# Software Engineering to Develop Online Phonetics Educational Training: Interdisciplinary Research with Communications Sciences and Disorders

**Sicheng Li, Marisha Speights Atkins, Dallin Bailey, Jueting Liu, Yang Cao, Robertson Bassy, and Cheryl D. Seals**
*Auburn University Auburn, AL USA*

## Abstract

In this paper, we discuss an interesting technique to improve student intrinsic motivation through service-based learning projects. The User Interface Design course focuses on the theory and practice of designing and developing interactive systems and reinforcing Computer Science student software engineering skills. We enlist community partners with interesting visions for interactive systems. In this work, we will focus on the problem of improving user experience in phonetic transcription. The case discussed is a complex multivariate problem in the field of Communications Disorders of using the International Phonetic Alphabet (IPA) as a transcription language to support students learning the IPA language. The common practice is for students to transcribe in hand-written form followed by manual evaluation of this work by instructors as a tedious phoneme-by-phoneme process, which may take several weeks to provide detailed feedback for a large number of words. The research question was, "How do we provide more timely and consistent feedback on their transcription accuracy?". Our focus was to develop an engaging, easy to use platform that can support interactive online learning and reduce complexity of evaluation and feedback for transcript tasks.

## Keywords

E-learning, Edit Distance, Web Development, Automated Grading, Tools for Linguistic Education.

## Introduction

The User Interface Design (UID) course focuses on the theory and practice of designing and developing interactive systems and reinforcing student's software engineering skills through service-learning. This course is taken as an upper-level undergraduate course or graduate course. We begin the course with the theory of UID and after reviewing promising practices and techniques to improve user interface design, the students are presented with case studies as design and development challenges. These problems provide a rich learning experience that is applied, practical and engaging. The case that we will focus on for this paper is creating an online phonetics educational training tool, the Automated Phonetic Transcription grading tool (APTgt). A key feature of the application is a modified Edit Distance algorithm implemented to calculate the difference between the phonetic transcriptions of the student's answer and the professor's answer. These phonemic strings are recorded after the student listens to an online audio recording provided through our web application and transcribes the speech using a dedicated phonetic keyboard developed for this application (i.e., a variant of the IPA keyboard).

The proposed algorithms are the core of the APTgt platform, which is an online learning application created as an interactive and engaging examination tool for students in the Department of Communication Disorder at Auburn. The instructors generate online exams for students, which mainly consist of recorded words that the students transcribe which are subsequently scored using the algorithms to provided exam results immediately for the students. The goal is to improve learning outcomes in phonetic transcription using automated software tools to save man-power and reduce errors. The core of this platform is the scoring mechanism: the modified edit distance algorithm. Algorithms are designed to calculate the edit distance of two phonetic representations of a phonetic word (standard key provided by the instructor and the students' answer) and demonstrate the transformation path. The platform provides a total accuracy of approximately above 90% and also speeds up the grading process several orders of magnitude by reducing response time from several weeks to immediate response (i.e., compared to manual traditional manual grading).

This advanced technology application supports student learning through reinforcement and educator's creativity through this time-saving application. The trending usage of the Internet drives a shift in the way people learn and will have a profound impact on the evolution of computer usage. This in return gives birth to the need for new learning technologies [1-3]. Students are excited about new technologies, especially the Internet, and most of our student population are digital natives [3-6] who are accustomed to powerful technology right in the palm of their hands. It is no longer admissible to train students with only traditional learning approaches. Over the past few years, there has been a growing emphasis on the course content, but not much focus has been placed on this specific task-oriented system. Our focus was to develop an engaging, easy to use platform that reduces effort for instructors and supports interactive learning for linguistics education. The APTgt webservice platform with the edit distance component diagram is seen in Figure 1.

## Background

E-Learning [7, 8] is one of the most significant new instructional approaches available; it drives progress in both the teaching and learning process in a wide range of schools around the world today. E-Learning is delivered and supported using a variety of electronic media. These technologies allow us to deliver individualized and comprehensive learning content that facilitates learning, anytime, and anywhere. It does not replace the traditional classroom approach but creates an augmented learning environment. This approach rather promotes combined usage of teaching techniques to maximize the student's participation in the learning process. Accessing the learning materials allows students to go beyond the limits on time and place imposed by the traditional classroom.

*Case-based web learning versus face-to-face learning: a mixed-method study on university nursing students*

Case-based learning (CBL) is an effective educational method for enhancing students' learning and reasoning skills and is an alternative method to the traditional classroom approach. Sek-Ying Chair [9] and coauthors utilized a sequential mixed-methods study using both quantitative and qualitative research to conduct an exploration of CBL experience for nursing students in Hong Kong. The researchers compared traditional and Web-based approaches and concluded that

although their qualitative data support that CBL enhanced self-learning ability and critical thinking skills of participants, they found no difference between two methods in self-learning ability and clinical reasoning ability. The participants reported that CBL helped them apply learned knowledge to clinical situations and they appreciated the flexibility of a web-based approach. The authors concluded that a rigorous structural design, real-time synchronized e-discussions, and cultural sensitivity to students' learning behavior are all crucial components of a successful Web-based CBL approach in the context of their nursing education.

The development of The Automated Phonetic Transcription Grading Tool (APTgt), served as a mechanism for providing case-based learning in the communication disorders courses. Theoretical instruction is applied through case-studies in the advanced speech disorders courses. The case studies serve as a means of cultivating the ability for students to think critically to formulate clinical diagnostic decisions. One skill students in these courses learn is the ability to code spoken words (phonetic transcription) using the International Phonetic Alphabet (IPA) system. Phonetic transcription involves capturing the sounds of speech in written form to create a transcript that represents how words were produced by an individual speaker [15]. This written phonetic transcript is important for continued assessment and clinical diagnostics. Mastery of this skill for students requires regular practice and performance feedback. One factor that impedes the provision of applied practice opportunities is the widely agreed upon problem of grading phonetic transcription assignments by hand [13]. Addressing this problem presented a unique pedagogical opportunity for enhancing student learning with the use of an online learning platform that could automate the grading process in order to provide students with timely feedback that supported deep learning of applied clinical concepts [11]. The design of such an application created an applied learning opportunity for students in a user interface design course by engaging students in a case-based learning approach to solve a real client presented need that allowed for instruction in the iterative process of web application design.

Operationalizing and automating the phonetic transcription grading process in the APTgt learning system was implemented through the use of edit distance calculation by phonetic rules and word-length normalization. Edit distance is an accepted method for string-to-string comparisons when comparing differences between characters in words. The work of Bae et al. [10] proposed the well-known algorithm for string-to-string comparison in the context of the Korean language. This Edit Distance method applied a consonant normalization factor for syllable-structured word similarity. Their method was designed to improve the performance of the syllable-based and letter-based metrics for word similarity. They concluded that the performance of edit distance was improved with their strategy via the phonetic pronunciation rules and word-length normalization. They reported that the phoneme-based metric provided a better result compared to the approaches of letter-based, syllable-based and hybrid distance methods. Modifications to the traditional [12] edit distance calculation were applied to the APTgt algorithm to account for the complexities of spoken language and the phonetic representation.

## Methods

### UID Course Learning Model

The User Interface Design course, taken by software engineering upper-level undergraduate and graduate course, incorporates a component of the class that supports service learning. Addressing this problem within this course introduced the opportunity to provide a rich learning experience for the User Interface Design student that was practical and engaging. The course teams began this effort by gathering requirements from the subject matter experts in the field of communication disorders. Then based on these requirements, user scenarios were crafted for the Student User, Teacher User and Admin User of the system. The scenarios were captured utilizing UML (Unified Modeling Language) that is utilized to capture a pictorial view of the system and cataloging roles, actors, actions and classes within a system. Once the system scenario is captured, software requirements created, software language identified and environment identified the software development team will begin iteratively developing software to instantiate this software system. We also will need to pilot test the software, and at the end of the first cycle of development, the team will need to test the software and have users to validate that the system works as anticipated. At the end of the first cycle of development the development team, design team, and the content experts reach an agreement that the planned scenario meets the specified requirements for the user [11].

### Learning Outcomes for Engineering Course

Learning Outcomes for the Computer Science and Software engineering students. We deliver instruction for our students to understand the theory of user interface design. We have students engage in software development projects to have a practical exercise and fully elaborated case study. This learning episode begins with requirements, design, development, testing, and a project presentation of findings from preliminary user evaluations pertaining to the analysis of user satisfaction and system effectiveness. We have found that this gives students a great understanding of the user interface design process. With respect to teamwork, it also gives them a great experience in teamwork as they have to collaborate with a team of 4-8 individuals based on the size of the project and also give them more practice in programming skills that are indispensable for computer science students. Writing opportunities yield work products that provide students to refine writing skills and to produce scholarship, which they can reference when preparing for job interviews. Writing practice is particularly important for students pursuing the completion of a thesis or dissertation. The additional practice supports the writing of technical reports, a skill needed for transition to the industry. We have found that many upper-level engineering students do not have as much writing content to provide (i.e., as compared to freshman English and English composition classes). Research experiences give students a brief introduction to the entire process of research (i.e., problem, method, analysis, presentation of solution) and students have the opportunity to review scholarly articles or conference style papers that will inform their future writing practice [11].

### System Requirements

Based on the specification given by communications disorders design partners (Drs. Bailey and Speights Atkins), the following requirements were gathered for the development of an algorithm

for scoring the accuracy of phonetic transcriptions done by students within our learning management system:

1. Use the instructor-provided transcription (string1) as the model.
2. Strings consist of vowels, consonants, and diacritics. Diacritics always appear in combination with a vowel or a consonant, appearing as a one-character unit. Automatically include the diacritic as null if it is unspecified.
3. Find the optimal alignment of strings that minimizes the number of transformations (substitutions, deletions, and insertions) required to transform the model (string1) into the student's transcription (string2). In general, characters that are identical in both strings align with each other, except when that violates these conditions:
    a. Not allowed to change the order of characters in either string.
    b. Vowels may only align with vowels, and consonants may only align with consonants.
    c. Prioritize aligning vowels with each other if the number or order of consonants and vowels in the strings differs.
4. Perfect correspondence between the strings indicates a score of 0. Each transformation of a consonant (see list of consonants) or a vowel (see list of vowels) counts as a penalty of 1. If the diacritic changes, score .5. If the consonant or vowel and the diacritic changes score 1.5.

## The Problem

Assume you have to listen to an audio of a word (sound). You have to write down the phonetic representation of that word you heard of using a specialized phonetic coding system (IPA). Then someone else has to tell the difference between your answer and the real phonetic representation of that word (sound).

If it is about the difference between two words, then that is the classic edit distance. By definition, edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum spelling correction of operations required to transform one string into the other. One of the simplest sets of operations one can perform was defined by Levenshtein in 1966:

- Insert a character
- Delete a character
- Replace a character

For example, given the words A = "cat" and B = "cars", edit distance will be 2 since the minimum number of transformations needed to perform is to replace "t" in A by "r" and then delete the "s" from B. After that, both words will be car. One can also delete "s" in B first, then replace "r" in B as "t". That procedure will end up with both words as cat. And we can do other operations as well. All the distance in these cases will be the same, 2.

## Existing Solution

So how can we solve the problem? Let's start to solve a smaller and simpler problem. Assume that we have the previous two words (cat and cars), this time we treat the two words as a sequence of characters in the general form:

$$A = [A0, A1, \ldots Am]$$

B = [B0, B1, …, Bn]

Notice that initially, the lengths of A and B (m and n respectively) can be different.

We know that in the end, m and n will be the same since we want to transform one word into another. And on the other hand, one character at a given position must be the same. Now, imagine if we are dealing only with the first character of A and B, what choices do we have? We have 3 operations to perform, that means we have three choices:

1. We can insert a character into A to match the character in B[0], which has a cost of 1. After this operation, we still need to take care of the first character of A. However, we will go to the next character in B, since we already know A[0] is the same as B[0] now. The left job is to compute the edit distance for A[0…m] and B[1…n]. The final result will be the cost of insertion (1 in this example) plus the edit distance of A[0…m] and B[1…n].
2. We can delete a character in A to match B[0]. The cost is also 1. After this step, we have processed the first character in A, but we still have all the initial characters in B. Thus, we need to compute the edit distance of A[1…m] and B[0…n]. Again, the final result will be this value + 1 (the cost of deleting [0]).
3. The last choice we have is to replace the first character in A by the first character in B. The cost of this operation will be 1 if the two characters are different and 0 if they are the same. At this point, we only need to calculate the edit distance A[1…m] and B[1…n]. The final result will be 1 + the remaining edit distance of processing [1…m] and B[1…n].

Let's summarize what we have so far:

$$\mathrm{ed}(A,B) = \min\begin{cases}\mathrm{ed}(A[1\ldots n],\ B[1\ldots m] + \text{replace (a)}\\ \mathrm{ed}(A[1..n],\ B[0..m] + \text{delete (b)}\\ \mathrm{ed}(A[0\ldots n],\ B[1..m]) + \text{insert (c)}\end{cases}$$

The first (a) is the case when we replace A[0] as B[0]. (b) is corresponding to the deletion of A[0] and (c) is when we insert a character into A.

The base cases are:

editDistance(A, "") = length(A)

editDistance("", B) = length(B)

The recursive implementation of the above solution is in Algorithm 1.

Algorithm 1: Recursive version of edit distance

m: length of string A

n: length of string B

a. If the last character of the two strings is the same, do nothing. Ignore the last characters and get the distance for remaining substrings. So we recur for lengths of m -1 and n -1
b. Else, we consider all operations on string A. We have three options here and we obtain the minimum distance of them:
c. Insert: Recur for m and n - 1
d. Delete: Recur for m - 1 and n
e. Replace, Recur for m - 1 and n - 1

However, the time complexity of the above implementation is O(3n), where n is the maximum length of the two words. The problem here is could we do better than that? Of course, we can. However, the above implementation is letter-based implementation, which is not suitable for phonetic representations of words because a single phonetic representation may be represented by one, two, or even three letters. Then we run the edit distance algorithm to comparatively process two lists of strings. Hence comes the proposed solution.

## Proposed Solution: *String Preprocessing*

As mentioned above, the character-based solution is not suitable in our case, since one has to take the diacritics, diphthongs, and double consonants as a single unit instead of several characters. Therefore, the recipe is to group these thought groups together, counting as one single unit. The original version of this implementation was provided by our client, which was developed in Visual Basic (VB). It generated the correct results for the phonetic transcription but was a very complicated and long process with poor user-experience. The previous process was a tedious task of gathering hand-written data from students, and then our teachers manually entered all the students' answers into a Microsoft Excel sheet and exported the results out to the VB system to obtain the edit distances and grades for each student's assignment.

We also endeavored to improve the evaluation process and after thoughtful consideration, the system treats all the diacritics, diphthongs, and double consonants logically by dividing the initial input from the student into an array of characters, then combine the corresponding units into one position. As a result, instead of calculating the edit distances of two string representations of the words, we compute the edit distance of thought-group position-based edit distance. For instance, if we have the phonetic representation fɪŋgɚ͡neɪl, the result from the preprocessing will be [f, ɪ, ŋ, g, ɚ, n, e͡ɪ, l]. In this way, e͡ɪ will be in one single unit, as it should be. The time complexity for the preprocessing is O(*n*). The preprocessing is done by applying the preprocessing algorithm and, on the other hand, achieved by restricting the users to use only the keyboard provided by our implementation, for the reason that we only allow users to type legitimate input string. Otherwise, it is very complicated to do the tokenization step, since there are millions of possibilities of input strings.

> Algorithm 2: String preprocessing
>
> m: length of input string
>
> result = empty list
>
> The algorithm goes for 0…m:
>
> 1. If this character is belonging to a diacritic, a double vowel or consonant, based on the information in Table 1: find the whole group of characters, and add them as a group into the result list
> 2. Else (not in any of diacritics, double vowels, or double consonants): add this single character to the result list

## Dynamic Programming Implementation of Edit Distance

Once we obtain the list of strings from the preprocessing step, we can run the dynamic programming version edit distance, as the time complexity of recursive edit distance is O(3n). This algorithm is shown in Algorithm 3.

Algorithm 3: Dynamic programming implementation of edit distance

M[][] is the matrix to store the edit distance

For the list of strings S1 = [S1,0, S1,1, … S1,m]

For the list of strings S2 = [S2,0, S2,1, … S2,n]

M[i][j] = case 0: M[i-1][j-1] + 0 if S1[i-1] == S2[j-1]

case 1: M[i-1][j-1] + replaceCost if replace

case 2: M[i-1][j] + deleteCost if delete

case 3: M[i][j-1] + insertCost if insert

return M[S1.length][S2.length] as the final edit distance

Costs are calculated based on the requirements, shown in the requirements section.

## Results and Analysis

Let's look at one specific example here to illustrate the algorithms. Say the student's answer for a word is prɛd͡ʒ, while the key is bɹɪd͡ʒ. The first step is to get the list of strings for both of them. Thus, [p, ɹ, ɛ, d͡ʒ] and [b, ɹ, ɪ, d͡ʒ] for the above two words, respectively. Then the dynamic programming version of edit distance is run to get the difference (distance) for them. The matrix representation of the execution process is shown in Table 1. As we can tell from the matrix, the final distance is listed in the last cell. So, the distance is 3.0. For comparison, we list the test cases for the algorithm with and without the preprocessing procedure. As we can tell clearly from Tables 2 and 3, the preprocessing procedure will affect the words that have diacritics, double vowels, and/or double consonants. Without appropriate preprocessing, the edit distance will be inaccurate for most of the cases involving the above-mentioned scenarios.

**Table 1. The matrix representation of the distance of prɛd͡ʒ and bɹɪd͡ʒ. The bolded values make up the final transformation path.**

| | . | b | ɹ | ɪ | d͡ʒ |
|---|---|---|---|---|---|
| . | **0.0** | 1.0 | 2.0 | 3.0 | 4.0 |
| p | 1.0 | **1.0** | 2.0 | 3.0 | 4.0 |
| r | 2.0 | 2.0 | **2.0** | 3.0 | 4.0 |
| ɛ | 3.0 | 3.0 | 3.0 | **3.0** | 4.0 |
| d͡ʒ | 4.0 | 4.0 | 4.0 | 4.0 | **3.0** |

**Table 2. Test cases for edit distance results without preprocessing procedure. For the path, we use 's' to denote replace, 'i' to indicate insert, '|' to mean same consonant, and '*' to denote the same vowel.**

| Case # | key | Answer | Path | Distance |
|---|---|---|---|---|
| case1 | dʌk | dæt | \|ss | 2.0 |
| case7 | keɪd͡ʒ | keɪd͡ʒ | \|***\|* | 0.0 |
| case8 | geɪ̯t | geɪ̯t | \|***\| | 0.0 |
| case9 | kɪŋ | kɪŋ | \|* | 0.0 |
| case10 | fɪs | geɪ̯t | %is*s | 4.0 |
| case11 | fænd | fænt | \|*\|s | 1.0 |
| case12 | d͡ʒaɪ | d͡ʒɔr | \|*\|ss | 2.0 |
| case13 | t͡ʃiz | t͡ʃiz | \|*\|* | 0.0 |
| case14 | wat͡ʃ | wat͡ʃ | \|*\|* | 0.0 |
| case15 | klaʊ̯n | klaʊ̯n | \|\|***\| | 0.0 |
| case16 | glʌv | gləv | \|\|s\| | 1.0 |
| case19 | bɹɪd͡ʒ | prɛd͡ʒ | sss\|*\| | 3.0 |
| case21 | bæskɪtbɔl | bæskɛtba | \|*\|\|s\|\|ds | 3.0 |
| case22 | pɛɹəʃut | peɪ̯rəʃut | \|iiss*\|*\| | 4.0 |

**Table 3. Test cases for edit distance results with preprocessing procedure. For the path, we use 's' to denote replace, 'i' to indicate insert, '|' to mean same consonant, and '*' to denote the same vowel.**

| Case # | key | Answer | Path | Distance |
|---|---|---|---|---|
| case1 | dʌk | dæt | \|ss | 2.0 |
| case7 | keɪd͡ʒ | keɪd͡ʒ | \|** | 0.0 |
| case8 | geɪ̯t | geɪ̯t | \|*\| | 0.0 |
| case9 | kɪŋ | kɪŋ | \|*\| | 0.0 |
| case10 | fɪs | geɪ̯t | sss | 3.0 |
| case11 | fænd | fænt | \|*\|s | 1.0 |
| case12 | d͡ʒaɪ | d͡ʒɔr | *ss | 2.0 |
| case13 | t͡ʃiz | t͡ʃiz | **\| | 0.0 |
| case14 | wat͡ʃ | wat͡ʃ | \|** | 0.0 |

| | | | | |
|---|---|---|---|---|
| case15 | kla͡un | kla͡un | ‖*‖ | 0.0 |
| case16 | glʌv | gləv | ‖s‖ | 1.0 |
| case19 | bɪɪdʒ͡ | prɛdʒ͡ | sss* | 3.0 |
| case21 | bæskɪtbɔl | bæskɛtba | \|*‖s‖ds | 3.0 |
| case22 | pɛɹəʃut | peɪ͡rəʃut | \|ss*\|*\| | 2.0 |

## Conclusions

During the UID course, students were instructed in the theory and practice of designing and developing interactive systems, which reinforces student's software engineering skills through programming and development practice and illustrates service-learning. These case-based service-learning problems provide a rich experience and this work focuses on the APTgt system requirements, design, development and preliminary evaluation. The research question that we addressed was, "How do we provide more timely and consistent feedback for students on their linguistics transcriptions with feedback on their transcription goodness and accuracy?". Our focus was to develop an engaging, easy to use platform that reduces effort for instructors and support interactive online learning with quick feedback. In this research, we have proposed and developed a version of a modified edit distance to calculate the distance between two phonetic representations of a word (sound), where the phonetic representations may contain diacritics, double consonants, and/or double vowels. With the preprocessing procedure to split the input strings for the edit distance algorithm into a list of strings based on the scoring requirements provided by the clients, we obtain correct edit distance for the phonetic representations. During this project, we have improved the user experience for two university linguistics professors and their students by improving their students' efficacy and learning of transcription with the support of a tool for the Automatic Phonetic Transcription grading tool - APTgt. Our hope is that this tool becomes a standard for all linguistics professionals that utilize transcription in their teaching and practice.

## References


1 Cheng, I., and Basu, A., & Goebel, R., (2009). Interactive multimedia for adaptive online education, *IEEE Multimedia magazine*, 16-24.

2 Seals, C., Zhang, Q., & Cook, T. (2017). An M-Learning Application to Enhance Children's Learning Experience, October 16 Vol. 4 No. 10, *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC)*, ISSN: 2321-8169, 214 – 222.

3 Tripp, L.O., Seals, C. D., & Bassy, R. Spectrum Educational Tool: Animated Scenarios for Teacher Preparation. *Journal* of Online Learning Research and Practice. Vol. 7, No. 2, 15-36.

4 Margaryan, A., Littlejohn, A. & Vojt, G. (2011). Are digital natives a myth or reality? University students' use of digital technologies, *Computers & Education*, Volume 56, Issue 2, 429-440. https://doi.org/10.1016/j.compedu.2010.09.004

5 Nagler, W. & Ebner, M. (2009). Is your university ready for the Ne(x)t-Generation? *Proceedings of 21st world conference on educational multimedia, hypermedia and telecommunications (EDMEDIA)* (June 22–26, 2009), 4344-4351. Honolulu, Hawaii, USA.

6 Nemo, J. (2016). The $107 Billion Industry That Nobody's Talking About. Inc.com retrieved from https://www.inc.com/john-nemo/the-107-billion-industry-that-nobodys-talkingabout.html

7 Anshari, M., Alas, Y. & Guan, L. S. (2015). Developing online learning resources: Big data, social networks, and cloud computing to support pervasive knowledge. *Education and Information Technologies*, 21(6), 1663–1677. doi:10.1007/s10639-015-9407-3

8 Kazmer, M. M., & Haythornwaite, C. (2004). Multiple perspectives on online learning*, ACM SIGGroup Bulletin*, 25(1), 7-11.

9 Chan, A.W. K., Chair, S.Y., Sit, J.W.H., Wong, E.M.L., Lee, D.T.F. & Fung, O.W.M. (2016). Case-based web learning versus face-to-face learning: A mixed-method study on university nursing students. *The Journal of Nursing Research,* 24(1), 31Y40. doi:10.1097/jnr.0000000000000104

10 Bae, B., Kang, S. S. & Hwang, B. Y. (2012). Edit distance calculation by phonetic rules and word-length normalization. *Proceedings of the European Computing Conference (ECC'12)*, Prague, Czech Republic, 315-319.

11 Speights Atkins, M**.,** Seals, C., Bailey, D. J. (2019) At the intersection of applied sciences: Integrated learning models in computer science and software engineering and communication disorders. *Science Education and Civic Engagement: An International Journal*, 11(1), 37-43.

12 Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10, No. 8, 707-710.

13 Heselwood, B. (2007). Teaching and assessing phonetic transcription: a roundtable discussion. *Centre for Languages Linguistics & Area* Studies https://www. llas. ac. uk/resources/gpg/2871 DOA, 16, 2015.

14 Speights-Atkins, M., Seals, C.D. & Bailey, D. (2018). The Automated Phonetic Transcription Grading Tool: Where Computer Science Meets Clinical Problem Solving in Communication Disorders, SENCER Summer Institute August 2018. National Center for Science & Civic Engagement *(highly acknowledged in Education Scholarship)*.

15 Speights-Atkins, M., Seals, C.D. & Bailey, D. (2018). The Automated Phonetic Transcription Grading Tool: Where Computer Science Meets Clinical Problem Solving in Communication Disorders*, SENCER Summer Institute,* August 2018. National Center for Science & Civic Engagement.

16 Knight, R. A. (2010). Sounds for study: Speech and language therapy students' use and perception of exercise podcasts for phonetics. *International Journal of Teaching and Learning in Higher Education*, 22(3), 269-276.


## Author Bios

**Sicheng Li** is a PhD candidate at the Department of Computer Science and Software Engineering, Auburn University. He has solid background in both computational science and Computer Science. He currently is a Human Computer Interaction researcher at Dr. Seals' group. His research focuses on developing a full-stack web application of phonetic transcription examination system and measuring learning outcomes for students of phonetic transcription, and also enhancing students understanding on Computational Thinking. He is strong at algorithmic thinking and implementations and offering software solutions.

**Marisha Speights Atkins**, Ph.D., CCC-SLP, is an Assistant Professor in the Department of Communication Disorders at Auburn University. Her current research focuses on the development and translation of innovative acoustic-based tools for assessing speech production, speech intelligibility and expressive language disorders in preschool age children. She works in collaboration with computer science and software engineers to research and develop computer-based approaches for identifying speech production differences between children who acquire speech according to expected developmental trajectories and those with speech sound disorders that affect intelligibility.

**Dallin J. Bailey**, PhD, CCC-SLP, is an Assistant Professor in the Department of Communication Disorders at Auburn University. His research interests involve various aspects of assessment of and treatment for aphasia and apraxia of speech. He is also interested in the use of mental practice in speech motor learning in healthy individuals. He received his PhD from the University of Utah.

**Jueting Liu** is a PhD student followed Dr. Seals in CSSE, Auburn University. He used to work and do research in VR/AR group and he currently focus on the backend and server technology. Algorithm is also an important part in his research.

**Yang Cao**, PhD, Research Scientist at Facebook performed her graduate research in Computer Science in the area of Human Computer Interaction, focusing on Data Visualization and Data Analysis. She has worked on several projects in iOS Development and Web Development (e.g. Machine learning to support Automatic Phonetic Transcription grading tool (APTgt) exam creation, package tracking app, website for AMNSTC, personal budget website, etc.).

**Robertson Bassy** is a third-year doctoral student who focuses on learning technologies, linguistics, and data manipulation. Mr. Bassy interests focus on the advancement of linguistic technology and implementation of different platforms that include mixed reality.

**Cheryl D. Seals** is a professor of Computer Science & Software Engineering. She has taught and developed eight courses during her tenure and dozens of software systems through iterative development and participatory design related to her background in Human-Computer Interaction and Usability Evaluation. Dr. Seals has worked as a leader in the field of computing and supports initiatives for computing retention and outreach scholarship programs and other initiatives to improve computing education at all levels with over $2 million in funding in the area of computing technologies and outreach scholarship to support areas of under-representation in computing (e.g. support the recruitment and retention of women and under-represented groups in engineering). Dr. Seals has published over 70 research and outreach publications, and more than 50 presentations either research or supporting the recruitment and retention of students in STEM fields. With respect to graduate mentoring, Dr. Seals has graduated 8 Computer Science PhDs and numerous Masters and undergraduate students. She has also supported regional academic partnerships, STARS Alliance (starscomputingcorps.org) with over 50 institutions, industry, K-12 and the community to strengthen local broadening of computing by focusing on K-12 outreach, community service, student leadership and computing diversity research. Dr. Seals works with many programs focused on increasing the computing pipeline by getting students interested in STEM disciplines and future technology careers.