The Journal of Computing Sciences in Colleges

Papers of the 17th Annual CCSC Mid-South Conference

April 12-13, 2019
University of Arkansas Little Rock
Little Rock, AR

Volume 34, Number 7 **April 2019**

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges. Printed in the USA. POSTMASTER: Send address changes to Susan Dean, CCSC Membership Secretary, 89 Stockton Ave, Walton, NY 13856.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors **5**

CCSC National Partners & Foreword **7**

Welcome to the 2019 CCSC Mid-South Conference **9**

Regional Committees — 2019 CCSC Mid-South Region **10**

Reviewers — 2019 CCSC Mid-South Conference **11**

Generating Synthetic Data to Support Entity Resolution Education and Research **12**
Yumeng Ye, John R. Talburt, University of Arkansas at Little Rock

Flipping One Day Each Week in a Smaller CS1 Course: An Experience Report **20**
Mark Hodges, Dominican University

Categorizing And Ameliorating Generalization-Specialization Design Mistakes in an Intermediate Programming Class **28**
John W. Coffey, University of West Florida

Scoring Matrix Combined With Machine Learning For Heterogeneously Structured Entity Resolution **38**
Xinming Li, John R. Talburt, Ting Li, Xiangwen Liu, University of Arkansas at Little Rock

Digital Distraction Outside the Classroom: An Empirical Study **46**
Rajvardhan Patil,Matt Brown, Mohamed Ibrahim, Jeanine Myers, Kristi Brown, Muhammad Khan, Rebecca Callaway, Arkansas Tech University

Data Science Academic Programs in the U.S. **56**
Ismail Bile Hassan, Jigang Liu, Metropolitan State University

An Analysis of the Effect of Stop Words on the Performance of the Matrix Comparator for Entity Resolution **64**
Awaad Alsarkhi, John R. Talburt, University of Arkansas at Little Rock

Developing a Guided Peer-Assisted Learning Community for CS Students **72**
Yi Liu, Gita PhelpsA, Georgia College and State University, Fengxia Yan, Morehouse School of Medicine

Google Analytics — Conference Tutorial **81**
Daniel Brandon, Christian Brothers University

Software Design Patterns Applied To Building An Interpreter — Conference Tutorial **83**
Larry Morell, Arkansas Tech University

Teaching Object-Oriented Programming with Geometry — Conference Tutorial **84**
Serge Salan, Christian Brothers University

Scalable Processing of Massive Text Data Stores for NLP — Conference Tutorial **85**
Brittany Bright, Cesar Cuevas, Israel Cuevas, Andrew Mackey, University of Arkanas at Fort Smith

Longest Pattern Lock — Nifty Assignment **86**
Jingsai Liang, Westminster College

Understanding the Identity Function in SML by Theory — Nifty Assignment **87**
Cong-Cong Xing, Nicholls State University, Jun Huang, Chongqing Univ. of Posts and Telecommunications

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Jeff Lehman, President (2020), (260)359-4209, jlehman@huntington.edu, Mathematics and Computer Science Department, Huntington University, 2303 College Avenue, Huntington, IN 46750.

Karina Assiter, Vice President (2020), (802)387-7112, karinaassiter@landmark.edu.

Baochuan Lu, Publications Chair (2021), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Department of Computer and Information Sciences, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2020), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

Susan Dean, Membership Secretary (2019), Associate Treasurer, (607)865-4017, Associate Editor, susandean@frontier.com, UMUC Europe Ret, US Post: 89 Stockton Ave., Walton, NY 13856.

Judy Mullins, Central Plains Representative (2020), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, School of Computing and Engineering, 5110 Rockhill Road, 546 Flarsheim Hall, University of Missouri - Kansas City, Kansas City, MO 64110.

John Wright, Eastern Representative (2020), (814)641-3592, wrightj@juniata.edu, Juniata College, 1700 Moore Street, Brumbaugh Academic Center, Huntingdon, PA 16652.

David R. Naugler, Midsouth Representative(2019), (317) 456-2125, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

Lawrence D'Antonio, Northeastern Representative (2019), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

Cathy Bareiss, Midwest Representative (2020), cbareiss@olivet.edu, Olivet Nazarene University, Bourbonnais, IL 60914.

Brent Wilson, Northwestern Representative (2021), (503)554-2722, bwilson@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.

Mohamed Lotfy, Rocky Mountain Representative (2019), Information Technology Department, College of Computer & Information Sciences, Regis University, Denver, CO 80221.

Tina Johnson, South Central Representative (2021), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308-2099.
Kevin Treu, Southeastern Representative (2021), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.
Bryan Dixon, Southwestern Representative (2020), (530)898-4864, bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

Serving the CCSC: These members are serving in positions as indicated:
Brian Snider, Associate Membership Secretary, (503)554-2778, bsnider@georgefox.edu, George Fox University, 414 N. Meridian St, Newberg, OR 97132.
Will Mitchell, Associate Treasurer, (317)392-3038, willmitchell@acm.org, 1455 S. Greenview Ct, Shelbyville, IN 46176-9248.
John Meinke, Associate Editor, meinkej@acm.org, UMUC Europe Ret, German Post: Werderstr 8, D-68723 Oftersheim, Germany, ph 011-49-6202-5777916.
Shereen Khoja, Comptroller, (503)352-2008, shereen@pacificu.edu, MSC 2615, Pacific University, Forest Grove, OR 97116.
Elizabeth Adams, National Partners Chair, adamses@jmu.edu, James Madison University, 11520 Lockhart Place, Silver Spring, MD 20902.
Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Dept. of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.
Deborah Hwang, Webmaster, (812)488-2193, hwang@evansville.edu, Electrical Engr. & Computer Science, University of Evansville, 1800 Lincoln Ave., Evansville, IN 47722.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Platinum Partner

Turingscraft
Google for Education
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Bronze Partners

National Center for Women and Information Technology
Teradata
Mercury Learning and Information
Mercy College

Foreword

Welcome to the 2019 issues of our journal for the CCSC spring 2019 conferences: Southwestern (March 22-23), Central Plains (April 5-6), South Central (April 5), Mid-south (April 12-13), and Northeastern (April 12-13).

Please plan to attend one or more conferences, where you can meet and exchange ideas with like-minded computer science educators. Each conference covers a variety of topics that are practical and stimulating. You can find detailed conference programs on the conference websites, which are listed on the CCSC conferencecalendar: `http://www.ccsc.org/regions/calendar.`

From January 2019, this journal will be published electronically on the CCSC website and links to the journal issues will be sent to CCSC members via email. Those of you who would like hard copies of journal issues can order them from Amazon. Simply search for "CCSC Journal" to find available issues. The journal will continue to be available in the ACM Digital Library.

As an author, you may post your papers published by CCSC on any website. Please make sure to use the PDF versions of your papers with CCSC's copyright box. Such PDFs can be downloaded from the ACM Digital Library or extracted from our electronic journal.

Please feel free to email me directly at `blu@sbuniv.edu` if you notice any issue with our publications.

Baochuan Lu
Southwest Baptist University
CCSC Publications Chair

Welcome to the 2019 CCSC Mid-South Conference

On behalf of the University of Arkansas at Little Rock, I extend greetings and a warm welcome to the attendees of the 17th Annual CCSC Mid-South Conference. We are honored to have the privilege of hosting the conference on April 12-13, 2019. A special thanks to Dean Lawrence Whitman, for providing the space for the conference events in the George W. Donaghey College of Engineering and Information Technology building.

A lot of hard work has gone into the planning of the conference, and the Conference Steering Committee has put together an outstanding program. This year the program includes 12 papers, 4 tutorials, and two Nifty Assignments. The conference will begin on Friday with a student programming contest. There will also be student poster sessions and a vendor showcase in addition to the paper sessions, tutorials, and Nifty Assignment sessions.

The CCSC-MS Conference has been a wonderful platform for sharing ideas and building community among the colleges and universities in our region. I want to especially thank the faculty members who have taken time from the regular duties to make this conference happen. I also want to thank all of the students and faculty who submitted their papers, posters, tutorials, and assignments, the program committee for reviewing the submissions, and the volunteers who moderate these sessions.

Finally, I want to thank everyone attending in the conference. I hope it will be a productive and satisfying experience for each of you. While you are here, I hope you will have time to tour our campus and meet some of our students. We are really happy to have you here on our campus. If there is anything at all we can do to help while you are here, please let us know. Also please consider helping with the 2020 CCSC-MS Conference. If you are interested in hosting or assisting in any way, please contact one of the Steering Committee members to find out how you can contribute.

John R. Talburt
University of Arkansas at Little Rock
CCSC-MS Site Chair
Gabriel Ferrer
Hendrix College
CCSC-MS Conference Chair

2019 CCSC Mid-South Conference Steering Committee

Gabriel Ferrer, Conference Chair ..... Hendrix College
John Talburt, Site Chair ..... University of Arkansas at Little Rock
James McGuffee, Papers Co-chair ..... Christian Brothers University
David Sonnier, Papers Co-chair ..... Lyon College
David Middleton, Panels/Workshops/Tutorials Chair ..... Arkansas Tech University
Matt Brown, Nifty Ideas Co-chair ..... Arkansas Tech University
Kriangsiri 'Top' Malasri, Student Programming Contest Co-chair . University of Memphis
Brent Yorgey, Student Programming Contest Co-chair ..... Hendrix College
David Middleton, Student Papers Chair ..... Arkansas Tech University
David Middleton, Past Conference Chair ..... Arkansas Tech University

Regional Board — 2019 CCSC Mid-South Region

Gabriel Ferrer, Board Chair ..... Hendrix College
David Naugler, Editor ..... Southeast Missouri State University
Mark Goadrich, Registrar ..... Hendrix College
Brian McLaughlan, Treasurer ..... University of Arkansas - Fort Smith
David Hoelzeman, Webmaster ..... Arkansas Tech University
David Naugler, National Board Representative ..... Southeast Missouri State University

Reviewers — 2019 CCSC Mid-South Conference

Matt Brown ..... Arkansas Tech University, Russellville, AR
Gabriel Ferrer ..... Hendrix College, Conway, AR
Mark Goodrich ..... Hendrix College, Conway AR
Kriangsiri Malasri ..... University of Memphis, Memphis, TN
Larry Morell ..... Arkansas Technical University, Russellville, AR
David Naugler ..... Southeast Missouri State University (retired), Cape Girardeau, MO
Melody Penning University of Arkansas for Medical Sciences, Little Rock, AR
Pei Wang ..... University of Arkansas for Medical Sciences, Little Rock, AR
Cong-Cong Xing ..... Nicholls State University, Thibodaux, LA
Brent Yorgey ..... Hendrix College, Conway, AR

Generating Synthetic Data to Support Entity Resolution Education and Research*

Yumeng Ye and John R. Talburt
Department of Information Science
University of Arkansas at Little Rock
Little Rock, AR 72204
{yxye1,jrtalburt}@ualr.edu

Abstract

Almost all organizations use some type of Entity Resolution (ER) methods to uniquely identify their customers and vendors across different channels of contact. In the case of persons, this requires the use of personally identifying information (PII) such as name, address, phone number, and email address. Because of the growing concerns over data privacy and identity theft, organizations are reluctant to release personally-identifiable customer information even for education and research purposes. An alternative is to generate synthetic data to use in student exercises and for research related to entity resolution methods and techniques. One advantage of synthetically generated data for ER is it can be fully annotated with the correct linking making it very easy to calculate the precision and recall of linking operations. This paper discusses a simple method to generate synthetic data as input for ER processes. The method allows the user to randomly assign certain types and levels of data quality errors along with other types of non-error variations to the data, such as nicknames, different date formats, and changes in address. For ER research in particular, the method can create introduce data redundancy by copying records referencing the same person into the same file or into different files with different record layouts.

Background

Entity Resolution (ER) is the process of linking two references in the information system which refers to the same entity in the real world [5]. In industry, it can happen that the same customer may interact with the company through multiple channels. Therefore, organizations need powerful ER tools to recognize and classify their customers throughout the system. In academic area, both ER education program and research projects require large amount of sample data for teaching or testing purpose. Some research projects involve with how to improve the efficiency of record linking under big data. However, it is difficult to obtain real-world datasets for research and academic education. When real data is hard to be obtained, or specific data characteristics are required in ER research, generating synthetic data can often solve the problem [4].

Introduction

In real-world data, one entity can have several records in the system or across multiple data sources; and these records may not be identical. For example, if one person moves to a new location, the system could have a new record with different address information. It can also happen that in one company, different departments have some records of the same person, but several attribute values are changed. In addition, several common data quality issues can also exist across the same entitys records, such as missing value, digits transpose, inconsistent formats etc. This paper describes a computer program written in R language [2], which was designed to generate synthetic data and include 14 types of data quality issues. The original entities are generated with a random number of duplicate records, then the users can customize the level of data quality issues randomly assigned to each of the records. The outputs of the program are multiple data files containing copies of the records derived from original entities. Each data file has customized delimiters and attributes. This synthetic data generator can be used for research and instruction in entity resolution and information quality.

Preparation of Seed Data

For the experiments described in this paper, the names (including gender) and addresses generated by the program were derived from two sources. One is a sample of publicly available occupancy records with addresses in four U.S. states, California, Texas, Florida, and Arkansas. The other is a R package

called rlErrorGeneratoR package used to introduce errors into a dataset and create a dirty version of the data [1]. rlErrorGeneratoR contains a dataset which includes 253,326 unique fake person name, gender and address information. In addition, several other attributes including social security number, homephone, date-of-birth, were derived from a R package called generator, which generates data containing fake person-identifying information [3].

The synthetic identity of a person in this generator comprises the following items

- Identity Number
- Name comprising 3 items: First, Middle, Last
- Gender Code (M, F, U) consistent with the First Name value
- Social Security Number (9-digit value in XXX-XX-XXXX format)
- Home Phone (10-digit in (XXX)-XXX-XXXX format)
- Date-of-Birth (date value in Microsoft Excel date format)

To create the original entities table for this generator, first use the rlErrorGeneratoR package to generate unique fake person name, gender and address information. Each address information includes street number, street name, city, state, zip code. Then use generator package to create fake social security number, home phone and date-of birth for each corresponding entity. Finally, a record ID was assigned to each entity. The original entities table has 253,326 rows, and 9 columns. Figure 1 shows show some of the rows of this table.

recid	fname	lname	mname	gender	full_address	ssn	homephone	dob
1	IAN	AADLAND	LARS	M	7715 ABINGTON DR, KERNERSVILLE, NC 27284	490-46-2048	(361)-924-5829	1911/8/25
2	KAVASSANA	AANAI	HIKARI	F	2165 MAURINE WAY, WINSTON SALEM, NC 27127	756-23-7625	(483)-549-7645	1906/4/6
3	JUDY	AANSTAD	ANN	F	221 HARMON CT, WINSTON SALEM, NC 27106	555-37-4439	(247)-793-3157	2008/9/11
4	MYRA	AARGAARD-ESPERSEN	NA	F	1224 MAGNOLIA ST, WINSTON SALEM, NC 27103	117-15-8521	(183)-249-7916	1927/10/7
5	ALLEN	AARON	IKAIKA	M	3830 COUNTRY CLUB RD # J, WINSTON SALEM, NC 27104	784-85-1840	(642)-697-9754	1964/11/1
6	ANDREW	AARON	STEPHEN	M	2475 SPICEWOOD DR, WINSTON SALEM, NC 27106	601-70-6106	(159)-928-5341	1917/3/21
7	ANGELA	AARON	DENSIE	F	1117 E SEVENTEENTH ST, WINSTON SALEM, NC 27105	636-32-8781	(738)-153-1254	2016/7/7
8	ANINA	AARON	ROSSOUW	F	1740 ABBOTTSFORD DR, KERNERSVILLE, NC 27284	378-53-9395	(687)-894-2439	2015/4/22
9	CELIA	AARON	KAY	F	6965 WEST RD, WALNUT COVE, NC 27052	344-37-4232	(521)-235-6821	1941/12/28
10	CHRISTOPHER	AARON	NA	M	2013 WILLIAMSBURG MANOR CT, WINSTON SALEM, NC 27103	741-75-7354	(235)-523-5439	1977/5/24
11	DARRYL	AARON	WARREN	M	624 N STRATFORD RD, WINSTON SALEM, NC 27104	282-85-6947	(495)-756-2417	1909/6/29
12	DAVIS	AARON	SCOTT	M	3211 KINNAMON RD, WINSTON SALEM, NC 27104	834-59-6144	(376)-612-9375	2004/1/10
13	DONALD	AARON	DWIGHT	M	6965 WEST RD, WALNUT COVE, NC 27052	131-53-2482	(314)-459-3419	2013/11/21
14	DOUGLAS	AARON	NA	M	3401 LOCHURST CT, PFAFFTOWN, NC 27040	692-52-6046	(873)-465-8247	1972/2/5
15	GREGORY	AARON	ALFRED	M	7514 DIVALDI ST, LEWISVILLE, NC 27023	672-52-2262	(298)-347-6438	1910/7/15
16	JAMES	AARON	GARFIELD	M	2599 SHIELDALE DR, WINSTON SALEM, NC 27107	986-66-9423	(839)-863-1249	1958/2/16
17	JEREMY	AARON	TYLER	M	3211 KINNAMON RD, WINSTON SALEM, NC 27104	363-19-7202	(182)-264-6792	1911/8/14

Figure 1: Example Records from the Original Entities Table.

Several additional tables must be created for the error generation process. One is the nickname lookup table. The nickname lookup table has 4 columns: looku_id, lookup_name, lookup_alternate, lookup type. When the nickname variation is generated, the system looks up the original name in the table, and replaces it with the alternate name is replaced. Figure 2 shows part of the nickname lookup table as an example.

Another table is a "new address table." It is used to replacing an address in the original dataset to simulate when a person moves and changes address.

lookup_id	lookup_name	lookup_alternate	lookup_type
1	AARON	RON	to_nick
2	ABEL	ABE	to_nick
3	ABEDNEGO	BEDNEY	to_nick
4	ABIJAH	AB	to_nick
5	ABIJAH	BIGE	to_nick
6	ABIGAIL	AB	to_nick
7	ABIGAIL	ABBIE	to_nick
8	ABIGAIL	ABBY	to_nick
9	ABIGAIL	ABBY	to_nick
10	ABIGAIL	GAIL	to_nick
11	ABIGAIL	NABBY	to_nick
12	ABNER	AB	to_nick

Figure 2: Example Records in the Nickname Lookup Table.

The new address table only has one column which is in the same full address format as the address in the original dataset. Figure 3 shows part of the new address table as an example.

newaddress
11881 GULF POINTE DR APT E38, HOUSTON, TX 77089
4149 WALSH LN , GRAND PRAIRIE, TX 75052
2943 N COTTONWOOD ST UNIT 3, ORANGE, CA 92865
9247 TOBIAS AVE , PANORAMA CITY, CA 91402
3536 N BERLIN AVE , FRESNO, CA 93722
10237 CASANES AVE , DOWNEY, CA 90241
1841 MICHIGAN ST , COLTON, CA 92324
2943 N COTTONWOOD ST UNIT 3, ORANGE, CA 92865
9247 TOBIAS AVE , PANORAMA CITY, CA 91402
3536 N BERLIN AVE , FRESNO, CA 93722
12840 SANTA LUCIA RD , ATASCADERO, CA 93422

Figure 3: Example Records of New Address Table.

Data Error Functions

Several types of data error and variation can be created by rlErrorGeneratoR package.

1. Generate duplicate records (data redundancy)
2. Swap the value between two columns (misfielding error).
3. Remove hyphens, parentheses, forward slash.
4. Randomly transpose two characters of a string.
5. Repeat random one character.
6. Only keep the initial (first) character of a string.
7. Convert as name to a nickname (name variation)

8. Change females last name (change in marital status).
9. Change address (moving to a new address)
10. Create twins identity (one of the most problematic issues for ER)
11. Randomly delete entire value (missing value error)
12. Only capitalize the first letter (letter casing variation)
13. Convert to lower case (letter casing variation)

The logic of all functions are almost the same. The user only needs to specify the name of the original entities table, the attribute to be changed, and the percentage of errors to generate. The error generator function randomly samples a number of records, replacing the value with the error or variation. Each function returns two datasets. The first dataset is a completely new table, in which the selected changes have been made. The second dataset is an error lookup table, which contains the record ID, type of error, name of the attribute, string before changing, string after changing. Figure 4 shows the R code for generating the variation which only keeps the initial (first character) of a selected name value.

```
initials <- function(df,n_errors,col_names,recid){
  lookup <- data.frame()
  col <- col_names
  rows <- sample(nrow(df), n_errors)
  for(i in 1:length(rows)){
    row <- rows[i]
    beforestring <- df[row,col]
    df[row,col] <- substr(beforestring,1,1)
    afterstring <- df[row,col]
    lookup_i <- data.frame(recid = df$recid[row],
                           error_type = "initials",
                           column = col_names,
                           before = beforestring,
                           after = afterstring)
    lookup <- rbind(lookup,lookup_i)
  }
  initials_combo = list(df = df,lookup = lookup)
  initials_combo
}
```

Figure 4: Generating the Variation Keeping the Initial of a Name Value.

Synthetic Data Generation

After the seed data are prepared, the generator can create synthetic data records by the following procedure:

1. Duplicate Record Generation

(a) Define a random number range of duplicate records to be generated for each entity.
(b) Create a column called id_truth, in which each group of duplicate records has the same id_truth (annotation).

2. Random Error Assignment
   (a) Randomly select a certain number of records and assign the data error function to certain attributes of the records.
   (b) Use the output dataset generated from one error function to create as an input to another data error function.
   (c) Merge all error lookup tables together to generate single set of synthetic records with the desired errors and variations.
3. Create Multiple Files
   (a) Create the truth set by extracting the record ID and id_truth from the dataset.
   (b) Define the number of files to be created.
   (c) Randomly extract a certain number of records and write them to a separate data file.
   (d) Reformat the record ID for each data file.
   (e) Update the error lookup table with the new record ID.
   (f) Reformat the attributes for each data file. For example:
      i. Parse the address into street, city, state, zip in one file but leave together in another file
      ii. Remove or include some attributes such as date-of-birth or gender in one file, but not another
   (g) Assign different delimiters for different data files, such as comma delimited, pipe delimited.
4. Output all the data files.

A more detailed application procedure is provided below as an example. The example shows the generating error notes when applying these steps. In the end, three data files are created which have 506,288 records in total.

1. Generate a random number between 1-7 for duplicate records in **each entity**.
2. Randomly select 10,000 records, swap the value in **fname**, **lname** column.
3. Randomly select 100,000 records, remove the hyphens in **ssn**.
4. Randomly select 50,000 records, remove the parentheses in **homephone**.
5. Randomly select 50,000 records, remove the hyphens in **homephone**.
6. Randomly select 100,000 records, remove the forward slash in **dob**.

7. Randomly select 10,000 records, transpose two characters in **ssn**.
8. Randomly select 20,000 records, transpose two characters in **homephone**.
9. Randomly select 20,000 records, randomly repeat one character in **fname**.
10. Randomly select 20,000 records, randomly repeat one character in **lname**.
11. Randomly select 100,000 records, only keep the initial of **mname**.
12. Randomly select 10,000 records, only keep the initial of **fname**.
13. If the **fname** exists in both the data and nickname table, convert to nickname.
14. Randomly select 10,000 records (females only), change **lname** for marriage.
15. Randomly select 50,000 records, change the **full_address**.
16. Randomly select 50 records, for **twins identity (only change fname and ssn)**.
17. Randomly select 50,000 records, delete **ssn**.
18. Randomly select 50,000 records, delete **homephone**.
19. Randomly select 50,000 records, delete **dob**.
20. Randomly select 50,000 records, delete **mname**.
21. Randomly select 50,000 records, delete **fname**.
22. Randomly select 50,000 records, delete **gender**.
23. Randomly select 50,000 records, **fname, lname, mname** capitalize first letter
24. Randomly select 500,000 records, change **full_address** to lowercase
25. Create full error lookup table
26. Randomly assign 30% of the records to List A, 40% to List B, 30% to List C.
27. Reformat (assign new) recid for List A (A848132, A999999)
28. Reformat (assign new) recid for List B (B858256, B999999)
29. Reformat (assign new) recid for List C (C787384, C999999)
30. Update Lists A,B,C full_error_lookup table with new recid.
31. Create truth table named as TruthABC.
32. Assign attributes to List A, B, C.
    (a) List A: fname, lname, mname, address, city, state, zip,ssn,homephone
    (b) List B: full name, street number, address, city, state, zip, ssn,dob
    (c) List C: fname_lname,mname, gender, full_address, homephone, dob
33. Output Lists A, B, C with different structures
    (a) List A: tab delimited
    (b) List B: quote comma delimited, no column names
    (c) List C: | delimited
34. Output full_error_lookup table, truth table

Conclusion and Future Work

Several synthetic data files which were generated by this method, are currently used in both information quality courses and entity resolution research projects. The students use different entity resolution methodologies, such as deterministic matching, probabilistic matching, and machine learning to link records together across different data files. The linking performance can then be measures by comparing the process created links with the true links in the truth set. The research team has written a Java program to take the process generated links and true links as input and automatically generate the precision, recall and F-measure obtained by the linking process.

However, it is not clear whether the synthetic data generated by this method can satisfy all types of data quality and entity resolution research. In particular, the effectiveness of the method to generate data supporting research on probabilistic matching for record linking. For this type of research in addition to making individual records reflect realistic data entry errors, the data also need to ensure that the higher the frequency of a name value, the higher the number of different persons using the name value, a fundamental assumption of probabilistic matching. Therefore, the next step is to confirm whether this approach can generate data with these characteristics.

References

[1] rlErrorGeneratoR. `https://github.com/ilangurudev/rlErrorGeneratoR`.

[2] What is R? `https://www.r-project.org/about.html`.

[3] Package 'generator' generate data containing fake personally identifiable information, 2016. `https://cran.r-project.org/web/packages/generator/generator.pdf`.

[4] J.R. Talburt, Y. Zhou, and S.Y. Shivaiah. SOG: A synthetic occupancy generator to support entity resolution instruction and research. In *2009 International Conference on Information Quality*, 2009.

[5] Y. Zhou and J. Talburt. Entity dentity information management (EIIM). In *International Conference on Information Quality*, pages 327–341, 2011.

Flipping One Day Each Week in A Smaller CS1 Course: An Experience Report*

Mark Hodges
Computer Science Department
Dominican University
River Forest, IL 60302
mhodges@dom.edu

Abstract

This paper describes the authors experience using recorded video to flip one day each week of a CS1 course taught at a small liberal arts university. By moving content from class onto videos, the author was able to use one class period each week to treat as a lab for student practice with the material. Experiences from the two semesters this approach was used, and lessons learned from those semesters are presented, including important considerations about course structure as well as mechanisms to ensure that students watch the videos.

Introduction

In teaching a first programming course that does not have a lab component, I found a particular difficulty in fitting much hands-on active time into the class time. As a result, students were getting their first extensive programming experience with many concepts only after class, often when there is not much assistance available.

To address this issue, I partially flipped the class: creating videos for students to watch outside of class each week in order to use one day each week as a lab. Although designed to address a very specific need, I believe the lessons

learned form this experience can be valuable for moving more lecture time out of other CS1 courses, including those that have a lab already. In this experience paper, I describe the approach that I took, including the reasons I refrained from fully flipping the course, the advantages this approach had, as well as the limitations, and other lessons I learned along the way from the two semesters that I have taught using these videos.

Related Work

There is significant research into the pedagogy of flipping classrooms, including the use of pre-lecture video in those classrooms. Research primarily highlights introductory courses such as this one that are flipped, both in computer science and in other fields [3].

Examples of other experience papers on flipping a classroom include Largent, who flipped a CS0 course using lecture video outside of class and focusing on various active learning activities, particularly clickers in the classroom [5]. Campbell, et al. flipped a CS1 course using video lectures [2]. Fryling, et al. compares using a traditional model, a fully flipped model, and a half flip model for the same introductory programming course [1].

Flipping is often, though not always, used in large lecture classes [3]. For example, Campbell, et als class size was over 300 [2]. The ability to incorporate active learning techniques or small group collaboration is particularly valuable in large lectures where it might not be possible without an intentional design to permit it. However, there is also research into the experience of flipping smaller classrooms and the value that can be gained by doing soSarawagi, for example, described flipping a CS0 class with under 30 students [7].

There are important considerations about how flipping a course will impact inclusion, particularly of students with a variety of disabilities, including physical and learning disabilities. However, not a great deal of research has gone into ensuring that a flipped classroom remains or becomes inclusive. Talbert [8] and Milman [6] present opposing viewpoints about whether flipping a classroom increases the students control over the stream of information, thus benefitting students with learning disabilities [8] or removes beneficial scaffolding activities and the opportunities for questions, making it detrimental to those students [6]. In a previous paper, I address some of the questions about inclusion and makes recommendations based on the experience of creating videos to flip a classroom [4].

Course Description

The CS1 course is a three credit-hour class, taught in three 50-minute sessions each week. For many students, it is their first experience with computer programming. I often teach this course in parallel with another instructor, using similar weekly assignments and the same tests. In both years that I used videos, my sections had an enrollment of 24 students, which is typical for the course.

Examples of other experience papers on flipping a classroom include Largent, who flipped a CS0 course using lecture video outside of class and focusing on various active learning activities, particularly clickers in the classroom [5]. Campbell, et al. flipped a CS1 course using video lectures [2] . Fryling, et al. compares using a traditional model, a fully flipped model, and a half flip model for the same introductory programming course [1].

Course Pedagogy

In the five years before flipping the course, I have taught the course five times, always with a more traditional lecture format, and almost always in parallel with another instructor who taught another section at the same time. The content and format of the course has remained relatively static during that time.

Structure of Revised Course

I created approximately 20 minutes of video for each week, typically broken into two videos of 10 minutes each. Videos were posted well in advance of the week, along with associated material such as PowerPoint slides and written code.

Students were expected to have completed watching each weeks videos by the start of the week, typically for Mondays class although the due date was pushed back one class period if there was an assignment due on Monday. That day would have a lab, which was often, but not always, a modest programming exercise.

The second time that I taught the course, I changed the mechanism for ensuring that students watched the videos. The first semester, I used short, unannounced quizzes several times during the semester at the start of the lab day, which covered material from that weeks videos. In the second semester, I instead required students to complete worksheets while they watched the videos; students submitted the worksheets for credit. I discuss the reason for the change and the experiences from both semesters later in this paper.

On the other days in class, I used a more traditional lecture format. A fully flipped classroom would have included more videos outside of class, and incorporated completing the assignments during class. I decided against that for a variety of reasons: wanting to encourage group work in the classroom but still having individual assignments, wanting to incorporate lectures that could include more student involvement and answer student questions, and not being able to have a teaching assistant for the class time. As a department we also had concerns that fully flipping one section would create too large of a difference between the two CS1 sections, since they are often run in parallel. This was a particular concern since, during some semesters, students do not choose which section they go intothe two sections meet at the same time and students are assigned by the department to keep a balance of majors and non-majors, class levels, male/female, etc. between the two sections.

Video Content

I created the videos during the summer before the first semester I used them in the course. Each video focused on one specific topic and averaged about 10 minutes in length. For larger topics, I tried to find a natural way to split the topic into two rather than producing longer videos. Videos were always of the screen, recorded using Camtasia, and alternated between PowerPoints (using many custom animations to visualize the topic) and programming in an IDE. Videos were created with an eye towards students watching them on phones and other small screens, which meant zooming in and panning around a screen to write code and demonstrate a running program.

The videos did not cover all the topics for a particular week, but they typically taught the majority of that weeks content. During the days of typical lecture coverage, I would cover any additional topics for that week, as well as delve further into the topics covered by the videos. These were often spurred by questions that students had from the videos or the in-class lab.

Each 10-minute video typically required between four to six hours to plan, prepare, record and edit, and then to produce the associated content. At the time that I created each video, I would write the lab as well as the associated quiz if there was to be one.

Observations / Lessons Learned

Student Feedback and learning

Based on informal feedback from students, students generally liked the video format. In particular, students praised the ability to go back later and watch

the videos as they were completing an assignment. Student engagement in the labs was strong, and as I note in the next section, students appeared to have an increased investment in understanding the solution to the lab as compared to the short exercises in class from previous semesters.

Although I did not formally study student learning and compare it to previous years, I believe that student learning was improved by using this approach. In particular, it allowed students to wrestle with challenging concepts while doing the lab during class time instead of struggling with it while they were completing the assignment away from as many resources. Because assignments were similar to those in previous semesters, students still applied the material in the same way out of class, but seemed to have fewer questions when doing so because of the additional practice time in class.

Organization of Class Time

Inherently, the modified class structure gave students more time to practice writing code. Of particular help was the fact that an instructor was nearby during this time, so students benefited by being able to get help when they ran into problems.

Although originally planned to take one-third of each weeks class time, the videos and labs became the focus of many weeks. During each lab, I answered student questions individually, and would often take a few minutes during the lab to bring the class together and discuss a concept that several students were struggling with, or demonstrate a piece of code. However, students regularly requested that we discuss questions from the lab in the next class, and that class often involved walking through a solution to the lab exercise, with students asking questions throughout that process. Although not my original design, I felt that this process was particularly helpful for the students. It tended to have students more engaged than when I have written code together with them in class before, since they had already spent time struggling with the material on their own. Students seemed to be much more invested in finding a correct solution, and seemed to understand that solution better.

Mechanism to Ensure Students Watched Videos

In a previous computer applications course, where I had flipped just one unit in class, I simply used the assignments to ensure that students were, in fact, watching the videos. This went poorly, resulting in students waiting until just before each assignment was due to actually watch the relevant videos. This meant they were often completely lost during the in-class exercises, and thus didnt have a meaningful opportunity to ask questions or practice using the skills in the video before they were doing the assignment on their own. This

process really undermined any value of the videos and flipped classroom. As a result, I wanted to implement a mechanism to ensure that students watched the videos before the in-class labs.

The first time I taught the CS 1 course using the videos, I decided to use short, unannounced quizzes at the start of some labs. The goal of the quizzes was to determine whether the students had watched the videos that were assigned for that week. I wanted the quiz to be very quick at the beginning of a lab-based class period to allow the vast majority of class time to be spent on the lab.

As a result of this philosophy, I did not allow students to use physical notes during the quiz. I felt it would have been very difficult to create questions that a student who watched the video could answer very quickly, but that a student who had not watched the video could not answer, if I allowed students to bring the printed code from the video (which I provided for students since I felt it was a valuable resource).

The quiz structure was generally a failure. I knew going in that it would not really gauge understanding of the video material, but just evaluate whether they had watched the videos at all. But it had other major disadvantages as well. Not allowing written notes during the quiz gave students the message that they should not take notes as they watched the videos. In addition, the format of a brief quiz at the start of class made accessibility accommodations, such as extra time, very challenging, particularly since I try to avoid making a students disability public unless they choose to do so.

The second time I taught the course, I removed the quizzes and instead gave students a worksheet to complete each week, which they submitted for credit at the start of the lab. This approach was much more successful. One major downside of a worksheet, and the reason I initially decided on a different approach, is that one student can easily copy off of another student, rather than watch the videos. However, I feel the advantages strongly outweighed this disadvantage. The worksheets allowed my evaluation to focus more on understanding the key concepts from the videos than on simply whether the student watched the videos or not. Inherently, students were writing some when watching the videos, even if they didnt take other notes. It also encouraged students to go back and re-watch the videos if they were unable to answer the worksheets questions. I received more questions about video content (and received more emails from students when there were problems with the video) than I had during the semester with the quizzes.

Accessibility

Using videos had both positive and negative impacts on accessibility of the course. The negative impact is perhaps strongest for students with hearing or

visual impairments. The negative impact on those who are hearing impaired can be mostly mitigated by adding captions to the video. This can be done without a great deal of effort using software like Camtasia, sometimes with the first draft created automatically using speech recognition. A small disadvantage is that some formats for publishing the videos do not have optional captions, meaning the captions must always be on. The negative impact on students with visual impairments is harder to offset, and specialized material may be needed for those students.

A big advantage is the ability to watch at the best pace and in the best environment for the student, which is of particular benefit for students with learning disabilities. Multiple students commented on their preference for being able to re-watch a lesson, and this was particularly true of students for whom English was not their first language.

Venue for Publishing Videos

There are multiple considerations when deciding where to publish the videos. For the first semester, I chose to publish the videos directly through Canvas, our Learning Management System, which had the advantage of integrating well with other class material. Canvass system included the ability to speed up and slow down videos, which students found helpful. I encouraged students to watch videos at least twice: once at the regular pace, and one more time with the speed increased if they felt comfortable with that.

During the second semester, I instead used Panopto to publish the videos. Panopto still integrates relatively well with Canvas, and includes the ability to control video speed, but adds some additional features as well. Panopto uses voice recognition and screen reading to index search terms in the video, allows students to record notes either for private use or that other viewers can see, and to allow the professor to see which students have watched the video, and how many times they have watched.

One limitation of using our LMS, either directly or by making the Panopto videos available through the LMS, was that students did not have access to the videos after the semester ended. It may be valuable to provide students with another platform, either for the entire semester, or just for use after the end of the semester, where they can still access the videos.

References

[1] Eric Breimer, Meg Fryling, and Robert Yoder. Full flip, half flip and no flip: Evaluation of flipping an introductory programming course. *Information Systems Education Journal*, 14(5):4, 2016.

[2] J. Campbell, D. Horton, M. Craig, and P. Gries. Evaluating an inverted CS1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 307–312, 2014.

[3] M. N. Giannakos and J. Krogstie. Reviewing the flipped classroom research: Reflections for computer science education. In *Proceedings of the Computer Science Education Research Conference*, pages 23–29, 2014.

[4] M. R. Hodges. Ensuring videos for a flipped classroom increase inclusivity. *Journal of computing Sciences in Colleges*, 34(1):81–88, 2018.

[5] D. L. Largent. Flipping a large CS0 course: an experience report about exploring the use of video, clickers and active learning. *Journal of computing Sciences in Colleges*, 29(1):84–91, 2013.

[6] N.B. Milman. What is it and how can it best be used? *Distance Learning*, 9(3):85–87, 2012.

[7] N. Sarawagi. A flipped CS0 classroom: applying bloom's taxonomy to algorithmic thinking. *Journal of computing Sciences in Colleges*, 29(5):21–28, 2014.

[8] R. Talbert. Inverting the linear algebra classroom. *Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 24(5):361–374, 2014.

Categorizing And Ameliorating Generalization-Specialization Design Mistakes in an Intermediate Programming Class*

John W. Coffey
Department of Computer Science
University of West Florida
Pensacola, FL. 32514
jcoffey@uwf.edu

Creating complex generalization-specialization relationships is challenging for students who are new to object-oriented design. The current work seeks to categorize student design mistakes in inheritance modeling involving multiple inheritance and a larger number of super and sub-classes than students routinely encounter in introductory inheritance-related code examples. Student responses to a task requiring the creation of an inheritance lattice has led to a preliminary categorization of major error types and an assessment of relative frequencies of the errors. Examples of student work that illustrate various errors are included. This article concludes with a discussion of how these examples can be used in active learning exercises.

Introduction

Within the endeavor of teaching and learning object oriented software design, understanding inheritance and generalization-specialization relationships remains a challenge for many beginning students. The problem is further complicated because students must develop an understanding of inheritance in the context of related concepts including polymorphism, dynamic binding, and in

some programming languages, a mechanism akin to the interface mechanism provided by Java.

Much of the pedagogical work on object modeling for inheritance described in the literature is broad and encompasses the various associated topics just mentioned. The issue of sequencing these topics remains unresolved as evidenced by different topic orders in currently popular textbooks. For instance, in [1], Horstmann discusses interfaces and polymorphism before inheritance and in [2] he presents polymorphism in the context of an inheritance discussion that precedes interfaces. Other authors including Savitch [7] have created texts with different orderings of these topics.

Inheritance and polymorphic programming examples provided in textbooks usually include only a few classes and single inheritance. Experience in teaching an intermediate level object oriented programming class revealed that students struggle with more complex generalization specialization relationships. Specifically, in the current study, students had to create a lattice of relationships for a problem involving eight classes, three levels of generalization-specialization, and multiple inheritance. An analysis of results revealed several recurring, characteristic categories of errors.

The remainder of this paper contains a description of various characteristics of the problem of teaching and learning inheritance. After a review of relevant literature, the current study is presented. An attempt is made to characterize the types of mistakes learners make in constructing hierarchies or lattices involving more than minimal arrangements of classes. Several examples that are illustrative of the types of errors students made are presented. The paper concludes with a strategy pertaining to the use of these examples as teaching materials in an active learning scenario to help students gain a broader and deeper understanding of generalization-specialization relationships.

Literature on Teaching/Learning Inheritance

Liberman, Beeri, and Kolikant [3] studied inheritance and polymorphism mistakes in K-12 teachers who were learning to teach object-oriented programming. They found that students lacked sufficient understanding of hierarchies, had troubles based upon deficient training in other aspects of OO programming, made incorrect application of analogies to everyday life, and had a tendency to think in terms of alternative, simplifed models. Liberman, Beeri, and Kolikant further concluded that learners lacked adequate understanding of the computational model underlying compilation and execution of OO programs to make good decisions regarding inheritance orderings.

Reek [6] described efforts to help first year CS students understand the difference between inheritance and inclusion (generalization/specialization and

composition). He had students implement two problems, one of which naturally lent itself to inheritance and the other to inclusion. Before each lab, students had to decide which organizational construct applied. Reek concluded that the preparatory work they performed helped students understand that inheritance was the better mechanism for implementation of one of the first problems and that inclusion was the better choice for the other problem. Without the preparatory work, students struggled to decide whether to use hierarchy or composition.

Mascarell [4] presented several visual representations for data structures, algorithms, and scoping and access rules for object-oriented programming for inheritance. She utilized a diagram that is similar to a UML class diagram that contained both inheritance and inclusion. She provided an example of class Car that extended class Vehicle and includes an engine attribute. She did not present data regarding student learning outcomes, but she concluded with the notion of "one concept, one drawing", a compelling idea of providing example diagrams creating strong intuitions for the nature of the construct being studied.

Or-Bach and Lavy [5] had third year CS students create class diagrams, with a format of their own choosing, requiring the definition of an abstract class with two concrete classes, and inheritance and polymorphism mechanisms required. Out of 33 participating students, four generated the exact solution the instructors visualized, six students omitted the required abstract base class, and numerous students included additional, unnecessary classes. Of the unnecessary classes, some were irrelevant, and some might have been included, but more properly would have been included as attributes.

Schmolitzky [8] describes dilemmas in deciding how to present inheritance and polymorphism - whether to teach reuse by extension or polymorphism first and how to sequence teaching the Java interface construct. He decided upon a sequence of teaching interfaces first, the concept of subtyping (including substitutability of a subtype where supertypes are expected) and dynamic binding next, and inheritance last. As described earlier in this article, the issue of how to sequence these topics has manifest as different topic sequences in textbooks and represents what appears to be an ongoing challenge in the design of curricula pertaining to inheritance/polymorphism/interface.

The Current Study

The current study represents an attempt to examine errors intermediate programming students make in trying to create generalization/specialization relationships for relatively complex scenarios. Students were required to create an inheritance lattice for eight classes. The following sections contain descriptions

of the course, the methods employed in the study and the study's results.

The Course

The course in which the current study was performed was the second in a three-course series on objectoriented programming with Java. The first course followed an objectsfirst approach and provided background in variables, control constructs, the use of predefined classes, creating userdefined classes, and basic containers. The current course continued with inheritance, polymorphism, interfaces, exceptions, file I/O, database and recursion.

Methods

Students in the study were introduced to the vocabulary pertaining to object oriented programming and generalization-specialization relationships including class, attribute, method, and the notions of single and multiple inheritance. They examined several examples of class hierarchies, and performed an interactive activity of defining a class hierarchy. They were also required to produce class hierarchies for a programming project involving inheritance. In order to assess their ability to produce correct inheritance relationships, they had to answer the following question on the final examination:

Draw an inheritance diagram for the following classes: student, undergraduate student, professor, employee, teaching assistant, secretary, department chair, person. Note: a teaching assistant is a student who works for a professor; a department chair is a professor.

Additionally, students had to indicate if the problem required single or multiple inheritance.

Results

A total of 21 students answered the question, and of those, two produced completely correct answers. The arithmetic mean grade for all students on the question was 79.5%, surprisingly low, as the expectation was that this question would be one of the easier ones on the exam. Table 1 provides a taxonomy of errors and their frequency of occurrence.

As can be seen in Table 1, the most commonly occurring error type was failure to represent a superclass-subclass relationship. A typical example of this type of error is having `department_chair` and `professor` at the same level in the inheritance structure. Interestingly, students made this particular error even though the problem description states explicitly that a department chair `is_a` professor. More than 50% of all students made this type of error.

Table 1: The most frequent error types with frequencies.

Error Description	Occurrences
Missed superclass subclass relationship	12
Improper symbol for inheritance	6
Real-world relationship, not generalization-specialization	5
No root class specified	4
Represented other relationships than is_a	3

Almost a quarter of all students made connections based upon real-world relationships between entities rather than on generalization-specialization relationships between them. This type of error was the third most commonly occurring error. Liberman, Beeri, and Kolikant [1] noted this type of problem in their work, so the current work confirms their result. Examples of this type of error included having teaching assistants subclasses of both professors and department chairs since TAs work for both, and having secretary as a subclass of department chair because department chairs have secretaries.

Surprisingly, 20% of students failed to indicate `class Person` as the base class. Failure to use the proper symbol to indicate inheritance occurred frequently. This error is viewed as readily correctable. What follows are a few illustrative examples of typical student errors.

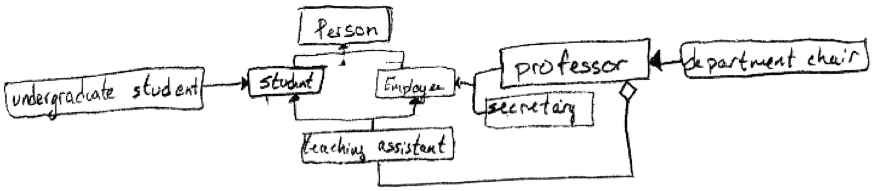


Figure 1: Inclusion of a composition relationship in an inheritance diagram

Figure 1 was created by one of the highest performing students in the class. It is diffcult to explain why, in an inheritance diagram, the student utilized the symbol for composition, which features importantly in class diagrams. It is of note that the composition relationship the student indicated is an example of a real-world relationship between the two entities rather than a generalization-specialization relationship between them. Without the composition relationship, that diagram constitutes the answer to the question that the instructor

sought. This example provides an interesting case to present to students indicating that they should use the symbology that is relevant to specific diagram types, and not introduce inappropriate symbols into a diagram.

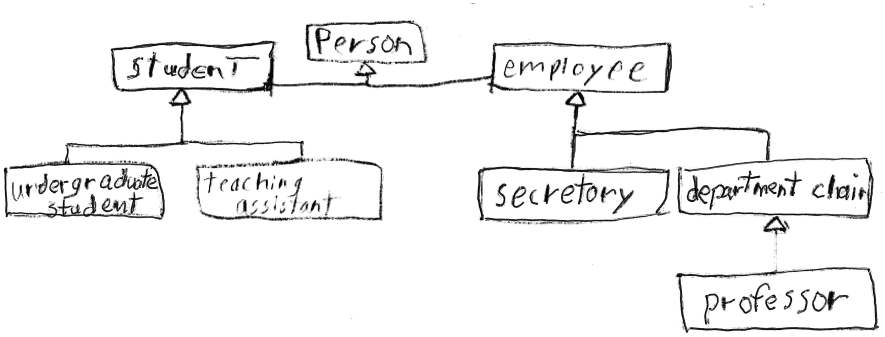


Figure 2: Reversing a generalization-specialization relationship and missing an `is_a` relationship.

In Figure 2, the student made two errors: as specified in the problem description, a teaching assistant is both a student and an employee. Interestingly, in a separate part of the same question, the student whose work is shown in Figure 2 correctly indicated that the solution to the problem required multiple inheritance, even though none appeared in his diagram. Additionally, the fact that a department chair `is_a` professor was stated explicitly in the problem description, yet the relationship between them was reversed. This relationship was judged to be a real-world relationship mistake as professors are seen as working for department chairs. The characterization of a teaching assistant as a student rather than an undergraduate student was correct.

Figure 3 contains another example of modeling real-world relationships rather than generalization-specialization relationships. Teaching Assistant should have appeared as a subclass of both Student and Employee. Instead, teaching assistant is represented as a subclass of department chair and professor, presumably because teaching assistants work for both. This answer also failed to account for the generalization-specialization relationship between professor and department chair. The `is_a` relationship between professor and department chair was stated explicitly in the question.

The work presented in Figure 4 is interesting because the student correctly indicated the relatively more difficult multiple inheritance relationship of a teaching assistant, but missed the relatively simple base `class Person`. The student also failed to use the inheritance symbology that should have featured a

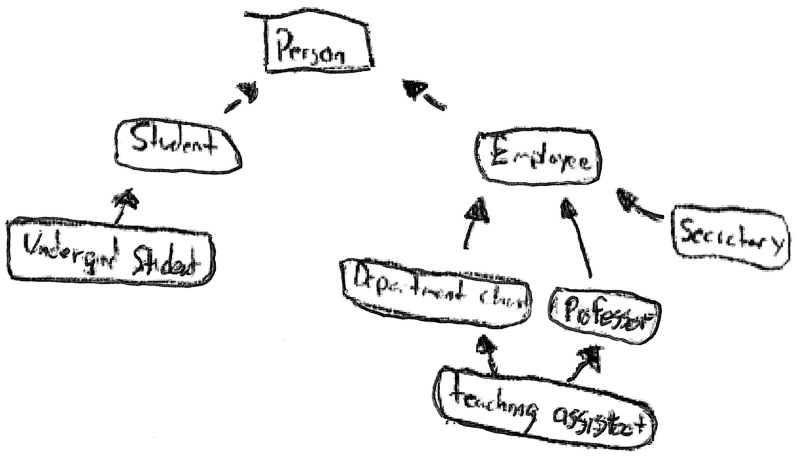


Figure 3: An error modeling real-world relationships rather than generalization-specialization

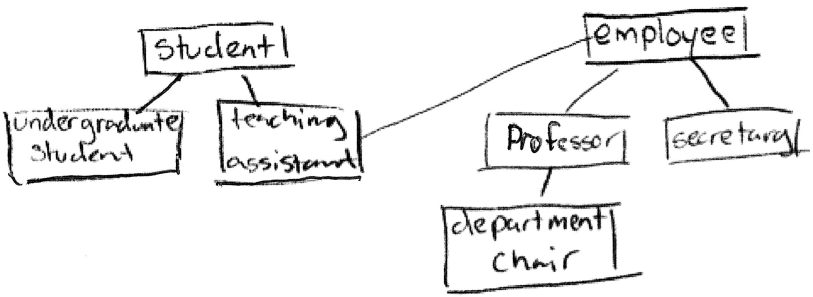


Figure 4: Missing the base class, textttis_a symbology, and inheritance.

triangle on the connector pointing to the superclass. Interestingly, this student correctly dealt with several of the more diffcult aspects of the problem, only to miss two of the simpler parts of the problem.

The final example, presented in Figure 5, was the lowest quality individual

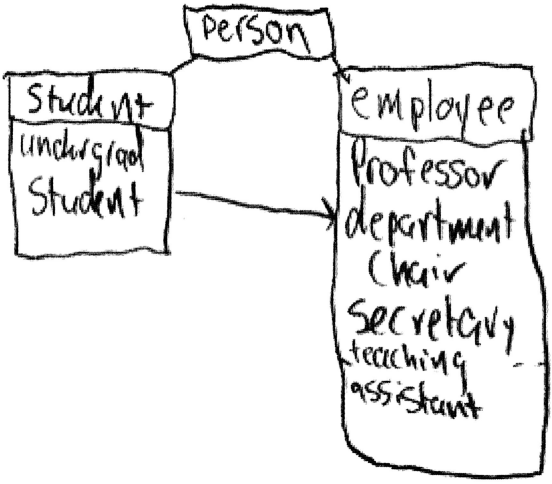


Figure 5: Numerous errors: rolling many subclasses into one, incorrect symbology for inheritance, etc.

answer delivered in the class, and it was an outlier from the rest. This student got the top-level relationships (person, student, and employee) correct, but after that, not much else. The partitioning of the rectangles into two or three areas is similar to the three areas in a node representing a class in a class diagram class name at the top, then boxes for attributes and methods. It is likely the student was confusing the type of diagram to draw. It is diffcult to see any pattern in the various classes in the employee box other than as a status ranking Professor above chair, which is in turn above secretary and then teaching assistant. It is also inexplicable why a directed link was made from the box containing `class student` to the box containing `class employee`.

Discussion

Overall, the performance of the students was acceptable, but not up to expectations for the problem. Confusing real-world relationships with generalization-specialization is the most concerning error students made and one that must be addressed. Failing to designate a root class was another important error students made. A different version of this question (that was used in previous

course offerings) omitted mention of the need for `class Person` and tasked students with creating a hierarchy, furnishing additional classes as needed. Students have struggled with that question even more than with the question posed here.

The erroneous examples presented here comprise a range of errors. The diagrams presented can be utilized as examples in an active learning activity on object-oriented design. In the next offering of the course, students will be given an interactive exercise in which they will read the problem and then identify each of the diagram presented above as correct or having errors. If the students state that the diagram contains errors, they will be required to state the nature of the error and to propose a revised diagram that corrects the error. This exercise is envisioned as a small group exercise with two or three students per group.

Also, as Reek [5] stated, it is important that students have a firm foundation in basic `is_a` and `part_of` relationships if they are to have any hope of mastering the other issues surrounding inheritance, including polymorphism and interfaces. The proposed exercise can be extended by adding other items that might better be represented as attributes, in order to help students gain proficiency at determining if noun-like entities should become classes or attributes.

Conclusions

Motivation for the current work came from the fact that students had an unexpectedly difficult time dealing with what was supposed to be a relatively simple exam question. It is demonstrable that the current task involved more classes than students had been tasked with organizing in their prior experiences. However, the most common error was failure to indicate a basic generalization-specialization relationship. As documented in the literature, the tendency to create an association based upon a real-world relationship rather than on a generalization/specialization relationship was also noted frequently. It is recognized that more time should be spent learning about larger class hierarchies and lattices, and that the error examples presented here (and others) can be used as supplementary teaching materials for this important topic.

References

[1] C. Horstmann. *Big Java, 4th edition.* 2009.

[2] C. Horstmann. *Big Java, Early Objects, 5th edition.* John Wiley and Sons, Inc., 2013.

[3] Beeri C. Liberman, N. and Y. Kolikant. Difficulties in learning inheritance and polymorphism. *ACM Trans. Comput. Educ.*, 11, 1, Article 4(10):891–921, 2015.

[4] J. B. Mascarell. Visual help to learn programming. *ACM Inroads*, 11, 1, Article 4(2(4)):42–48, 2011.

[5] R Or-Bach and I. Lavy. Cognitive activities of abstraction in object orientation: An empirical study. *ACM Inroads - The SIGCSE Bulletin*, 36(2):82–86, 2004.

[6] K. A. Reek. Teaching inheritance versus inclusion to first year computer science students. In *Proceedings of SIGCES96, the ACM Special Interest Group on Computer Science, Philadelphia, PA.*, 1996.

[7] W. Savitch. *Absolute Java, 5th Edition.* Pearson Education, Inc.

[8] Axel Schmolitzky. Teaching inheritance concepts with Java. In *Proceedings of the 4th International Symposium on Principles and Practice of Programming in Java*, PPPJ '06, pages 203–207, New York, NY, USA, 2006. ACM.

Scoring Matrix Combined With Machine Learning For Heterogeneously Structured Entity Resolution*

Xinming Li, John R. Talburt, Ting Li, Xiangwen Liu
Information Science Department
University of Arkansas at Little Rock
Little Rock, AR 72204
{xxli3, jrtalburt, txli1, xxliu10}@ualr.edu

Abstract

This paper describes how machine learning works with "coring matrix", which is designed for measuring the similarity between heterogeneously structured references, to get a better performance in Entity Resolution (ER). In the scoring matrix, each entity reference is tokenized and all pairs of tokens between the references are scored by a similarity scoring function such as the Levenshtein edit distance. In so doing, a similarity score vector can measure the similarity between references. With the similarity score vector, machine learning is used to make the linking decision. Our experiments show that machine learning based on score vector outperforms TF-IDF and FuzzyWuzzy benchmarks. One possible explanation is that a similarity score vector conveys much more information than a single similarity score. Random forest and neural network even get better performance with raw score vector input than with the statistic characteristic input.

Introduction

Entity resolution (ER) is the process of determining whether two references to real world objects in an information system are referring to the same object,

or to different objects [8]. Generally speaking, an ER system takes entity references and makes linking decisions based on the degree to which the values of the attributes of two references are similar [4]. If the two references are similar enough, the ER system will link them; otherwise, not link. The two most common types of linking rules are Boolean rules and scoring rules. The Boolean rules follow the IF-THEN logic of Boolean expressions and produce a True (link) or False (no link) decision. On the other hand, scoring rules follow the FellegiSunter model of probabilistic weights and yield a numerical score representing the relative similarity between two references [3]. Extensive research has been focused on both methods. Using XML scripts to define Boolean rules [12], blocking using user-defined inverted indices [13], calculation of weights for the scoring rule [9], Multi-valued attributes comparison [6], blocking strategy for the scoring rule for very-large datasets [10], and evolving rule logic [11]. Very recently, machine learning and deep learning is also applied in entity resolution field [2, 7]. In our big data era, entity references tend to be from multiple sources, and thus references might follow different structures at the attribute level. In this case, the classic Boolean rules and scoring rules cannot be applied without data standardization. Therefore, it is necessary to design a method specifically for the heterogeneously structured references. Overall, this research makes two main contributions. First, it designs a similarity measurement tool for the heterogeneously structured reference in ER; besides no need to standardize heterogeneously structured references to a unified attribute structure, another advantage of a scoring matrix is using a vector rather than a single value to represent the references similarity. Second, this research shows that machine learning could make the best use of similarity information conveyed by the score vector, thus making a much better linking decision. Although normalizing, such as averaging, the score vector into an aggregated score which is then compared to a threshold value to make the linking decision is intuitive and easy to understand, the normalization process actually discards some information since a vector conveys much more information than a single value. This point is proved by the result that a scoring matrix combining with machine learning outperforms the benchmark, which are using classic TFIDF to represent references first and then use cosine similarity to measure reference similarity.

Problem Statement

The key problem this research addresses is heterogeneously structured references. When the references are heterogeneously structured at the attribute level, the existing Boolean matching rules and scoring rules cannot be directly applied unless the references are standardized with the same attribute struc-

ture. For example, if the ER system is designed to compare peoples first names and last names as separate fields, then a reference source where both names are in a single field must be preprocessed and reformatted to separate and properly classify the name words. However, this data standardization process can take substantial analysis and processing effort. Moreover, it should be done by an analyst with sufficient domain knowledge and understanding of the source data. In addition, from the ER systems perspective, the standardization pre-process creates a dependency between the standardization and the ER process itself. Changes to the preprocess can potentially cause the ER engine to produce significantly different results even though the input references and matching rules are unchanged. This preprocess dependency increases the complexity and scope of versioning and change control over the entire ER process to assure consistent ER results. The practical problem of this research is a very general one. Multiplesource of the same information is regarded as the first among the ten root conditions of data quality problem [5]. The problem of performing ER on sources with heterogeneous formats is becoming even more acute as organizations begin to follow the new paradigm of "structure-on-read" rather than structure-on-write underpinning the "data lake" strategy. Therefore, the purpose of this research is to explore methods that can operate effectively without pre-process standardization and directly ingest heterogeneously structured reference sources into the ER process. The next section explains the scoring matrix design to solve this problem.

Similarity Measurement Tool: Scoring Matrix

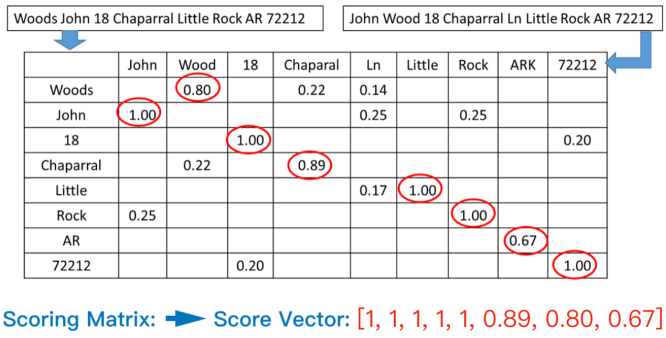
In ER, the decisions to link or not link two references is based on the degree of similarity between the references. The basic assumption is that the more similar two references are, the more likely they refer to the same real-world entity, i.e. are linked. The similarity functions are used to make the similarity decision. In the scoring matrix design, we adopt one of the most commonly used similarity function called LED (Levenshtein Edit Distance). It measures the similarity between two strings by the minimum number of operations that must be performed to transform one string to another; the operations could be character deletions, insertions, or substitutions. For example, the LED between "Jim" and "James" is three in that at least three operations are needed to transform "Ji" to "James". They are substituting "i" with "a", and then adding "e" and "s". In order to get a normalized similarity value between zero and one, in which one represents the exact match and less than one represents proportionally less degree of similarity, we adopt the normalized LED.

The core concept of our scoring matrix method is the matrix comparator, with which two references can be compared in a matrix through tokenizing

the whole reference as a string. Figure 1 illustrates the logic of scoring matrix. The reference being compared is an unstructured combination of "name" and "address" fields. The first reference is "Woods John 18 Chaparral Little Rock AR 72212"; the second reference is "John Wood 18 Chaparral Ln Little Rock ARK 72212". With the scoring matrix method, these two references are tokenized into the white-space and punctuation delimited substrings (tokens). The tokens are used as labels for rows and columns of a matrix. Each cell of the matrix contains a value representing the similarity between the tokens labeling the row and column of the cell. The similarities are given as the normalized LED between the two string as mentioned above. For visual clarity, cells with a similarity of 0.00 are left blank. The basic scheme is to find the highest similarity score (best match) for each token and then average the highest scores to get an overall score. Finding the highest similarity scores is realized by an iteration as the following pseudo shows.

```
Initialize an array r
While matrix M is not empty Do
  Find the maximal value of matrix M
  s := maxM0[i][j]
  Add s to r
  Delete row i of M
  Delete column j of M
End
```

With the above scoring matrix design, given any pair of references, no matter what the structure is, a score vector will be derived to represent the similarity between the two references.



	John	Wood	18	Chaparal	Ln	Little	Rock	ARK	72212
Woods		0.80		0.22	0.14				
John	1.00				0.25		0.25		
18			1.00						0.20
Chaparral		0.22		0.89					
Little					0.17	1.00			
Rock	0.25						1.00		
AR								0.67	
72212			0.20						1.00

Figure 1: Matrix Comparator Example

Machine Learning Based On The Score Vector

Although normalizing, such as averaging, the score vector into an aggregated score which is then compared to a threshold value to make the linking decision is intuitive and easy to understand, the normalizing process actually discards some information since a vector conveys much more information than a single value. Machine learning takes multiple dimension numerical features as input. However, the problem to apply machine learning is that the length of the score vector is not identical, depending on the number of tokens of two compared references. Therefore, we need to fix the length of score vector before applying machine learning.

Data Processing: To fix the length of input, two methods are used. The first is to use the statistic characteristics of the score vector. They are the mean of score values, the variance of the score values, and the length of score vector. In other words, we use mean, variance and length as three features in machine learning; this method is referred as feature data input in this paper. The second method is to pad the raw score vector to identical length that is the longest of all the score vector, i.e., that is 12 in the experiment dataset. This method is referred as raw data input.

Dataset: There are 499,500 samples to train and test machine learning model. These samples are derived from all the combination of two references in a 1,000 references dataset. Each sample has one score vector derived from the scoring matrix and one label 0 or 1 showing the reference pair refers to the same person or not.

Experiment Design: As the linking decision could be seen as a binary decision, 0 for not link and 1 for link, the classification machine learning algorithms can be directly applied after the data processing. We start with logistic regression to test the effectiveness of scoring matrix, using two alternative similarity measurements as benchmark. The first benchmark uses TF-IDF to represent references and then use cosine similarity to measure references similarity. The second benchmark directly uses existing FuzzyWuzzy similarity measurement [1]. Then we investigate other machine learning algorithms, including support vector machine (SVM), decision tree, random forest, and neural network. Finally, we further test the raw data input.

Experiment 1: Effectiveness of Scoring Matrix

The results show that scoring matrix outperforms two benchmarks significantly. Scoring matrix performs very well on recall. One possible explanation is that scoring matrix use a vector rather than a single value to make link decision, and a vector could provide more information than a single value to make linking decision. In other words, score vector renders machine learning extra ability

Table 1: Results of experiment 1

	Precision	Recall	F-measure
TF-IDF	0.84	0.49	0.62
FuzzyWuzzy	0.84	0.42	0.56
Scoring Matrix	0.81	0.68	0.74

to recognize linking pattern.

Experiment 2: Explore Machine Learning Algorithm

Table 2: Results of experiment 2

	Precision	Recall	F-measure
SVM	0.78	0.74	0.76
Decision Tree	0.85	0.70	0.77
Random Forest	0.83	0.75	0.79
Neural Network	0.77	0.75	0.76

This experiment shows that the performance could be further enhanced by exploring other machine leaning algorithms. For this dataset, the random forest algorithm performs best.

Experiment 3: Raw Score Vector Input

Table 3: Results of experiment 3

	Precision	Recall	F-measure
Logical Regression	0.83	0.67	0.74
SVM	0.82	0.69	0.75
Decision Tree	0.81	0.72	0.77
Random Forest	0.86	0.78	0.82
Neural Network	0.83	0.73	0.78

These results show that machine learning achieves almost equal performance with the raw data input and with the feature data input. Especially, the the random forest and neural network even get the better performance with raw data input than with feature data input (Table 2). One possible explanation is that, compared to other machine learning algorithms, random forest and neural network could extract more information from the raw score vector.

CONCLUSIONS

In the big data era, entity references from multiple sources are very common; and references from different sources would be heterogeneously structured. Facing this practical problem in entity resolution, this research proposes the method of scoring matrix plus machine learning, in which a scoring matrix is the tool to measure the similarity between heterogeneously structured references and machine learning is used to make the linking decision based on the score vector derived from scorning matrix. The experiments show that the combination of scoring matrix and machine learning could boost the performance significantly compared to the benchmarks. For future research, the scoring matrix design could consider weights for the tokens during similarity calculation.

References

[1] FuzzyWuzzy Using Python. `https://www.neudesic.com/blog/fuzzywuzzy-using-python` retrieved November 15, 2018.

[2] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467, 2018.

[3] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[4] F. Kobayashi and J. R. Talburt. Improving the quality of entity resolution for school enrollment data through affinity scores. In *Proceedings ofă19th MIT International Conference on Information Quality*, pages 69–80, 2014.

[5] Yang W Lee, Leo L Pipino, James D Funk, and Richard Y Wang. *Journey to data quality*. The MIT Press, 2009.

[6] Pablo N Mazzucchi-Augel and Héctor G Ceballos. An alignment comparator for entity resolution with multi-valued attributes. In *Mexican International Conference on Artificial Intelligence*, pages 272–284. Springer, 2014.

[7] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[8] J. R. Talburt. *Entity resolution and information quality*. 2011.

[9] P. Wang, D. Pullen, N. Wu, and J. R. Talburt. Iterative approach to weight calculation in probabilistic entity resolution. In *2014 19th International Conference on Information Quality (ICIQ-19)*.

[10] Pei Wang, Daniel Pullen, John R Talburt, and Cheng Chen. A method for match key blocking in probabilistic matching. In *Information Technology: New Generations*, pages 847–857. Springer, 2016.

[11] Steven Euijong Whang and Hector Garcia-Molina. Entity resolution with evolving rules. *Proceedings of the VLDB Endowment*, 3(1-2):1326–1337, 2010.

[12] Yinle Zhou, John R Talburt, Fumiko Kobayashi, and Eric D Nelson. Implementing boolean matching rules in an entity resolution system using xml scripts. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer , 2012.

[13] Yinle Zhou, John R Talburt, and Eric Nelson. User-defined inverted index in boolean, rule-based entity resolution systems. In *2013 10th International Conference on Information Technology: New Generations*, pages 608–612. IEEE, 2013.

Digital Distraction Outside the Classroom: An Empirical Study*

Rajvardhan Patil, Matt Brown, Mohamed Ibrahim, Jeanine Myers, Kristi Brown, Muhammad Khan, Rebecca Callaway
Arkansas Tech University
Russellville, AR 72801
{rpatil, hbrown11, mibrahim1, jmyers32, kspittlerbrown, mkhan3,rcallaway}@atu.edu

Abstract

The purpose of this study was to investigate the impact of the use of digital devices on students performance on assignments completed outside of the classroom across different academic disciplines. The investigators employed between subject design to examine the relationship between student performance on an assignment that was delivered in two different formats (paper or electronic) to 281 students in three different colleges within the same university (computing and engineering, mathematics, and education). The results revealed that the digital distractions significantly and negatively correlated with assignment score and thus, digital distraction corresponded with lower scores on the assignments. Furthermore, the study found that digital distractions were significantly and positively correlated with time required to complete the assignment (the longer the time students spent on phone applications and internet sites not related to the assignment, the more time they need to complete the assignment). Finally, the results of this study found that the digital distraction were significantly lower for students in a computing related major than for non-computing majors, indicating that digital distractions may not impact all students equally. This paper discusses the experimental setup, methodology, and findings of the research in detail.

1 Introduction

The digital age is also the age of information overload. Digital distraction is prevalent in virtually every environment people operate. Education is no different, where students can be negatively impacted by digital distractions. Although recent studies have identified the potential negative impacts of digital distraction on the learning process both inside the classroom and outside of the classroom, more of the literature appears to focus on digital distraction inside the classroom. However, the impact of digital distraction outside of the classroom is arguably more important, since students are expected to achieve more learning on their own in a typical post-secondary education environment. This research focuses on digital distractions during assignment completion outside of the classroom in a post-secondary environment. An experiment was conducted where college students were given homework assignments followed by a questionnaire concerning digital distraction during the assignments. More than one course was used, two different types of assignment were given, and each assignment was given on both paper and electronic format. The results indicate digital distraction negatively impacts performance on assignments.

2 Background

Digital distraction has been identified as a concern in both the workplace and educational environments. Agrawal et al. define digital distraction as "distraction due to electronic devices and media that breaks the concentration from the main piece of work that is being done" [2]. "Cyber slacking" is another term used to identify this problem [9]. Various studies have addressed digital distraction in the classroom, many finding a significant negative impact. In one study, over 70% of students using laptops in a college class spent half of their class time doing nonacademic activities on their laptops [15]. In a 2017 study, 94% of college students admitted they wanted to use their cell phones for non-class related activities [26]. In another study, 80% of students indicated they pay attention less in the classroom because of digital devices [21]. Because of studies such as these, the potential negative impact of digital distractions in the classroom is well established. Some work has also been to done to try and measure the impact of different aspects of digital distraction outside the classroom. One study found that over 40% of students observed in the library for at least 30 minutes spend time using their phone [1]. Indications that phones are a source of distraction to students outside of a classroom are further established in [4, 27]. In one example, [6], found that college students who studied over a 3-hour period while having access to their mobile devices, were pulled off-task by their mobile devices an average of 35 times. Similarly,

[25] observed that college students stayed on task for only 65% of a 15-minute study period, checking their mobile devices once every 5 minutes. These findings align with self-report studies indicating that more than 60% of college students use mobile devices for off-task purposes while doing schoolwork outside of class [12, 22]. Another study finds that as the amount of time spend on social networks increases; college GPA decreases [3]. The same result was echoed in [20, 25] where the relationship between Social Network Sites (SNS) use and academic performance was significantly negative. [19] stated that the constant digital media distractions interrupt the thought and communication processes, obstructing students ability to learn and showed a negative correlation between frequency of media use and academic performance. These results were also echoed by other researchers who have found that multitasking and digital distraction hinder academic achievement due to playing games or surfing the internet during learning [11, 17, 5, 16].

While some have argued to embrace digital devices in the learning process, other researchers have suggested that the academic benefits of students using digital devices are overstated. Regarding the benefits of social media in the classroom, two studies found no relationship between the use of digital devices in the classroom and student academic achievements [10, 23]. In addition, [8] found that student collaboration with digital devices did not improve students thinking or understanding. Further, regardless of whether digital devices are properly handled in a classroom, the potential issue of digital distraction during learning outside the classroom, where instructors have no control of the learning environment, looms large. This study contributes to the understanding of this issue by considering the impact of digital distraction on student performance on an assignment completed out of the classroom for different types of students, different types of assignments, and different assignment formats.

3 Methodology

An experiment was conducted using students in courses convenient to the researchers over a two-semester time frame. Students in each class were given a homework assignment on paper or in a digital format. Upon completion of the assignment, students electively completed a questionnaire regarding digital distraction during completion of the assignment. Students not completing the questionnaire were not included in the results of the study. The experiment was repeated with a different type of assignment in a second semester. IRB approval was obtained for this study and all ethical considerations were met. In total, 281 students in courses in computing, education, and mathematics departments participated in the study. One student only partially completed the questionaire and therefore was included in only some of the analysis (sample

sizes are noted in tables).

3.1 Students

All students were degree seeking students from a Southern Regional Educational Board (SREB) four-year category three university in the mid-south. The following statistics describe the demographics of the students participating in the study. Of the 281 participants, 17% were graduate students and 83% were undergraduate students. The majors represented included 52% computing or closely related majors (Information Systems, Information Technology, Computer Science, or Electrical Engineering), 35% education majors, 7% business majors, 4% majors from natural and health sciences, and the remaining 2% from other majors. Ages: 18-21 59%, 22-25 19%, 26-30 8%, 31-40 9%, 41 and over 5%. Gender: 42% were female; 58% were male. Students were also asked to provide their preferred learning style with the following results: Hands-on 24%, Reading 9%, Video/movies 8%, Lectures 6%, a mixture of all of these learning styles 53%.

3.2 Materials

Two different assignments were given to participating students. Regardless of the assignment, the same questionnaire was administered following completion of the assignment. The first assignment was given during the spring 2018 semester. In this assignment, students were required to read a journal article and answer questions concerning the article. The second assignment, given to a second group of students (independent of the first group), was administered during the fall 2018 semester. In the second assignment, students were required to read a brief introduction to logical thinking including propositions and truth tables with examples. The students were then asked a series of thirteen questions. Blooms taxonomy framework was used to design these questions. Except for create category, the other five hierarchies in the taxonomy were covered, where the complixity of questions increased as the students advanced through each level. In each semester, approximately half of the students were given the assignment on paper and the other half of the students were given the assignment in a digital format. Both the assignments were takehome, and students were given a day or two to complete them. Immediately after the assignments were submitted, students were given a questionnaire (the questionnaire was voluntary, but students received a small amount of bonus points for completing the questionnaire). The questionnaire contained ten questions primarily regarding their use of digital devices while completing the assignment and five demographic questions.

Table 1: Correlations between assignment grade and digital distractions

Factor	Correlation with Assignment Grade	p (test of significance)	Sample Size
Time on Internet (sites not related to assignment)	-0.17*	0.0037	280
Time on Phone Apps (not related to assignment)	-0.15*	0.0093	280
Frequency of non-assignment related Internet Use	-0.14*	0.0233	280
Frequency of non-assignment related Phone Use	-0.16*	0.0061	280
Time Spent seeking online help for the assignment	-0.06	0.3592	280
Assignment Type (0=reading, 1=logic)	-0.62*	<0.0001	280
Assignment Format (0=paper, 1=digital)	0.09	0.1276	280
Major (0=non-computing, 1=computing major)	0.51*	<0.0001	280

3.3 Project Goals (Research Questions)

The research questions considered in this study include: 1. Is student performance on an assignment significantly correlated with digital distractions, how the assignment was delivered (paper or digital), the type of assignment (reading comprehension or logical thinking), or the type of major (computing versus non-computing), or the use of digital devices to help with the assignment? 2. Is time to complete the assignment significantly correlated with digital distractions, how the assignment was delivered (paper or digital), the type of assignment (reading comprehension or logical thinking), or the type of major (computing versus non-computing), or the use of digital devices to help with the assignment? 3. Are computing students more or less likely to be distracted?

4 Results

A simple correlation analysis was used to determine what factors correlated with the grade students received on the assignment. (Note: In Tables 1, 2, 3: *denotes a significant correlation, $p<0.05$.) A simple correlation analysis was used to determine what factors correlated with the grade students received on the assignment. (Note: In Tables 1, 2, 3: *denotes a significant correlation, $p<0.05$.)

As can be seen in 1, digital distractions as measured by time on the internet,

Table 2: Correlations between assignment completion time and digital distractions

Factor	Correlation with Assignment Completion Time	p (test of significance)	Sample Size
Time on Internet (sites not related to assignment)	0.26*	<0.0001	280
Time on Phone Apps (not related to assignment)	0.20*	0.0008	280
Frequency of non-assignment related Internet Use	0.05	0.3964	280
Frequency of non-assignment related Phone Use	0.06	0.2845	280
Time Spent seeking online help for the assignment	0.15*	0.0111	280
Assignment Type (0=reading, 1=logic)	0.25*	<0.0001	280
Assignment Format (0=paper, 1=digital)	-0.01	0.8211	280
Major (0=non-computing, 1=computing major)	-0.16*	0.0062	280

time on phone apps, and frequency of phone use and internet use were all significantly negatively correlated with assignment score. Thus, digital distraction corresponded with lower scores on the assignments. The type of college major and type of assignment were correlated with assignment score. Assignment scores could not be shown to be significantly correlated with the format of the assignment, that is whether the assignment was given on paper or digitally had no impact on the assignment score. Further, time spent seeking online help on the assignment was not shown to be siginficaly correlated with assignment performance.

As can be seen in 2, digital distractions as measured by time on the internet, time on phone apps were significantly positively correlated with time required to complete the assignment. That is, as the time students spend on internet sites not related to the assignment or the time spent on phone apps not related to assignment, the time to complete the assignment increased. Also, the time spent seeking online help for the assignment was positively correlated with the time to complete the assignment. The type of college major and type of assignment were correlated with assignment completion time. Again, assignment completion time could not be shown to be significantly correlated with the format of the assignment, that is whether the assignment was given on paper or digitally had no impact on the assignment completion time.

As can be seen in 3, four of the five measures of digital distraction were significantly lower for computing students than for non-computing students.

Table 3: Correlations between major and digital distractions

Factor	Correlation with Major (non-computing=0, computing =1)	p (test of significance)	Sample Size
Time on Internet (sites not related to assignment)	-0.10	0.0859	281
Time on Phone Apps (not related to assignment)	-0.14*	0.0222	281
Frequency of non-assignment related Internet Use	-0.16*	0.0075	281
Frequency of non-assignment related Phone Use	-0.13*	0.0318	281
Time Spent seeking online help for the assignment	-0.17*	0.0036	281

Thus, there is an indication that digital distractions will not impact all types of students in the same way. It is unclear if these results are confounded by the students interest levels in the assignment. Because, although both assignments dealt with topics relevant to all majors, the assignments dealt with topics computing students would likely find more familiar, potentially making them less distracted.

5 Conclusion

Digital distractions are an issue in post-secondary education beyond just the classroom. The results of the present study indicate that the digital distractions impact both the quality and quantity of student work outside the classroom. Further, the students type of major did significantly change both the propensity to be distracted and the assignment score. In contrast, the format of the assignment (whether the assignment was given on paper or online) did not significantly differ with digital distraction or assignment performance. In addition, the use of digital resources to help complete the assignment did not significantly correlate to improved performance. The main conclusions of this study support previous findings produced in different settings and with other populations and provide empirical evidence that validates the negative effect of digital distraction on students assignment performance. For example, previous studies found that digital distraction is linked to drops in time spent studying [6, 28], homework assignment performance [7], homework completion rates [14], course grades [18, 24], and cumulative college grade-point average [13, 25]. While the negative impact of digital distraction seems clear, effective approaches of how to address the issue may be less easy to identify.

References

[1] Farhad Mohammad Afzali and Briana B. Morrison. Cellphone usage in academia: The problem and solutions. In *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '18, pages 325–328, New York, NY, USA, 2018. ACM.

[2] Priyanshi Agrawal, H. S. Sahana, and Rahul De'. Digital distraction. In *Proceedings of the 10th International Conference on Theory and Practice of Electronic Governance*, ICEGOV '17, pages 191–194, New York, NY, USA, 2017. ACM.

[3] Priti Bajpai and Maneesha. Analyzing effect of social media on academic performance of university graduates. In *Proceedings of the 2018 The 3rd International Conference on Information and Education Innovations*, ICIEI 2018, pages 40–44, New York, NY, USA, 2018. ACM.

[4] Robert E. Beasley, Jacob T. McMain, Mathew D. Millard, Dylan A. Pasley, and Matthew J. Western. The effects of college student smartphone use on academic distraction and dishonesty. *J. Comput. Sci. Coll.*, 32(1):17–26, October 2016.

[5] Saraswathi Bellur, Kristine L. Nowak, and Kyle S. Hull. Make it our time: In class multitaskers have lower academic performance. *Computers in Human Behavior*, 53:63–70, 2015.

[6] Charles Calderwood, Phillip L Ackerman, and Erin Marie Conklin. What else do college students "do" while studying? an investigation of multitasking. *Computers & Education*, 75:19–29, 2014.

[7] Charles Calderwood, Jeffrey D Green, Jennifer A Joy-Gaba, and Jaclyn M Moloney. Forecasting errors in student media multitasking during homework completion. *Computers & Education*, 94:37–48, 2016.

[8] G Falloon and Elaine Khoo. Exploring young students' talk in iPad-supported collaborative learning environments. *Computers Education*, 77:1328, 08 2014.

[9] Abraham E. Flanigan and Kenneth A. Kiewra. What college instructors can do about student cyber-slacking. *Educational Psychology Review*, 30(2):585–597, Jun 2018.

[10] Håkan Fleischer. What is our current understanding of one-to-one computer projects: A systematic narrative research review. *Educational Research Review*, 7:107122, 06 2012.

[11] Mathias Hatakka, Annika Andersson, and Åke Grönlund. Students use of one to one laptops: a capability approach analysis. *Information Technology & People*, 26(1):94–112, 2013.

[12] Wade C Jacobsen and Renata Forste. The wired generation: Academic and social outcomes of electronic media use among university students. *Cyberpsychology, Behavior, and Social Networking*, 14(5):275–280, 2011.

[13] Reynol Junco. Too much face and not enough books: The relationship between multiple indices of Facebook use and academic performance. *Computers in Human Behavior*, 28:187–198, 01 2012.

[14] Reynol Junco and Shelia R. Cotten. No A 4 U: The relationship between multitasking and academic performance. *Computers Education*, 59(2):505–514, September 2012.

[15] Robin Kay and Sharon Lauricella. Assessing laptop use in higher education: The laptop use scale. *Journal of Computing in Higher Education*, 28, 12 2015.

[16] Jeffrey H. Kuznekoff and Scott Titsworth. The impact of mobile phone usage on student learning. *Communication Education*, 62(3):233–252, 2013.

[17] Jing Lei and Yong Zhao. One-to-one computing: What does it bring to schools? *Journal of Educational Computing Research*, 39(2):97–122, 2008.

[18] Andrew Lepp, Jacob Barkley, and Aryn C. Karpinski. The relationship between cell phone use, academic performance, anxiety, and satisfaction with life in college students, 11 2014.

[19] Jean-Louis Leysens, Daniel B. le Roux, and Douglas A. Parry. Can I have your attention, please?: An empirical investigation of media multitasking during university lectures. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '16, pages 21:1–21:10, New York, NY, USA, 2016. ACM.

[20] Dong Liu, Paul Kirschner, and Aryn C Karpinski. A meta-analysis of the relationship of academic performance and social network site use among adolescents and young adults. *Computers in Human Behavior*, 77, 12 2017.

[21] Bernard Mccoy. Digital distractions in the classroom: Student classroom use of digital devices for non-class related purposes. *Journal of Media Education*, 4:5, 10 2013.

[22] Kouider Mokhtari, Julie Delello, and Carla Reichard. Connected yet distracted: Multitasking among college students. *Journal of College Reading and Learning*, 45(2):164–180, 2015.

[23] William R. Penuel. Implementation and effects of one-to-one computing initiatives: A research synthesis. *Journal of Research on Technology in Education*, 38(3):329–348, 2006.

[24] Susan Ravizza, Zach Hambrick, and Kimberly Fenn. Non-academic internet use in the classroom is negatively related to classroom learning regardless of intellectual ability. *Computers Education*, 78:109114, 09 2014.

[25] Larry Rosen, Mark Carrier, and Nancy Cheever. Author's personal copy facebook and texting made me do it: Media-induced task-switching while studying. *Computers in Human Behavior*, 29:948–958, 05 2013.

[26] Suliman S. Aljomaa, Mohammad Qudah, Ismael Bursan, Salaheldin Bakhiet, and Adel Abduljabbar. Smartphone addiction among university students in the light of some variables. *Computers in Human Behavior*, 61:155–164, 08 2016.

[27] Aaron Smith. U.S. smartphone use in 2015. `http://www.pewinternet.org/2015/04/01/us-smartphone-use-in-2015/`.

[28] Diane Wentworth and June H. Middleton. Technology use and academic performance. *Computers Education*, 78:306311, 09 2014.

Data Science Academic Programs in the U.S.*

Ismail Bile Hassan and Jigang Liu
Computer and Information Sciences and Cybersecurity
Metropolitan State University
Saint Paul, MN 55106
{ismail.bilehassan, jigang.liu}@metrostate.edu

Abstract

As the development of big data and cloud computing has taken the center stage of the technology advance for the last decade, data science programs at various levels have been established for meeting the demands from all sectors of our economy. In this project, we study data science programs offered by the U.S. institutions. We are particularly interested in regular programs at Bachelor, Master, and Ph.D. levels offered on campus. The result of our study will not only help the educators who are currently running a Data Science related program to find their counterparts at other institutions for improving and promoting their programs but also support the educators who are planning to offer a Data Science program to locate an exemplary program for drafting their curriculum and preparing their proposal. Although Data Science provides a promising field in supporting the economy growth, employment opportunity, and technology advancement, a conscious and balanced decision should be made for developing an academic program in the field based on a thorough assessment on allocated resource, faculty preparation, and enrollment potential.

1 Introduction

According to a study by Louridas and Ebert [5], approximately 1,200 exabytes of data are produced annually. This resulted the concept of Big Data which

is defined as datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze by McKinsey Global Institute (MGI) in 2011 [6]. The massive amount of unstructured data presents enormous challenges for the business and IT sectors. Data science, business intelligence, analytics, and other related fields in big data and data analytics have become increasingly vital in both academic and business communities.

Big data has been used to transform medical practice, modernize public policies, and inform business decision-making [7]. Several reports have revealed the demand for data science. Manyika et al. predict that by 2018 the U.S.A. could face a serious shortage of deep analytical skills required for as many as 190,000 positions. The U.S. will not be able to fill this gap simply by changing graduation requirements, waiting for students to graduate with more Data Science skills, or by importing the talent from overseas.

Therefore, data science and data analytics fields have been developed as to deal with the big data tsunami as well as other data related theory, technology, tools, and applications. Data Analytics involves the process of investigating data sets in order to derive conclusions on the information they contain by applying specialized systems and software.

Until now, a number of research studies have examined Data Science programs within a particular discipline, such as Business [2] or iSchool [10]. Since Data Science is a multidisciplinary field, it is essential and critical to have an overall view of the current status of the program development across disciplines so that a conscious and balanced decision could be made. In next section, an overview on Data Science is introduced. The status of an overall development of Data Science programs in the U.S. is presented in Section 3, followed by the analysis of the result in Section 4. Finally, in Section 5, our conclusion and future work is provided.

2 Data Science in a Nutshell

Data Science can be defined as being comprised of three areas: analytics, infrastructure, and data curation [4]. It is experiencing rapid and unplanned growth, spurred by the proliferation of complex and rich data in science, industry, and government. Fueled in part by reports, such as the widely cited McKinsey report [6], that forecast a need for hundreds of thousands of data science jobs in the next decade, data science programs have exploded in academics as university administrators have rushed to meet the demand [3].

The need for new and innovative tools for managing and deriving insights from big, unstructured data continues to grow. Consequently, there is a rapidly increasing demand for data scientists who know how to apply these new tools to handle big, unstructured data and to solve business problems [1]. In re-

sponse to this novel demand, several U.S.A. higher education institutions have launched blended programs such as Data Analytics, Business Intelligence, or Data Science [10]. In addition to the effort made by the academia from the U.S., the faculty members from the European Union drafted their Data Science framework through the EDISON project [4] as shown in Figure 1 below,

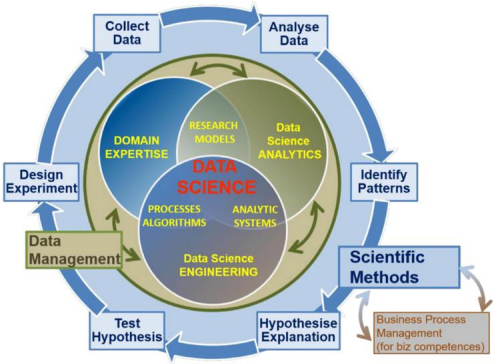


Figure 1: Data Science Competence Groups [4]

Although the proposed framework in Figure 1 is based on the structure introduced by NIST [8], the relationship and transition between the competence groups demonstrated in the framework provide a solid foundation for designing the corresponding curricula. For instance, this framework indicates the three cornerstones of Data Science in Analytics, Engineering, and Domain Expertise, which are corresponding to Statistics, Computer Science, and Domain Expertise in either Business, Biology, or Healthcare.

In addition to presenting a framework of Data Science shown Figure 1 above, the EDISON project [4] studies many aspects of Data Science in terms of sub-areas of three key components in Engineering, Analytics, and Expertise, and also analyzes the professional skills and responsibilities required for various occupations in Data Science. Since the EDISON framework also considers the ACM Information Technology Competence Model, its competences and knowledge domains outlined for Data Science provide a solid foundation for crafting a Data Science academic program.

The data science programs are multidisciplinary and a single subject domain is not enough to cover the magnitude of content and skills needed for Data Science (DS) programs. A DS program should represent a combination of subject areas from several disciplines including applied mathematics, statistics, and computer science [9].

3 Data Science Programs In The U.S.

An extensive study on the current Data Science programs in the U.S. higher education by collecting the programs through google search engine as well as some selected websites, such as `http://datascience.community/colleges`, `http://datasciencedegreeprograms.net`, and `http://www.discoverdatascience.org`. Website `http://datascience.community/colleges` lists 581 programs in data science, data analytics, and other related fields at more than 200 universities around the world. In the forms of on-campus and online, these programs include Ph.D., Master, Bachelor, and Certificate programs in the different academic levels of graduate, bachelor, and associate.

However, our study focuses on campus-based Ph.D., M.S. and B.S. Data Science programs in the U.S., thus, we filtered all the DS programs in U.S. from `http://datascience.community/colleges` by using key word "Data Science". In addition, we also checked with the search results from other websites and then verified and eliminated all duplicated search results as well. Although DS programs can be introduced as graduate, graduate certificate, undergraduate, certificates and minors, we are only interested in studying the regular programs at B.S., M.S., and Ph.D. levels in this project. Based on the discussion above, Data Analytics, Data Mining, and Big Data are considered as sub-areas of Data Science. Therefore, all programs in those sub-areas are not included in this study.

To have a coherent view of the distribution of DS programs among the institutions in terms of research funding, curriculum spectrum, available resources, and enrollment capacity, we adopt the Carnegie Classification, which is "the leading framework for recognizing and describing institutional diversity in U.S. higher education for the past four and a half decades" at `http://carnegieclassifications.iu.edu`, maintained by Indiana University. The first publication of the classification was in 1973 and the one used for this paper was published in 2015 and updated in 2017. Table 1 shows the definitions of the classifications that are used to classify the institutions of higher education in the U.S. We coded with a letter with a number to each class in order to simplify the data presentation and discussion. Letters R, M, and B represent"Research," "Master," and "Bachelor," respectively.

Instead of listing all the institutions along with their programs, we present an aggregated result of our findings based on the Carnegie classifications and corresponding programs offered by each of the categories of institutions. As Table 2 shown below, the distributions of the Data Science programs in B.S., M.S. and Ph.D. along with each of 10 Carnegie classifications listed in Table 1 is well illustrated.

It should be stated that Table 2 only shows the numbers of the programs, not the number of the institutions. In other words, a university can offer all

Table 1: The Carnegie Classification and Definition

Category	BASIC2015	Definition from Carnegie Higher Education Classification (BASIC 2015)
R1	15	Doctoral Universities: Highest Research Activity
R2	16	Doctoral Universities: Higher Research Activity
R3	17	Doctoral Universities: Moderate Research Activity
M1	18	Master's Colleges & Universities: Larger Programs
M2	19	Master's Colleges & Universities: Medium Programs
M3	20	Master's Colleges & Universities: Small Programs
B1	21	Baccalaureate Colleges: Arts & Sciences Focus
B2	22	Baccalaureate Colleges: Diverse Fields
B3	23	Baccalaureate/Associate's Colleges: Mixed Baccalaureate/Associate's
B4	24 - 33	Other Special-Focus Four-Year/Tribal Colleges

Table 2: Data Science Academic Programs by the Carnegie Classifications

Program	Total	Category									
		R1	R2	R3	M1	M2	M3	B1	B2	B3	B4
Ph. D.	13	6	4	1	1	0	1	0	0	0	0
M.S.	45	22	8	4	7	2	0	2	0	0	0
B.S.	49	13	7	5	11	4	1	4	4	0	1
Total	107	41	19	10	19	6	2	6	4	0	1

three levels of the programs in B.S., M.S., and Ph.D. We also realized that 2 Ph.D. programs in Data Science are offered by non-research institutes.

4 Discussion And Recommendation

As indicated in Table 2 in section 3, there are 107 campus-based Data Science programs at either Ph.D., M.S. or B.S. level. With regarding to the number of the programs, 38% of them are offered by R1 universities/colleges while 27% of them are offered by R2 and R3 universities/colleges combined. In other words, out of 107 campus-based programs in Data Science, 70, or 65% of them, are offered by Doctoral universities/colleges. Without considering 13 Ph.D. programs (107 13 = 94), only 27 programs, or 29% of them, are offered by Masters universities/colleges. Out of those 27 programs offered by Masters institutions, 18, or 70% of them, are offered by the universities/colleges under the classification of M1. Baccalaureate Colleges contribute the smallest potion of the programs in either M.S. or B.S., with 11, which is about 11% of the total number of M.S. and B.S.programs.

To understand who are running those programs in terms of hosting department or unit, we further divide our findings based on their hosting departments.

Table 3 highlights the hosting departments for those 13 Ph.D. programs in Data Science. Although there is no a dominated department shown in the table, 10 out of 13 Ph.D. programs, or 77%, are administrated by non-computing related departments, while 8 out of 13, or 62%, are hosted in Math/Statistics/Data Science related departments or centers.

Table 3: On Campus Ph.D. Programs in Data Science

Data Science Program	Total	%
Total Programs	13	100
Computing and Informatics	3	23.1
Math	1	7.7
Statistics	2	15.4
Arts and Science	1	7.7
Data Science	2	15.4
Independent Data Science Centers	3	23.1
Interdisciplinary	1	7.7

As the layout of Table 3, Table 4 below gives the distribution of the hosting department for those M.S. campus-based programs. Not like the Ph.D. programs, there are two dominated departments in Computer Science and Arts and Sciences with 10 and 9, or 22% and 20%, respectively. It is also noticed that there are 4 programs offered by "Data Science"department, which means some institutions have moved faster than others by updating their department structure.

Table 4: Masters and Bachelor's Programs in Data Science

Data Science MS Programs	Total	%
Computer Science	10	22.2
Computing and Informatics	2	4.4
Math and Computer Science	2	4.4
Math and statistics	1	2.2
Statistics	2	4.4
CS and EE	1	2.2
Arts and Sciences	9	20
Data Science	4	8.9
Business	1	2.2
Graduate School	4	8.9
Engineering	2	4.4
Independent DS Centers	2	4.4
Interdisciplinary	4	8.9
Professional Studies	1	2.2
Total Programs	45	100

Data Science BS Programs	Total	%
Computer Science	10	20.4
Computing and Informatics	2	4.1
Math and Computer Science	5	10.2
Math and Statistics	5	10.2
Math	4	8.2
Statistics	5	10.2
Computer Science and EE	3	6.1
Arts and Science	3	6.1
Data Science	8	16.3
Business and Management	3	6.1
Interdisciplinary	1	2.1
Total Programs	49	100

Baccalaureate programs in Data Science compose of most programs within this study with 49. Table 4 demonstrates the distribution of the hosting departments. The standout departments from this category are again Computer Science plus Data Science with 10 and 8, or 20% and 16%, respectively. If

we cluster computer science related departments together, the number of the B.S. Data Science programs becomes 20, which is 40%. We also noticed that there are 8 institutions moving ahead with a standalone unit in Data Science to catch up the trend and demand of the development of the new technology. Without listing all the programs studied in this study due to the length and space, exemplary programs are provided in Table 5 below for a starting point to study specific curricula.

Table 5: Exemplary Ph.D., M.S., and B.S. programs in Data Science

Program	Institution	Website	Category	Type	State
Ph.D.	New York University	https://cds.nyu.edu/academics/	R1	Private	NY
M.S.	Rutgers University	https://www.cs.rutgers.edu/	R1	Public	NJ
B.S.	College of Charleston	http://compsci.cofc.edu/	M1	Public	SC

To provide computer science educators a starting point in studying academic programs in Data Science, both B.S. and M.S. programs listed in Table 5 above are hosted by Department of Computer Science, while the Ph.D. program at NYU is administrated by the Center of Data Science of the University.

Apparently, there is no one-size-fits-all solution to build a Data Science program at an institution. One must consider the following factors during their planning phase: 1) available resources; 2) faculty preparation; 3) enrollment potential. Resources can be many different things, namely funding, administration support, and local employment market. Those factors can play against each other or work cooperatively. The leadership is critical to orchestra the development. Data Science is heavily related to Math, Statistics, as well as Computer Science and Information Technology. It is not common to have faculty members ready under this kind of preparation. Although the first two factors are critical, it is impossible to run a successful program in Data Science without a reasonable enrollment. In other words, no matter how advanced and completed program you might have, it cannot survive without reasonable enrollments. The result of this study reflects the three considerations discussed above as 38% of all programs are offered by the universities and colleges in R1 category since those institutions have the advantages in available resources, faculty preparation, and student enrollment over all other categories of universities and colleges.

5 Conclusions and Future Work

The study of this project tells us that the academic programs in Data Science have grown healthily for the last decade, especially those 13 Ph.D. programs, which provide the foundation for a sustainable future of data science programs. However, it is interesting to noticing while with 115 universities/colleges under the category R1, only 5% of them offer a Ph.D. program in Data Science. If

we consider all doctoral universities and colleges (334), only 3% of them offer a Ph.D. program in Data Science. So, as AI and machine learning begin taking the essential role in data science, more advanced programs, especially at Ph.D. level is definitely needed.

Although knowing what is the current distribution of the programs in Data Science is important, we are planning to further look into each of the programs studied in this work in terms of the curriculum design and implementation in the future so that we can have a complete view in how to build a successful Data Science program.

References

[1] L. Burtch. The burtch works study: Salaries of data scientists, may 2018. `https://www.burtchworks.com/wp-content/uploads/2018/05/Burtch-Works-Study_DS-2018.pdf`.

[2] Chiang R. H. L. Chen, H. and V. C. Storey. Business intelligence and analytics: from big data to big impact. *MIS Quarterly*, 36(4):1165–1188, dec 2012.

[3] R. D. De Veaux and et al. Curriculum guidelines for undergraduate programs in data science. *The Annual Review of Statistics and Its Application*, 4(2):1–16, 2017.

[4] Manieri A. Demchenko, Y. and E. Spekschoor. EDISON data science framework: Part 1. data science competence framework (cf-ds), edison, release 2, v0.8, jul 2017. `http://edison-project.eu/sites/edison-project.eu/files/filefield_paths/edison_cf-ds-release2-v08_0.pdf`.

[5] P. Louridas and C. Ebert. Embedded analytics and statistics for big data. *IEEE Software*, 30(6):33–39, nov 2014.

[6] Chui M. Brown B. Bughin J. Dobbs R. Roxburgh C. Manyika, J. and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity, jun 2011. `https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation`.

[7] V. Mayer-Schönberger and K. Cukier. *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt, Boston, Massachusetts, 2013. ISBN-10: 0544227751, ISBN-13/EAN: 9780544227750.

[8] National Institute of Standards and Technology. *NIST Big Data Interoperability Framework: Volume 1, Definitions, NIST Special Publication 1500-1r1*, sep 2018.

[9] Megan ONeil. As data proliferate, so do data-related graduate programs, feb 2014. `https://www.chronicle.com/article/As-Data-Proliferate-So-Do/144363`.

[10] R. Tang and W. Sae-Lim. Data science programs in u.s. higher education: An exploratory content analysis of program description, curriculum structure, and course focus. *Education for Information*, 32(3):269–290.

An Analysis of the Effect of Stop Words on the Performance of the Matrix Comparator for Entity Resolution*

Awaad Alsarkhi, John R. Talburt
Department of Information Science
University of Arkansas at Little Rock
Little Rock, AR 72204
{aalsarkhi, jrtalburt}@ualr.edu

Abstract

Abstract: This paper investigates the effect of removing stop words on the performance of the matrix comparator for linking unstandardized entity references. Experiments on annotated synthetic customer references produces three outcomes as indicated by the F-measure of the linking results. These outcomes are (1) performances improves using stop words, (2) stop words have no significant effect, and (3) stop words degrade performance. Preliminary results indicate the standard deviation of the token frequencies and the ratio of the highest frequency token to the number of references are predictors of these outcomes. In particular, reference data with high standard deviations and ratios see the greatest improvement (Outcome 1) with the use of stop words.

1 Intoduction

Entity resolution (ER) is the process considered as one of the basic tools to determine whether two references to real-world objects in an information system are referring to the same object, or to different objects[6]. References to the same entity called equivalent references. The goal of ER is to link two

references if and only if the references are equivalent. For this reason, ER is sometime referred to as record linking.

Is measured by the precision, recall, and F-measure of the links it makes between references. The linking precision is the ratio of true positive links (links between equivalent references) to the total number of links. Linking recall is the ratio of the true positive links to the total number of possible true positive links (all equivalent pairs), and the F-measure is the harmonic mean of the precision and recall.

ER processes are based on the Similarity Assumption [7] which states the more similar two references are, the more likely they are equivalent, and similar they are equivalent. Traditionally, measured the values of the same attributes, e.g. First names, street numbers, and dates-of-birth. However, this approach assumes the attribute values in both references have appropriate metadata tags to signify the attributes to which they belong, and that the same metadata tags are used for all references.

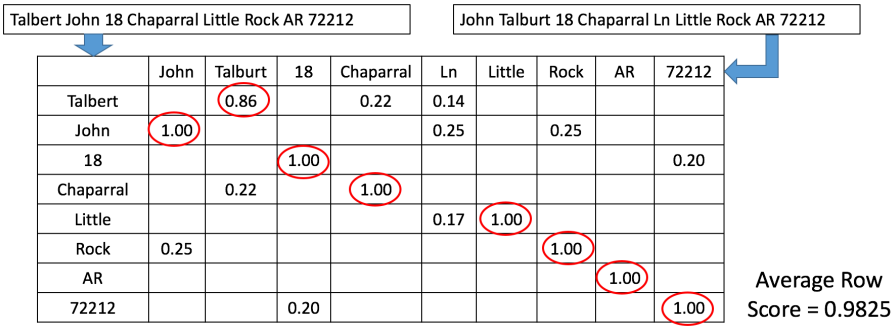
The process to create this type of uniform metadata tagging is called data standardization. Most ER processes rely on having standardized input. However, when there are many disparate sources of data, the standardization process may require a great deal of time and effort to harmonize [3] Even if the sources have already been standardized by the data provider, different sources may have different standardizations. For example, one source may have standardized the references with a separate field (tag) for the street number of a home address whereas another sources standardization leaves the street number as part of a single street address field.

2 Matrix Comparator

One of the approaches to ER developed to avoid the need to standardize references prior to the ER process is the matrix comparator [4]. In this approach, each reference is transformed into a list of tokens, where is each token in a string of characters in the reference separated by a non-word character. In addition, all letters are converted to upper case.

The tokens generated by the two references are compared in a two-dimensional row and column matrix as shown in Figure 1. The tokens from the first string are used to label the rows of the matrix, and the tokens from the second string label the columns of the matrix. The value in each row and column is the normalized Levenshtein Edit Distance (LED) between the row token and column token. For example, in Figure 1 the value 0.75 in the row labeled with token JOHN and column labeled JON is the LED similarity between the strings JOHN and JON. Note in Figure 1 cells with a zero similarity are left blank.

After all of the cells of the matrix have been populated with LED similarity



	John	Talburt	18	Chaparral	Ln	Little	Rock	AR	72212
Talbert		0.86		0.22	0.14				
John	1.00				0.25		0.25		
18			1.00						0.20
Chaparral		0.22		1.00					
Little					0.17	1.00			
Rock	0.25						1.00		
AR								1.00	
72212			0.20						1.00

Figure 1: Example Token Matrix

values, the comparator systematically selects the largest LED values in each row and column in descending order in an iterative process. In the first iteration, the row and column with the largest LED value for the entire matrix is identified and the value is the starting value of a running total. Next, all values in the row and column of the largest value are removed. In the second iteration, the largest value of the remaining values in the matrix is identified. The identified value is added to the running total, and all values in the same row and column are removed. The process continues in subsequent iterations until all of the values have been removed from the matrix.

The number of iterations will be equal to the number of tokens from the string generating the fewest tokens. After the last iteration, the running total is divided by the number of iterations. If the calculated average value is greater than or equal to a threshold value provided by the user, then the comparator returns a true result and links the references, otherwise, it returns a false result and the references are not linked.

There can be in the implementation of the comparator to try to improve its linking performance, such as using a different similarity function other than LED, or even using multiple similarity functions. The variation of interest for this paper is the use of stop words.

3 Stop Words

The use of stop words in the matrix comparator [4] is a simple form of token weighting in which the most frequently occurring words are given a weight of zero, i.e. are excluded from the matrix, and all other tokens are given a weight of one. This is a simplification of the term frequency inverse document

frequency (tf-idf) often applied in techniques for document retrieval [5]. In an ER application, each reference is considered a document. However, the reference are very small documents, and the term frequency value tf is almost always one. Hence, the inverse document frequency is directly related frequency of the token across all documents, in this case all references.

The motivation is that tokens with a very high frequency across all references are less indicative of linking because they are as likely to occur in non-equivalent references as in equivalent references. For example, if a set of references all have addresses in California, then the address token CA will be in every reference and will always contribute to the matrix score. Excluding CA will cause the average similarity score to be based on other tokens presumably more references or non-equivalence.

However, the implication that removing high-frequency tokens will improve the linking performance of the matrix comparator is only a hypothesis. The remainder discusses the results of a perhaps you meant "series of experiments designed to test this hypothesis.

4 Research Design

The experiments are based on of annotated, synthetic customer references generated by different processes. The first corpus was generated by the Synthetic Occupancy Generator (SOG) [8] designed to simulate the movement of consumers from address-to-address and changes of name through marriage over time. It also included gender coding, phone numbers, social security numbers, and dates-of-birth. At the same time, a number of data quality problems were deliberately injected into the SOG data to increase its realism. These included problems such as deleted (missing) values, misspellings, transpositions, truncations, and inconsistent date and telephone formats.

Most importantly for ER research, the SOG corpus includes many redundant (duplicate) references to the same customers in different file formats and with different information and data quality problems. The SOG corpus comprises around 271K records in three different file layouts.

The second corpus also represents customer references, but was generated with an R package called rlErrorGeneratoR [2] also design to produce realistic consumer data with various levels of data quality problems including duplicate records. The R generated corpus comprises about 800K references.

Both corpora also generated a separate annotation file in the form of a crosswalk table listing every generated reference together with identifier. While every reference in each corpus has a different record identifier, different references to the same customer have the same entity identifier in the crosswalk table. The crosswalk table allows an ER metrics program to quickly join ER

linking output to the crosswalk table, count the true positive, false positive, and links, and calculate the precision, recall, and F-measure.

Next, samples of approximately 5,000 references were drawn from each corpus. However, the records were not selected at random. In order to create samples exhibiting a reasonable level of linking, the first step was to append the entity identifier from the crosswalk table to each record in the file. Next, the file was sorted by the entity identifier. After sorting, a segment of 5,000 consecutive records were selected from a random starting point in the sorted file. Because references to the same customer are in adjacent records in the sorted file, this method of stratified sampling guaranteed each sample would contain a significant number of true positive pairs. In all, there were 12 samples drawn from the two annotated corpora.

The next step was to perform a frequency analysis on the tokens in each sample. This was done by using a regular expression to tokenize each reference into substrings delimited by non-word characters W. The collected tokens were then sorted to produce a token frequency table. Finally, the tokens were sorted into descending order by frequency to identify the highest frequency tokens as candidates for stop words. In addition, the frequency distribution of the tokens in each sample of analyzed with three statistics calculated.

1. The average frequency
2. The standard deviation of the frequency
3. The ratio of the highest frequency to the sample size

The final step was to determine the number of stop words producing the best F-measure for each sample. This was done by two trial and error processes. The first process was to select a number of stop words giving the best F-measure results. The starting point was a baseline of no (zero) stop words, then incrementing in steps of 25 stop words.

However, each selection of stop words required a second process for finding the matching threshold that produces the best result for the given set of stop words. For the experiments described here, the ER runs were performed with OYSTER, an open source ER system [9] available on BitBucket [1].

In OYSTER, the matrix comparator is i[1] implemented as a function of the form

MatrixComparator(x.xx, a|b|c|d)

Where x.xx represents the matching threshold given as a number from 0.00 to 1.00, and a|b|c|d represents a list of stop words separated by a pipe (|) delimiter.

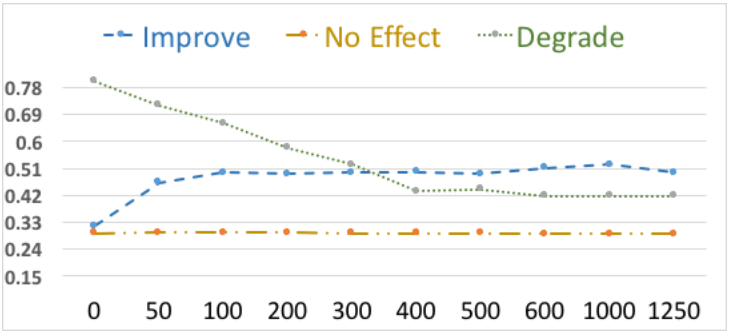
5 Results

Table 1 shows the results obtained from this process for each of the 12 reference samples. The Start F-Measure is the baseline showing the best measure achieved without using stop words. The Best F-measure is the best measure achieved when using stop words. The columns labeled Threshold and Stop Words give the linking threshold and number of stop words giving the best result.

Table 1: Experimental Results from 12 Samples

SampleCase	Start F-measure	Best F-measure	Threshhold	Stop Words	Effect	Freq.Std. Dev	Top Ratio
1	0.318	0.522	0.56	1000	Pos	26.89	0.44
2	0.322	0.518	0.6	200	Pos	25.77	0.39
3	0.293	0.297	0.81	25	None	20.38	0.31
4	0.293	0.294	0.81	25	None	20.05	0.29
5	0.802	0.802	0.67	0	Neg	3.04	0.01
6	0.796	0.796	0.67	0	Neg	3.05	0.01
7	0.872	0.930	0.71	200	Pos	55.47	0.87
8	0.875	0.934	0.69	150	Pos	57.05	0.86
9	0.857	0.922	0.71	150	Pos	55.53	0.87
10	0.851	0.913	0.72	300	Pos	54.9	0.87
11	0.869	0.912	0.74	400	Pos	58.27	0.85
12	0.9	0.931	0.79	100	Pos	61.31	0.82

In these cases, increasing the number of stop words seems to have no effect, either positive of negative. These three results are illustrated in Figure 2. This graph shows the F-measures obtained for varying numbers of stop words for three different samples.



The graph plot labeled Improve shows the results for Sample 1, the No Effect plot shows the results for Sample 3, and Degrade plots the results for Sample 5.

From these preliminary results, it appears these results are correlated with the distribution of the token frequencies. Samples in which the token fre-

quencies where widely dispersed from very large to very small were related to positive outcomes.

Two measures of the frequency dispersion have been added to Table 1 to illustrate this observation. The column labeled Freq. Std. Dev shows the standard deviation of the token frequencies. The column labeled Top Ratio is the ratio of the highest frequency token to the number of references in the sample.

6 CONCLUSION AND FUTURE WORK

These experiments show that when comparing non-standardized references with the matrix comparator, the addition of stop words does not always improve linking results, and in some cases, may actually degrade performance. Based on the samples used for these experiments it appears the use of stop words will only be effective when there is a wide dispersal of token frequencies, i.e. when the frequency distribution has a large standard deviation.

The results presented here using samples of 5,000 references, no significant improvement from the use of stop words occurred unless the standard deviation was 25 or greater. In addition, the best improvements were obtained with the highest frequency token occurred in 40% or more of the references.

Future work is need to confirm these results with other sources of references data. Also further statistical analysis can perhaps quantify the relationship. A regression analysis may allow a user to predict not only whether stop words are justified, but perhaps the number of stop words and the threshold value that will yield the best performance.

References

[1] Oyster Open Source Project. `https://bitbucket.org/oysterer/oyster/`.

[2] rlErrorGeneratoR. `https://github.com/ilangurudev/rlErrorGeneratoR`.

[3] A. Jurek-Loughrey and P. Deepak. *Semi-supervised and unsupervised approaches to record pairs classification in multi-source data linkage.* Springer, 2018.

[4] X. Li, J. R. Talburt, and T. Li. Scoring matrix for unstandardized data in entity resolution. In *Proceeding of the Computer Science and Computer Intelligence Conference*, Las Vega, NV, 2018.

[5] G. Salton and C. Budckley. Term-weighting approaches in automatic text retreival. *Information Processing & Management*, 24(5):513–523, 1998.

[6] J. Talburt. *Entity Resolution and Information Quality.* Morgan Kaufamann, 2011.

[7] J. R. Talburt and Y. Zhou. *Entity Information Life Cycle for Big Data*. Morgan Kaufamann, 2015.

[8] J. R. Talburt, Y. Zhou, and Y. Shivaiah. Sog: A synthetic occupany generator to support entity resolution instruction and research. In *Proceedings of the International Conference on Information Quality*, Cambridge, MA, 2009.

[9] Y Zhou and JR Talburt. OYSTER: An open source entity resolution system supporting identity information management. In *ID360-The Global Forum on Identity, Austin*, volume 90, 2012.

Developing a Guided Peer-Assisted Learning Community for CS Students*

Yi Liu[1], Gita PhelpsA[1], Fengxia Yan[2]
[1]Department of IS and CS
Georgia College and State University
Milledgeville, GA, 31061
{yi.liu,gita.phelps}@gcsu.edu
[2]Morehouse School of Medicineă
20 Westview Dr SW
fyan@msm.edu

Abstract

This paper describes an effective learning community run by CS students, and guided by CS faculty. The community offers a positive learning environment outside classrooms, providing timely supportive services to students who have different learning styles. It provides three types of services: online teaching videos made by freshmen and sophomores, onsite tutoring meetings provided by juniors and seniors, and supplemental instruction led by sophomores and juniors. The goal is to improve the academic performance of the students in the CS courses, and therefore, hopefully increasing the retention rate of CS majors. The data collected during the past six years show that the average percentage of the DFW rate in the first programming course is reduced after the community was assembled. At the same time, the community had positive effects on retaining CS students.

Introduction

The retention rate of CS majors is usually low. More than half of the students that initially choose to major inăCS either exit the field or drop out

of college [2]. Students who leave the major usually have low performance in their CS courses; therefore, efforts to improve the academic performance of the students in CS courses are important to retain CS students [12]. Many pedagogical techniques have been developed for use in classrooms, such as flipping the classroom, group work, and peer teaching. Additional methods found to be beneficial in CS courses include interactive teaching [3], subtitle labeling [7], and visualization [7]. The leading reasons mentioned by Guzdial [3] that make CS courses difficult to teach include instructors who underestimate the complexity of coding problem assigned, and students with diverse educational and social backgrounds who respond to different teaching styles and methods. Guzdial states that even a small change to a programming problem could alter the complexity of a problem. But the instructors, as experts, aren't good at predicting how difficult the modified problem would be. The students may need different remedial help if they fall behind. Moreover, although the students effort beyond the classroom is critical to the success of learning, few papers have systematically addressed how to work with the students after the class.

This paper describes a peer-assisted learning community run by CS undergraduate students, and guided by the CS faculty, focusing on providing different services to assist the learning outside classrooms. The community is comprised of CS students of all levels – from freshmen to seniors. Research has shown that students feel more comfortable and open when interacting with a peer [4]. The peer-assisted community can provide a comfortable environment to support the learning with low cost, which is particularly suitable for small liberal art universities. Three types of services provided are online teaching videos, on-site tutoring meetings, and supplemental instruction. Freshmen and sophomores create online teaching videos when they are enrolled in lower-level courses. Each video provides remedial support for a confusing and challenging concept in the first two CS courses. CS juniors and seniors provide on-site tutoring meetings. The meetings are usually available Monday through Thursday for many CS courses. Supplemental instruction (SI) is led by CS sophomores and juniors during the first and the second programming courses.

By providing three different services, the community helps students to individualize their learning in a way that fits their personal learning styles and paces. A student can reinforce a difficult topic by watching videos online, attending a one-on-one tutoring meeting, or participating in a one-on-many SI session when needed. The community also creates great bonding opportunities for CS students. The students offering the services usually are our top students and are role models for the other students. They show positive learning attitudes, provide support during potentially frustrating situations, and offer constructive learning experiences. This positive bonding may increase the

retention of CS students.

The data we collected during the past six years include grade distributions from the first programming course, the retention rates of CS majors, and the student evaluation surveys. The results show that the average percentage of DFW rates dropped in the first programming course; the 1-year and 2-year retention rates increased for CS majors; and the majority of the students agree or strongly agree that the community is a valuable asset, and would recommend it to other students. The 3-year and 4-year retention rates are not reported in this paper since the upper-level CS courses have been constantly modified to meet the requirements of ABET. ă

This paper describes the structure of the learning community, our experience assembling the community, and our analysis of ăthe data collected over the past six years. The last section shows the conclusions based on the grade distributions, the retention rates, and the surveys given at the end of each semester.

The Structure of the Guided Peer-Assisted Learning Community

Online Peer-Teaching Videos

With emerging technologies, the fast speed of the Internet, and the low costs associated with the video production, using videos to teach programming is increasingly prevalent on the Internet and in classrooms. Many students search for examples and solutions to assignments; however, obtaining high-quality videos to explain the entry-level programming concepts or questions that only typical freshmen have is difficult and time-consuming when the videos are everywhere. Furthermore, most of the online videos usually provide the solutions using different methods not discussed in class, so students are still not able to understand the topics discussed in the class. The videos found on the Internet are usually too long to view when a student wants a specific feedback to a single fundamental concept or a coding question. ă We decided to build our own peer-teaching video channel [1]. All videos are made by our students for their peers. Each video focuses on a challenging concept for many of our students. Meanwhile, the students who create the videos reinforce and extend their own learning, and improve their communication skills and self-esteem [4]. The other students watching the videos are not only benefited by receiving more time on the difficult topics to gain understanding of the concepts, but also feel motivated, knowing that the videos were created by peers at the same institution who have taken the course. The videos are very short, lasting about 5-10 minutes. They are posted on a YouTube channel created by the depart-

ment and now have more than 79,000 views. The top two popular ones have more than 10,000 views as heretofore [11].

On-Site Guided Peer-Tutoring

The community provides on-site peer-tutoring meetings after the class, guided by the CS faculty. Some research papers show that the tutoring works best when students of different ability levels work together [8, 10]. The guided peer-tutoring program can not only improve the academic performance of both tutors and tutees, but also build supportive relationships among peers [4, 6, 9]. In order to deliver an effective tutoring program, CS faculty screen the tutors based on their overall knowledge of CS, oral communication abilities, potential teaching skills, and personalities. Tutors are role models who are willing to share their positive learning experiences and passion toward CS to the tutees.

The guided peer-tutoring meetings are tailored to attract the students by scheduling tutoring hours according to the students needs. The goal is to provide timely one-on-one help to individualize the learning of tutees. In general, the tutoring meetings are offered in the late afternoons and early nights when course instructors are not available.

Supplemental Instruction for Lower-Level CS Courses

The community provides Supplemental Instruction (SI) to support the difficult lower-level CS courses by offering peer-assisted study sessions regularly after the class. Students who have successfully completed a course can be hired as a Supplemental Instruction leader. ăThe sophomores and juniors who deliver the SI service are paired with a professor and attend the class with the students. They host group study sessions twice a week, providing informal reviews on the course assignments and exams. In general, each lower-level programming class has one SI leader, working 8 hours each week. ăThey are required to attend every class and assist CS faculty by helping to answer questions during in-class activities. They host one-on-many review sessions related to the course. The sessions are offered in a comfortable and open place so the students can directly interact with the peers, and take the control of their own studying.

DATA COLLECTION AND ANALYSIS

The guided peer-tutoring service began the fall of 2010, and is reported two years later [5]; while the other two started during the years of 2011 and 2012. The strategy developed to build the effective guided peer-tutoring helps us to design the community. For example, the number of students who used the

peer-tutoring service were much lower than what we expected during the first year. In order to fully utilize the tutoring service, a strategy was developed to promote it. The strategy includes announcing it at the very early of each semester, posting flyers in the hallway and on the door of each instructors office, publishing tutoring schedules on course websites, frequently making teacher referrals, inviting tutors to visit the classroom, and assigning extra challenging assignments to encourage the good students to visit the tutors. The strategy is then applied to the other services of the community to boost the visits. At the same time, different surveys were developed and sent to the students to help us understand and maximize the benefits offered by the community.

Data Collection

The community was assembled in 2011. In order to understand how the community assists the learning, the data collected included the grade distributions from our first programming course, the retention rates of our CS program, and the results from surveys at the end of each semester. The grade distributions and the retention rates from 2005 to 2017 were categorized into two groups; one included the dataăbefore the community was created from 2005 to 2010; the other contained the data during which the community was implemented from 2011 to 2017. The results from both groups are presented in this paper.

Grade Distributions from the First Programming Course

The first programming course is one of the classes in the core curriculum of the university. It usually has a mixture of CS major and non-major students with different learning styles, and social and academic backgrounds. This course is listed as one of the courses with the highest DFW rates on the campus. Three instructors are usually allocated to teach the first programming course each year. However, this paper only reports the grade distributions from one instructor who has taught the course for more than 20 years, and has constantly advocated the peer-teaching videos. The second instructor joined the department in 2014, and sometimes applied neither SI norăthe peer-teaching videos; while the third one who was hired at the year of 2003 frequently altered her teaching styles and exercises for several years to work with the students in the liberal art college. She did not fully promote the peer-teaching videos until 2015.

Table 1: Grade Distributions

Years	A	B	C	DFW	Total
2005-2010	31.0%	15.7%	10.5%	42.9%	420
2011-2017	31.6%	20.9%	14.1%	33.4%	383

Table 1 shows the grade distributions of the first programming course between two groups, the group without the community from 2005 to 2010, and the other one using the community from 2011 to 2017. The first, second, third and fourth columns show the percentages of A, B, C, and DFW in each group respectively. The average percentage of students who earned DFW is dropped from 42.9% to 33.4% ăin the fifth column. The chi-square statistic test is then applied to the results. The chi-square test returns 7.5449 for the DFW rate with the p-value of .006018, indicating that the drop is significant at $p < .01$.

Retention Rates from 2005 to 2016

The 1-year retention rate for the year of 2017, and the 2-year retention rate for 2016 have not been released when we began to write the paper at the beginning of the fall of 2018. ăSo only the retention rates from 2005 to 2016 are used in the paper.

Table 2 lists the 1-year and 2-year retention rates for our CS program. The data are also categorized based on gender, male and female. The 1-year and the 2-year retention rates for male students gain about 5.8%, and 8.5%, respectively. ăThe 1-year and the 2-year retention rates for females increase about 33.4% and 31.3% separately. However, the Fisher exact test statistic value for females is 0.1279, indicating that the difference is not significant statistically. The exact power for Fisher's Exact Test returns 30%, implying that the sample size is not big enough to claim the difference statistically. However, the results are very encouraging, showing that 15 out of 27 females stayed in the program after the first year. It is worth for further considerations when more females enter the program. The overall 1-year and the 2-year retention rates for all students rise about 9.2%, and 11.4%, respectively. Table 3 then shows the retention rates of the university. The 1-year and the 2-year retention rates of the university for all students increase from 83.8% to 85.2%, and from 69.6% to 69.9% respectively. ăThe comparison of the retention rates between CS program and the university further confirms that the community is worth investing.

Student Evaluation Surveys

Tables 4-6 briefly list some samples of student evaluation surveys for the community. Most of the students who use the community services agree or strongly agree that videos, tutoring meetings, or SI sessions are helpful to understand the course material. ăThey will also recommend the services to the other students. Although the percentages of students who agree or strongly agree that the services help to improve their grades in the course are usually lower than

Table 2: Retention Rates of CS Majors

Years	Total CS Students	1-year-retention count	2-year-retention count	1-year-retention rate	2-year-retention rate
total males 2005- 2010	61	33	23	54.1%	37.7%
total males 2011-2016	142	85	55	59.9%	46.2%
total females 2005- 2010	9	2	1	22.2%	11.1%
total females 2011-2016	27	15	9	55.6%	42.9%
total 2005-2010	70	35	24	50.0%	34.3%
total 2011-2016	169	100	64	59.2%	45.7%

Table 3: Retention Rates of the University

Term-Desc	Cohort_Count	1-year-retention-rate	2-year-retention-rate
total males 2005- 2010	2574	82.8%	69.2%
total males 2011-2016	3125	82.5%	67.2%
total females 2005- 2010	4378	84.4%	69.8%
total females 2011-2016	5093	86.9%	71.6%
total 2005-2010	6952	83.8%	69.6%
total 2011-2016	8218	85.2%	69.9%

the other survey questions, the comments obtained from the students are very positive, indicating that the community is valuable to most of the students.

Table 4: SI Evaluation Survey

	Question	Average(1-5)
1	SI is well-prepared and capable	4.72
2	SI treated the others and me with respect	4.83
3	SI made himself available to students	4.94
4	My grade improved because of SI	3.89
5	SI sessions were helpful for me	4.28
6	Due to the skills I have gained from SI sessions I am now more confident about doing well in the university than I was at the beginning of the semester.	3.94
	18 out of 25 respondents attended SI sessions	72.0%

CONCLUSIONS

This paper presents a guided peer-assisted learning community conducted by CS undergraduate students from freshmen to seniors. ăThe community guided

Table 5: Tutor Evaluation Survey

	Question	Percentage of students who agree or strongly agree
1	My tutor(s) was(were) helpful.	88.5%
2	My tutor(s) knew the material well.	84.60%
3	My tutor(s) helped me to understand the course material.	80.80%
4	The CSCI tutor(s) helped me to improve my grade in the course.	76.9%
5	I would recommend using the CSCI tutoring to other students	88.0%

Table 6: Video Evaluation Survey

	Question	Agree or strongly agree	Somewhat agree	Disagree or strongly disagree
1	Overall, the videos were helpful.	90%	10%	0%
2	Overall, the videos helped me improve my grade in the course	57%	38%	5%
3	I would recommend using the CSCI videos to other students	90%	10%	0%

by the CS faculty is highly cost-effective, making it very suitable for small liberal art universities. The data collected from the past six years show that the community successfully reduced the DFW rate in the first programming course significantly. The 1-year and 2-year retention rates increased after the community was assembled. The results from the surveys show that the majority of the students agree or strongly agree that the community is beneficial and helpful to them. Therefore, we will continue to build the community to help the students to achieve higher academic performance by providing various services to assist the learning outside classrooms. Our further research will continue to investigate the effectiveness of the community on retaining students, especially females since the result is very encouraging, showing that 15 out of 27 females stayed in the program after the first year. The research will further explore ăhow to make the community more valuable to the other programming courses.

References

[1] CS YouTube Channel. `http://www.youtube.com/user/cspeerteacher`.

[2] Xianglei Chen and Matthew Soldner. STEM attrition: College students' paths into and out of STEM fields. `https://nces.ed.gov/pubs2014/2014001rev.pdf`.

[3] Mark Guzdial. Learning computer science is different than learning other STEM disciplines. *Communications of ACM*, 1 2018.

[4] Russ Hodges and William G. White. Encouraging high-risk student participation in tutoring and supplemental instruction. *Journal of Developmental Education*, 01 2001.

[5] Yi Liu, Gita Phelps, and J. F. Yao. Design and benefits of an on-site tutoring program for the first programming class. *Journal of Computing Sciences in Colleges*, 29(5):42–49, May 2014.

[6] Martha Maxwell. Does tutoring help? a look at the literature. *Review of Research in Developmental Education*, 7(4), 1990.

[7] Thomas L Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. Exploring the role of visualization and engagement in computer science education. In *ACM Sigcse Bulletin*, volume 35, pages 131–152. ACM, 2002.

[8] David C Rheinheimer, Beverlyn Grace-Odeleye, Germain E Francois, and Cynthia Kusorgbor. Tutoring: A support strategy for at-risk students. *Learning Assistance Review*, 15(1):23–34, 2010.

[9] Nathan Rountree, Janet Rountree, and Anthony Robins. Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin*, 34(4):121–124, 2002.

[10] Keith J Topping. The effectiveness of peer tutoring in further and higher education: A typology and review of the literature. *Higher education*, 32(3):321–345, 1996.

[11] Susan Wiedenbeck, Deborah Labelle, and Vennila N R Kain. Factors affecting course outcomes in introductory programming.

[12] Brenda Cantwell Wilson and Sharon Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. In *ACM SIGCSE Bulletin*, volume 33, pages 184–188. ACM, 2001.

Google Analytics*

Conference Tutorial

Daniel Brandon
MIS, School of Business
Christian Brothers University
Memphis, TN 38104
dbrandon@cbu.edu

Google Analytics (GA) and such systems are essential tools in todays complex corporate marketing environment. Businesses now gather vast amounts of information today not only via their transaction processing systems (TPS), but also thru a myriad of social media platforms such as Facebook and Twitter. Thus the term big data. TPS data is now generated not only from store point of sale systems but also from online web purchases.

Businesses need to digest and carefully analyze this vast amount of data to determine who their customers are, how to best serve and retain these customers, how to sell more product to these customers thru up selling and cross selling, who else could be their customers, and how best to convert these potential customers cost effectively thru targeted advertising.

Google Analytics allows organizations to generate, analyze, and visualize information about their website and mobile app usage by customers and other visitors. GA also helps organizations to better focus available resources to generate and retain more customers and enables organizations to work smarter to generate more success from their digital property. Specifically GA tracks the origin of website visitors and relates these origins to page interactions, conversions, purchases, downloads, and other online transactions.

This tutorial will discuss the following topics:

- What is GA
- Why is it vital to business today
- How to incorporate GA into a web site
- How to use GA data analytics tools
- How to use R to clean, process, and visualize GA data

Presenter Background

Dr. Brandon is a Professor of Management Information Systems at Christian Brothers University in Memphis TN where he teaches courses in MIS, programming, database, data analytics, statistics, and project management. He has authored two books and numerous journal articles and conference proceedings including some for CCSC. He has designed and developed modern web based information systems for a number of organizations and was formerly the Director of Information Systems at the NASA Stennis Space Center.

Software Design Patterns Applied To Building An Interpreter*

Conference Tutorial

Larry Morell
Computer and Information Science
Arkansas Tech University
Russellville, AR 72801
lmorell@atu.edu

While it is possible to teach software design patterns via a series of disconnected examples, it is useful for students to see how several design patterns can be applied in a single development effort. By doing so students see better how patterns interrelate. Furthermore, as they apply additional patterns to later stages of the development, their grasp of previously applied patterns is reinforced.

This tutorial demonstrates how to teach a variety of software design patterns in the context of building an interpreter for a simple programming language called ELSE (Experimental Language for Software Education). This approach has been taken and refined over the past five years in an undergraduate course in Software Engineering and in a graduate course in Software Design. The focus of the courses has been on teaching patterns, but students also learned a substantial part of how to build an interpreters for simple languages. The final project in the course required students to investigate independently one or more software patterns and apply what they have learned to the interpreter they had already built.

This tutorial describes the variety of design patterns that have been taught in the courses and how they are applied to aspects of the interpreter. Each attendee will receive a copy of the assignments given along with their solutions in Java.

Teaching Object-Oriented Programming with Geometry*

Conference Tutorial

Serge Salan
Department of Mathematics and Computer Science
Christian Brothers University
Memphis, TN 38104
ssalan@cbu.edu

This tutorial provides a method to teach object-oriented programming (OOP) to new programmers. The method relies on using a running example to explain basic to advanced concepts in OOP. The example is based on geometry objects, e.g., a point, a triangle, a rectangle, as well as known concepts such as calculating a distance. Geometry is an area of mathematics that is accessible to students, and the teacher does not usually need to provide background knowledge before starting the lecture. Furthermore, it gives a natural way to illustrate the topic with figures, which allows the students to rely on visualization to better understand the material.

The tutorial covers the main topics of OOP: abstraction, encapsulation, inheritance and polymorphism, and applies concepts from geometry in each topic. The language used is Java. We will show working Java code to accompany every example. Other important Java features are discussed, namely access control, static methods, array of objects, and multiple constructors in a class. Finally, this tutorial will suggest programming assignments that are based on the code learned during the lecture.

Speaker Background

Serge Salan is an assistant professor of computer science completing his fourth year at Christian Brothers University. His Ph.D. in computer science is from the University of Memphis. Courses that he regularly teaches include the freshmen level programming sequence, data structures, and algorithms. He is knowledgeable about Java, Python, and C++, and taught OOP in these languages.

Scalable Processing of Massive Text Data Stores for NLP*

Conference Tutorial

Brittany Bright, Cesar Cuevas, Israel Cuevas, Andrew Mackey
University of Arkanas at Fort Smith
Fort Smith, AR 72913
{brittany.bright, israel.cuevas, andrew.mackey}@uafs.edu
ccueva00@g.uafs.edu

The storage and processing of text are integral components of machine learning and natural language processing algorithms. As text-generating sources continue to enlarge the size of natural language data stores, the ability for individual systems to process overwhelming volumes of data becomes challenging. The emergence of systems capable of parallelizing text processing has enabled researchers to rapidly build, train, and deploy intricate NLP and machine learning models. In this tutorial, methods of parallel processing of massive text data stores for NLP and machine learning algorithms will be introduced. Tools for constructing models using a variety of approaches, ranging from typical frequency implementations to graph representations of text, will be reviewed. In addition, challenges for both industry and academia will be discussed.

Longest Pattern Lock*

Nifty Assignment

Jingsai Liang
Computer Science Department
Westminster College, Salt Lake City, UT 84105
JLiang@westminstercollege.edu

Pattern lock with nine vertices is a very common way to secure the smartphone. In this assignment, students are expected to find and draw the longest pattern lock in term of the length of the password under two different rules which decide if a path is a feasible password. This assignment is a great chance for students to apply permutation and brute force they learned in class to solve this real-world problem, which leads them to realize the data structure and algorithm are really helpful and powerful in the real world. Finally, students can easily visualize the pattern lock by using python package matplotlib, which code is provided in the starting code.

In order to calculate the length of a password, we convert nine vertices 1 to 9 of the pattern lock to nine points on an xy plane from (0,0) to (2,2). For example, the coordinate of 1 at the bottom left is (0,0) and the coordinate of point 6 is (2,1). A password on the pattern lock can be defined as a set of continuous line segments. A password is thus also a path. We will use password and path interchangeably in this assignment. The length of a password is the sum of distances between every two adjacent points on the path. Not all permutations of "123456789" are paths. We should follow one of the following two rules when designing a password. Students are expected to find all paths having the longest length and draw one such path under each rule.

Rule 1 of Path: You CANNOT directly draw a line segment from a to b without passing the middle point c. For example, "132456987" is not a feasible path since you cannot draw a line segment from 1 to 3 without passing 2. One feasible longest path under this rule is "276183495". Rule 2 of Path: You CAN directly draw a line segment from a to b without passing the middle point c if point c has already been used before a and b in the path. For example, "213456987" is not a feasible path under rule 1, but is a feasible path under rule 2 since 2 has been used before 1 and 3 in this path. One feasible longest path under this rule is "519283764".

Understanding the Identity Function in SML by Theory*

Nifty Assignment

Cong-Cong Xing[1], *Jun Huang*[2]
[1]*Dept. of Mathematics and Computer Science*
Nicholls State University
Thibodaux, LA 70310
cong-cong.xing@nicholls.edu
[2]*School of Computer Science*
Chongqing Univ. of Posts and Telecommunications
Chongqing, China 400065
xiaoniuadmin@gmail.com

Motivation Being one of the representative languages in the paradigm of functional programming, Standard ML (SML) offers many advanced features that are rarely seen in imperative programming. One such feature is the type/typing in SML, which can be exemplified by the identity function (and programs related to it). While SML provides some explanations to the typing issues of the identity function, these explanations may seem to be ad-hoc, obscure, and hard to grasp by students. We believe that a clearer approach to understanding these typing issues would be by using the theory of the 2nd order typed λ-calculus. Meanwhile, by doing so, students can be given a chance to see an application of the theory and to appreciate the importance of theory, as well.

Overview Identity function, which takes an argument and returns the same argument as the result, is an interesting instance in SML. It can be simply coded as `fun id x = x` in SML, and the (typing) response of SML to this definition is `id = fn: 'a -> 'a`, meaning that the identifier id is bound to a function that takes an input of *any type* and returns a result of the same type as that of the input. Here, `'a` is the notation of type variable in SML, signifying that the value of `'a` can be any type (e.g., `int, bool, int->int`, etc.). While the definition of the identity function `id` and its (deduced) type

are not that hard to understand for students, it is the following programs that confuse them tremendously. Given `id` as defined, on one hand, the code `id(id)` generates an error in SML. In other words, the application of `id` to itself does *not* give the identity function itself as expected by many students. On the other hand, the code `id(id:int->int)`, that is, the same application but with the type of the argument id being particularized, does produce the identity function (of type `int->int`) as expected. So, the question is how to understand this factual typing in SML. The standard textbook (e.g., Elements of ML Programming, by Jeffrey Ullman, Prentice Hall, 1998) interpretation for this uses notions of generalizable and non-generalizable type variables, and expansive and non-expansive expressions – terms that are coined specially to cope with this kind of typing instances, and have no meaning otherwise. As a result, these interpretations seem to be ad hoc, not convincing, vague, and indifferent to most students. Therefore, a clearer, more accurate, and more convincing explanation may be needed, and this is exactly what the 2nd order typed λ-calculus can provide. As such, an assignment requiring the explanation of the typing of following SML programs (1) `fun id x = x`, (2) `id(id)`, and (3) `id(id:int->int)` using the framework of the 2nd order typed λ-calculus can be formed.

Classroom Observations This assignment was given as homework for a senior-level Functional Programming course in computer science. Most students were able to more or less figure out the typing of the identity function, but had trouble dealing with the rest. After being explained, students realized that they actually did not truly understand the meanings involved in the typing rules, and agreed that the theory does give a formal and precise foundation for them to understand the typing behaviors of SML.

What can be gained Upon completion of this assignment, students are expected to

- Master the typing rules of the 2nd order typed λ-calculus.
- Have a clearer understanding of the typing issues associated with the identity function in SML.
- See the connection between theory and practice in programming languages. Realize and appreciate the importance of theoretical studies.

We hope that this assignment (solution can be obtained by contacting the authors) will be useful to colleagues.