history, Murphy-Hill and colleagues studied several existing recommendation algorithms to recommend commands in Eclipse. The approach that we propose in this paper, focuses on the algorithm used by OWL as a baseline and the user-based and item-based variations of the collaborative filtering algorithm used by CommunityCommands.

Both of the algorithms used in our work requires a source of function usage preference for the spreadsheet users' community. Although there are existent spreadsheet corpora like Enron [14] and EUSES [15] which have been valuable sources for spreadsheet analysts and researchers alike, in this paper we look into a very recently extracted spreadsheet corpus, Fuse [16]. Unlike Enron and EUSES which are very domain specific, Fuse contains a more diverse and much larger body of spreadsheets. Fuse is also reproducible, thus the recommender system in our work can leverage of a larger spreadsheet corpus if needed, which can be extracted by the system developed by Fuse's creators.

3 Headings: first level

First level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

3.1 Headings: second level

Second level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

3.1.1 Headings: third level

Third level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

4 Citations, figures, tables, references

These instructions apply to everyone, regardless of the formatter being used.

4.1 Citations within the text

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BIBTEX style `unsrt` produces this.) As to the format of the references themselves, any style is acceptable as long as it is used consistently.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]", not "In our previous work [4]". If you cite your other papers that are not widely available (e.g. a journal paper under review), use anonymous author names in the citation, e.g. an author of the form "A. Anonymous".

4.2 Footnotes

Indicate footnotes with a number[2] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[3]

4.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

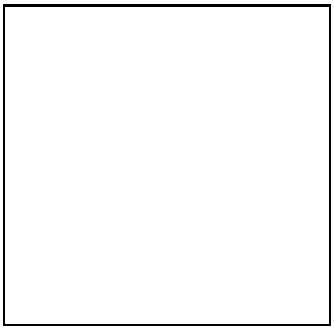

Figure 1: Sample figure caption.

[2] Sample of the first footnote

[3] Sample of the second footnote

Table 1: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

4.4 Tables

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

5 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

6 Preparing PostScript or PDF files

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- LaTeX users:
  - Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.
  - Otherwise, please generate your PostScript and PDF files with the following commands:

    ```
    dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
    ps2pdf mypaper.ps mypaper.pdf
    ```

    Check that the PDF files only contains Type 1 fonts.
  - xfig "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
  - The `\bbold` package almost always uses bitmap fonts. You can try the equivalent AMS Fonts with command

    ```
    \usepackage[psamsfonts]{amssymb}
    ```

    or use the following workaround for reals, natural and complex:

    ```
    \newcommand{\RR}{I\!\!R} %real numbers
    \newcommand{\Nat}{I\!\!N} %natural numbers
    \newcommand{\CC}{I\!\!\!\!C} %complex numbers
    ```

  - Sometimes the problematic fonts are used in figures included in LaTeX files. The ghostscript program `eps2eps` is the simplest way to clean such figures. For black and white figures, slightly better results can be achieved with program `potrace`.

- MSWord and Windows users (via PDF file):
  - Install the Microsoft Save as PDF Office 2007 Add-in from `http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=4d9`
  - Select "Save or Publish to PDF" from the Office or File menu
- MSWord and Mac OS X users (via PDF file):
  - From the print menu, click the PDF drop-down box, and select "Save as PDF..."
- MSWord and Windows users (via PS file):
  - To create a new printer on your computer, install the AdobePS printer driver and the Adobe Distiller PPD file from `http://www.adobe.com/support/downloads/detail.jsp?ftpID=204` *Note:* You must reboot your PC after installing the AdobePS driver for it to take effect.
  - To produce the ps file, select "Print" from the MS app, choose the installed AdobePS printer, click on "Properties", click on "Advanced."
  - Set "TrueType Font" to be "Download as Softfont"
  - Open the "PostScript Options" folder
  - Select "PostScript Output Option" to be "Optimize for Portability"
  - Select "TrueType Font Download Option" to be "Outline"
  - Select "Send PostScript Error Handler" to be "No"
  - Click "OK" three times, print your file.
  - Now, use Adobe Acrobat Distiller or ps2pdf to create a PDF file from the PS file. In Acrobat, check the option "Embed all fonts" if applicable.

If your file contains Type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

6.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package. Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

[1] Andrew J Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, et al. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, 43(3):21, 2011.

[2] Christopher Scaffidi, Mary Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. In *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*, pages 207–214. IEEE, 2005.

[3] WL Winston. Executive education opportunities millions of analysts need training in spreadsheet modeling, optimization, monte carlo simulation and data analysis. *OR MS TODAY*, 28(4):36–39, 2001.

[4] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 649–658. ACM, 2009.

[5] Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. Communitycommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 193–202. ACM, 2009.

[6] Emerson Murphy-Hill, Rahul Jiresal, and Gail C Murphy. Improving software developers' fluency by recommending development environment commands. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, page 42. ACM, 2012.

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

[8] Mei-Hua Hsu. A personalized english learning recommender system for esl students. *Expert Systems with Applications*, 34(1):683–688, 2008.

[9] Sean M McNee, Nishikant Kapoor, and Joseph 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*, pages 171–180. ACM, 2006.

[10] Frank Linton, Deborah Joy, Hans-Peter Schaefer, and Andrew Charron. Owl: A recommender system for organization-wide learning. *Educational Technology & Society*, 3(1):62–76, 2000.

[11] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Automatically extracting class diagrams from spreadsheets. In *ECOOP 2010–Object-Oriented Programming*, pages 52–75. Springer Berlin Heidelberg, 2010.

[12] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Breviz: Visualizing spreadsheets using dataflow diagrams. *arXiv preprint arXiv:1111.6895*, 2011.

[13] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineering*, 20(2):549–575, 2015.

[14] Felienne Hermans and Emerson Murphy-Hill. Enrons spreadsheets and related emails: A dataset and analysis. Technical report, Delft University of Technology, Software Engineering Research Group, 2014.

[15] Marc Fisher and Gregg Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–5. ACM, 2005.

[16] Titus Barik, Kevin Lubick, Justin Smith, John Slankas, and Emerson Murphy-Hill. Fuse: A reproducible, extendable, internet-scale corpus of spreadsheets. 2015.