Ahadu: Educational Object Oriented Programming Language for Amharic speakers

Hailemichael Alemneh
Department of Computer Science, Stanford University
CS 191: Senior Project
Prof. Sara Achour
Fall 2023

Abstract

People from nearly every country are now learning computer programming as programming emerges as a pivotal skill fostering innovation, effective problem-solving, and career advancement. However, the path to mastering programming is not without its challenges, particularly for non-native English speakers. English, being the predominant language in education, creates a barrier for learners whose primary language differs from this linguistic standard. Through literature review, this paper explores the challenges faced by non-native English speakers in the field of programming, where English is the predominant language, and argues for the use of programming languages based on students' native languages as a solution to enhance learning experiences and create opportunities for those deterred by English language barriers. The paper also explores Ahadu, an Amharic-based programming language, I developed to serve as an educational tool for Amharic speakers. The focus is on Ahadu, an Amharic-based programming language developed to address these challenges.

Introduction

In an era where digital literacy is becoming synonymous with empowerment, programming stands out as a critical skill, facilitating innovation, problem-solving, and career opportunities. However, the path to mastering programming is not without its challenges, particularly for non-native English speakers. English, being the predominant language in education, creates a barrier for learners whose primary language differs from this linguistic standard.

This linguistic challenge is compounded by the fact that, while the platonic ideal of a programming language may be rooted in pure mathematical logic, the reality is that programming languages, along with their expansive ecosystems of APIs, frameworks, IDEs, and online communities, are intricately entwined with the English language. Yet, 95% of the world's population does not consider English their native language, underscoring the global scale of this linguistic discrepancy within the field of programming (Guo, 2018).

This paper aims to substantiate the argument that using programming languages based on the students’ native language can help address these challenges and contribute to a more effective learning experience and beyond mere proficiency, can create opportunities for students who may have been deterred from delving into programming due to English language barriers. The paper will also focus on Ahadu, an Amharic-based programming language I developed, as an example project. The thesis is structured to navigate through various layers of this argument, drawing upon existing literature to provide an understanding of the multifaceted issues at play.

The first section of the paper will explore the challenges faced by students when engaging with education in a non-native language. The focus will then shift to programming-specific challenges that are exacerbated by the reliance on English-based programming languages, emphasizing the cognitive load and comprehension difficulties associated with this linguistic misalignment.

The subsequent section will delve into existing literature comparing the educational outcomes of students learning programming in their native language versus those using English-based languages. By examining these studies, I seek to establish a compelling case for the efficacy of education in a native language-based programming language in fostering a more inclusive and supportive learning environment.

And the final section will delve into Ahadu an Amharic-based programming language I developed to serve as an educational tool for Amharic speakers. I will examine the design decisions and the structure of the language and how it plans to serve Amahric speaking students, new to programming, to write, read and comprehend code better.

Literature Review

English stands as the predominant language of instruction in university, or even high school, classrooms worldwide, even for those whose native language differs. This universal adoption is influenced by factors like the prevalence of English-based textbooks, the use of English as a common working language in institutions with students from diverse linguistic backgrounds, and students' aspirations for roles in English-speaking multinational corporations (Hammer et al., 2016). Moreover, the dominance of English extends to the majority of online resources (Guo, 2018).

It has been shown in multiple studies that students, whose native languages are not English or are not proficient in English, face challenges when English is used in classrooms when learning a wide array of topics including math, science, engineering, medicine, and the humanities. Barriers range from cognitive to affective to social: Needing to mentally translate concepts into one's native language—especially in real-time while listening to a lecture—increases extraneous cognitive load and decreases comprehension (Sweller, 1994). The difficulty of formulating verbal questions, along with anxiety about lack of English fluency, makes them less likely to ask clarifying questions (Probyn, 2001).

A large body of education research and theory suggests that learning in one's local language at the primary and secondary levels supports positive learning outcomes. As early as 1953, UNESCO's publication on "The use of vernacular languages in education" argued for instruction in students' mother tongues both as early and as late, as possible, "on educational grounds, we recommend that the use of the mother tongue be extended to as late a stage in education as possible. In particular, pupils should begin their schooling through the medium of the mother tongue because they understand it best …"

Beyond the general use of English as a Second Language barrier described above, programming adds an extra layer of complexity since English is deeply intertwined into programming language keywords and API naming conventions. Coding style guidelines mandate writing not only comments but also all identifiers (e.g., variable, function, and class names) in English. (Guo, 2018).

In a study done on students at Sebha University, in Libya, and CQUniversity, in Australia, the Libyan students' survey, which asked for specific issues they faced while learning programming, revealed that there was a difference between these Libyan students and the CQUniversity students, and the researcher identified one problem (error messages) that was particularly influential on the ability of novice programmers to learn programming language at the main Sebha University campus. Two other issues of importance for these students were pointers and loop statements (Do...While). Error messages being indicated by students so frequently as an issue is likely not surprising given that each error message, in English, must be interpreted by someone with limited English ability. In contrast, the results from CQUniversity (Table 1) suggest that the main issues for novice programmers in learning programming are loop statements

(Do...While), all programming concepts, and program design (Nnass, Colwing, and Hadgraft, 2022).

Another survey done on a programming education website that spans 86 countries and 74 native languages found the most commonly reported barrier (34% of all responses) was problems reading English instructional materials such as textbooks, online tutorials, discussion forums, and API docs. Respondents reported the usual barriers to reading in a non-native language, such as slower reading comprehension. Technical jargon associated with programming likely made comprehension more challenging than casual prose reading. In addition to problems reading prose, non-native English speakers also reported trouble understanding source code. A common root cause is that English programming language keywords are in English. Beyond built-in keywords, programmers often use abbreviations and idiomatic naming conventions for identifiers (e.g., function, class, and variable names), which can be hard for non-native speakers to understand. Since non-native speakers must often rely on English language instructional materials, some reported trouble with learning enough English to comprehend those materials while simultaneously learning the given programming concepts. (Guo, 2018)

This begs the question: would the use of programming languages and resources based on the students' native language address the challenges listed above? While some studies are showing the advantages of using resources, like textbooks, and lectures, based in native languages are beneficial for students, limited studies are comparing the use of English-based programming languages and native language-based programming languages for education, especially for students being introduced to programming.

One of the explanations for the lack of studies exploring this question is the limited availability of non-English-based programming languages. Despite prior work in HCI highlighting the need for localized user interfaces, localization can be complex and expensive (Alabau and Leiva, 2015). As a result, many online learning systems are offered only in English. When interfaces are localized, it is often only into a small number of languages spoken by large populations. For speakers of other languages, especially for those belonging to small or marginalized linguistic groups, the only option is to use English (Dasgupta and Hill, 2017).

There have been attempts to create programming languages with non-English keywords, but so far none have been widely adopted. Attempts to translate APIs and error messages have faced a similar lack of adoption since programmers cannot as easily search for online help using localized terms (Guo, 2018). One promising line of work in this direction is to localize keywords and UI components in blocks-based languages for novices such as Scratch (Dasgupta and Hill, 2017).

Dasgupta and Mako Hill analyzed usage logs from five countries and found that learners who used a version of Scratch in their native language could more rapidly build more complex programs (a proxy for progress in learning) than those who used the default English version. Scratch is a programming language and online community designed for children aged 8-16. In Scratch, programs are constructed by dragging and dropping visual blocks to define the behavior of on-screen graphical objects called sprites. The Scratch language is supported by a large online community launched in 2007, where creators can share their projects, comment on each others' work, and remix projects created by their peers. Although participation in Scratch is generally open-ended and self-directed, Scratch is used in formal educational settings as well. (Dasgupta and Hill, 2017)

Dasgupta and Mako Hill present models that estimate a positive association between the growth rate of users' repertoires of programming blocks and the translation of their programming language and interface into their local languages indicating a more rapid learning in students using their native-language translation of Scratch. But more important is the effect that localization has on the degree to which engagement and learning are possible in the first place (Dasgupta and Hill, 2017). After all, Scratch is almost unique among programming languages in that it provides a completely localized interface, including a translated version of the programming language itself. Although evidence has been presented in support of the claim that young programmers learn more quickly using localized interfaces, the most important effect of localization, not captured by the analysis, maybe that being able to engage in one's primary language supports users who would otherwise not learn to code at all.

Ahadu

Ahadu is an Amharic, an Ethiopian Semitic language that serves as the official working language of the Ethiopian federal government with over 32.4 million mother-tongue speakers and more than 25.1 million second-language speakers as of 2019, based programming language I developed for computer programming education purposes. Ahadu aims to serve as an educational tool for Amharic-speaking individuals interested in learning programming. It also aims to introduce Amharic speakers previously not exposed to the concepts of programming or discouraged from learning programming due to language barriers to computer programming. As such, it is designed with beginner programmers in mind. It is structured to be comprehensible for Amharic speakers by mimicking Amharic sentence structures. The programming language is designed and built with the assumption the user knows little to no English. This means all the reserved keywords are in Amharic. These are not mere translations of the English keywords from other programming languages but words that are selected to represent the concept of the keywords and be as easily accessible and comprehensible to Amharic speakers as possible. Variable names, strings, and so on will also support the Geez Fidel letters in which Amharic is written, in addition to English letters. Error messages outputted by the interpreter are also in Amharic.

Conclusion

In conclusion, the paper emphasizes the critical need to address language barriers in programming education, particularly for the majority of the global population that does not consider English their native language. The challenges faced by non-native English speakers, as outlined in the literature review, underscore the necessity for a more inclusive approach. The development of Ahadu, an Amharic-based programming language, serves as a tangible example solution to bridge this linguistic gap. By utilizing native language programming languages, such as Ahadu, educators and learners can potentially alleviate cognitive load, enhance comprehension, and foster a more supportive learning environment. The paper suggests that initiatives like Ahadu contribute not only to individual proficiency but also open doors for a wider demographic to engage with programming, promoting diversity and inclusivity in the field. As the programming landscape continues to evolve, exploring and embracing linguistic diversity in education becomes imperative for unlocking the full potential of aspiring programmers worldwide.

References

Alabau, Vicent and Luis A. Leiva. “Automatic Internationalization for Just In Time Localization of Web-Based User Interfaces.” *ACM Transactions on Computer-Human Interaction* 22, no. 3 (May 27, 2015): 13:1-13:32. https://doi.org/10.1145/2701422.

Dasgupta, Sayamindu, and Benjamin Mako Hill. “Learning to Code in Localized Programming Languages.” In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, 33–39. Cambridge Massachusetts USA: ACM, 2017. https://doi.org/10.1145/3051457.3051464.

Guo, Philip J. “Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities.” In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–14. CHI ’18. New York, NY, USA: Association for Computing Machinery, 2018. https://doi.org/10.1145/3173574.3173970.

Nnass, Ibrahim, Michael A. Cowling, and Roger Hadgraft. “Identifying the Difficulties of Learning Programming for Non-English Speakers at CQUniversity and Sebha University.” *Journal of Pure & Applied Sciences* 21, no. 4 (October 3, 2022): 290–95. https://doi.org/10.51984/jopas.v21i4.2258.

Probyn, Margaret. “Teachers Voices: Teachers Reflections on Learning and Teaching through the Medium of English as an Additional Language in South Africa.” *International Journal of Bilingual Education and Bilingualism* 4, no. 4 (December 1, 2001): 249–66. https://doi.org/10.1080/13670050108667731.

Soosai Raj, Adalbert Gerald, Kasama Ketsuriyonk, Jignesh M. Patel, and Richard Halverson. “Does Native Language Play a Role in Learning a Programming Language?” In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 417–22. SIGCSE ’18. New York, NY, USA: Association for Computing Machinery, 2018. https://doi.org/10.1145/3159450.3159531.

Sweller, John. “Cognitive Load Theory, Learning Difficulty, and Instructional Design.” *Learning and Instruction* 4, no. 4 (January 1, 1994): 295–312. https://doi.org/10.1016/0959-4752(94)90003-5.

Uchidiuno, Judith, Amy Ogan, Evelyn Yarzebinski, and Jessica Hammer. “Understanding ESL Students’ Motivations to Increase MOOC Accessibility.” In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, 169–72. L@S ’16. New York, NY, USA: Association for Computing Machinery, 2016. https://doi.org/10.1145/2876034.2893398.

Wang, Patrick. “PseuToPy: Towards a Non-English Natural Programming Language.” In *Proceedings of the 17th ACM Conference on International Computing Education Research*, 429–30. ICER 2021. New York, NY, USA: Association for Computing Machinery, 2021. https://doi.org/10.1145/3446871.3469787.