Learning Latin Noun Endings With App Assisted Spaced Repetition

Eric Ianni
Georgia Institute of Technology, OMSCS
ebianni@gmail.com

Let's be honest, Latin is hard. In English, verb conjugating is rather simple (excluding irregular verbs). Of the six person/number pairings for verbs, English has only two forms (e.g. *run* and *runs*). So for a native English speaker, Latin's confounding insistence on having six forms for each verb tense ***and*** six forms for each noun, can seem outright daunting. The Romans succeeded in teaching their children this rudely complicated language using only wax tablets and recitation. With the technological advances that have taken place during the fifteen hundred years since the fall of Rome, there must be a better way!

Almost from the beginning of the computer age, developers have been creating tools to help people learn a new language (Hart 1995). Those efforts have continued into the present day, where students and teachers alike can choose from multiple pieces of software. Modern foreign language textbooks are often packaged with custom companion software. Mobile applications exist for most of the *popular* languages allowing learners to study on-the-go. Sadly, Latin is not often included as one of those *popular* languages.

Statement of the problem

In 2016, 155,258 students took AP Spanish, which made it the most popular language course. In the same year, there were only 6,584 AP Latin students: less than five percent of AP Spanish students. Based upon these numbers and considering that many people consider Latin a "dead" language, it is no surprise that there are more educational software tools available for Spanish than Latin. For example, Rosetta Stone Inc. offers five levels of its language software for most languages while only offering three levels for Latin. A popular online and mobile language tool, Duolingo, currently offers lessons in more than twenty languages. Duolingo also has plans for additional languages including Klingon (a fictional language), with no plans for a Latin module (Duolingo: Language Courses for English Speakers 2017).

This lack of available tools and software makes learning and studying Latin more difficult compared to more modern languages. Teachers of Latin have fewer technological resources available to supplement their instruction. Furthermore, compared to independent learners of "living" languages, Latin self-study students have few software tools designed specifically for them. Finally, without online and mobile tools, Latin lacks a key method of reaching a modern audience that enjoys learning on-the-go (Kukulska-Hulme 2009).

Existing Solutions

Rosetta Stone

The only fully realized commercial offering of Latin learning software is Rosetta Stone. This software follows the immersion method of learning a language. The program begins by presenting users with a series of images without any English as shown below (Figure 1).

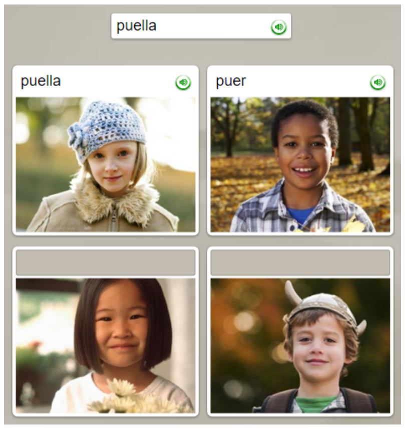

Figure 1: Rosetta Stone Introduction Screen

Using the pictures and Latin text, the user is expected to infer that *puella* and *puer* mean *girl* and *boy* respectively. From there, the software adds further words as shown below (Figure 2). The user knows that *puella* means *girl* so he or she can guess at the meaning of the provided Latin: the *girl* eats and the *girl* drinks.

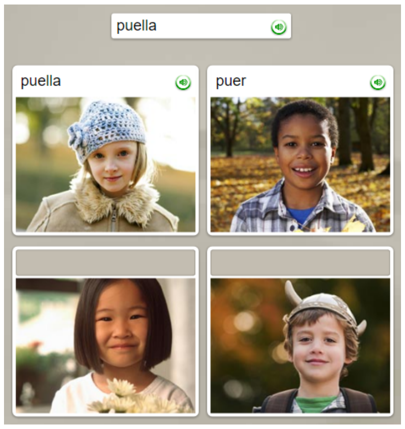

Figure 2: Rosetta Stone Second Screen

Rosetta Stone's software never has the user translate Latin to English, instead relying solely on images and Latin text. The immersion technique is highly regarded as it is seen to mimic how humans learn their first languages as children.

Rosetta Stone's immersion approach works well with modern languages, but not so much with Latin. As mentioned before, Latin is a "dead" language, and that means outside of a few dedicated academics, it is not a spoken language. Most Latin students will only experience Latin as text on a page, which is a different skill than speaking and listening. Furthermore, Latin is perhaps one of the more complicated languages when you take into account the vast number of endings for nouns, verbs, and adjectives. Finally, much of Latin instruction relies on source texts from antiquity which have very specific vocabularies and grammatical styles. Rosetta Stone's software does not take this into account and instead teaches Latin as any other spoken language. This means that a user who completed Rosetta Stone Latin will know how to say *breakfast* in Latin, but would not be able to read *Caesar's Bellum Gallicum*.

Even for the modern languages, Rosetta Stone really doesn't offer anything new outside of the immersion method. The software is just a digital version of walking around a foreign country and learning by observing. It does not actually leverage the computing power of computers.

Computer Managed Practice

Before discussing a few examples of how software can help manage learning, it is important to discuss one theory of how memory is gained, reinforced, and potentially lost.

The Forgetting Curve and Spaced Repetition

Learning a foreign language requires memorizing thousands of pieces of individual knowledge: syntax and vocabulary. Many students struggle with storing this information in long term memory. Psychologists have long studied how human memory works.

Herman Ebbinghaus was such a psychologist. He developed the concept of the *forgetting curve* and *spaced repetition* (SR) in 1885 (Settles and Meeder 2016). His own work was built upon the work of Adolph Jost, one of whose laws described how older learned material decayed slower than newly learned material. Ebbinghaus further theorized that a memory would fade exponentially as time passed(A Dictionary of Psychology 2008). Ebbinghaus described the *forgetting curve* with the equation below:

$$R = e^{\frac{-t}{S}}$$

where R represents memory retention, S is the strength of the memory, and t is how much time has passed since the knowledge was learned or strengthened.

Ebbinghaus also discovered that an individual could flatten the slope of the curve by reinforcing the knowledge through repetitions, which was the foundation of SR. With each new exposure to the material, the memory would fade slower, as shown below (Figure 3).

During the next hundred years, psychologists would refine the work of Ebbinghaus. Studies would later discover that as the time between repetitions, the *lag*, increased, the memories seemed to last longer (Bahrick et al. 1993). But these same studies also determined that if the *lag* became too great, the memory could be lost.

Multiple people attempted to discover a standardized system to manage the optimal *lag* between reviews. One such system of SR was developed by Sebastian Leitner in the 1970's, called

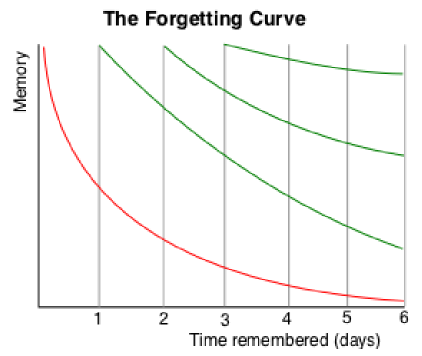

Figure 3: https://en.wikipedia.org/wiki/Forgetting_curve

the "learning box." In his system, students would make flashcards and use a series of envelopes or compartments in a box to determine when each card needed to be reviewed.

All the cards started in the first envelope and the student attempted to answer each. Each card answered correctly would be moved up one envelope and each incorrectly answered card would return to the first envelope. Along with the box, the students had a schedule of days for the month that identified which envelopes to review on that particular day.

The first envelope was reviewed each day and any others the schedule identified. As the envelopes went up in number the frequency of review decreased and therefore increased the *lag* and eventually the cards would "graduate" from the entire box and would be considered part of permanent memory (Godwin-Jones 2010).

While systems such as Leitner's "learning box" helped students learn facts and vocabulary more effectively, they were still not based on much data and were very inflexible. It was not long before the programmers began experimenting with better ways of calculating the optimal *lag*.

SuperMemo

Perhaps the most famous and successful of these programmers is Piotr Wozniak, who, in 1987, invented the SuperMemo program. Wozniak lived in the USSR and was attempting to learn English. Being the hyper-analytical man he was, Wozniak decided to use years worth of data to best calculate when each piece of knowledge should be reviewed so as not to forget anything.

SuperMemo was a computer version of his personal pencil-and-paper approach to SR. Wozniak realized that he could never keep track of all the pieces of information he learned **and** calculate the appropriate *interval*, his term for *lag*. By writing the SuperMemo software, Wozniak was able to apply his calculations to individual facts and not just large groups of material.

After analyzing years of data from his own learning, Wozniak developed what he called the SuperMemo 2 algorithm (SM-2).

$$I(1) := 1$$
$$I(2) := 6$$
$$I(n) := I(n-1) * EF \ for \ n > 2$$

where $I(n)$ is the *interval* after the nth repetition, EF is the *easiness factor* of memorizing and maintaining a piece of knowledge. After each repetition, the user must identify how easy the question was on a scale from 0 to 5 (0 being a total blank). The SM-2 would then use this *quality* rating, q, in the following equation:

$$EF' := EF + (0.1 - (5-q) * (0.08 + (5-q) * 0.02))$$

where EF' is the newly calculated *easiness factor* while EF was the value before the repetition.

Using SM-2, Wozniak was able to memorize 10,255 items with an adjusted retention rate of 92% (Wozniak 2017). Since 1987, SuperMemo has been updated and the algorithm revised (currently at SM-17).

Anki and Cerego

Piotr Wozniak has never hidden his research or his algorithms from the world. In fact, he has repeatedly shared and explained in great detail his work. For many years Wozniak's SuperMemo was considered the gold standard for SR learning. Over the years there have been multiple applications that have used Wozniak's work. Two such software tools are Anki and Cerego. Both exist as web and mobile apps.

Of the two, Anki is the most similar to SuperMemo. In the program, the user is presented with a question and the option to show the answer when ready. On the answer screen the user must rate how well he or she knew the answer: again, good, and easy. Each option also shows the user how long Anki will wait before having the user review that card again.

The "again" option will always repeat in minutes, "good" can be anywhere from minutes to days, and "easy" starts at days and can go up to multiple months. Based upon past performances by the user, Anki uses an algorithm similar to SuperMemo's to calculate the displayed *intervals*.

While Anki is a open source tool, Cerego is designed as a commercial product. This means Cerego has a much nicer interface and features, but ultimately uses SR in a similar manner to Anki.

Cerego also uses "decks" of information to practice. The user chooses how many items and subjects to practice each day and Cerego does the rest. Like Anki, the user is presented with a question, but unlike Anki Cerego asks the user if he or she knows the answer.

If the user selects "Don't Know It," Cerego immediately shows the answer. If the user selects "Know It," Cerego prompts the user to either pick from a list or input the correct answer. Cerego then lets the user know if he or she is correct before proceeding.

The thing that sets Cerego apart from Anki is that the user never has to identify how well he or she knew the answer, other than "know" and "don't know." The software infers this metric by analyzing how long the user took to select or input the answer and how often he or she is correct.

Both of these learning tools leverage the power of computing to enhance the learning process. Using the mobile apps for each, users are able to learn on the go. This is very important for many younger users who want to maximize their downtime by studying in short sessions while commuting, waiting for an appointment, or even during commercial breaks (Stockwell and Hubbard 2013). These two tools are also able to maximize the user's time by handling all the calculations for when to review each piece of knowledge. With the tools letting the user know when he or she is about to forget something, the user doesn't waste valuable time reviewing material that is still likely fresh in memory.

Limitations

Both Anki and Cerego are subject agnostic and provide no specialization towards any type of knowledge. They can be just as easily used for learning historical dates as the syntax for a new programming language. As mentioned before, the cards for each tool are organized into decks. These decks can either be downloaded from a database or created by the user.

For popular subjects like Biology and Spanish, there are dozens of decks to choose from already in the databases. But considering the small number of Latin learners, both services offer limited options for Latin decks and most of those that are offered are dedicated to a particular textbook.

Furthermore, since Anki and Cerego are subject agnostic, the user interface must be very generic. No specialization can be made to tailor the experience towards Latin. For example, Latin uses *macrons* to identify which vowels are long. Inputting these characters often requires non-standard input methods on both computers and mobile devices. This means that answering Latin knowledge questions is often awkward and focuses on recognition instead of recall.

Proposed Solution

After my research into SR and the *forgetting curve* I became convinced that they were ideally suited for learning and practicing Latin syntax. None of the existing tools had everything I felt was needed to properly assist Latin learners. Therefore I decided to create a proof-of-concept using the following criteria:

- The tool will be a mobile app so it can be used anywhere
- The app must leverage the power of SR to calculate the ideal *interval* between reviews
- The app must calculate the ideal *interval* without any user input
- The app must specifically cater to the needs of the Latin language
- The app must be textbook agnostic so any Latin student can benefit
- The app must inform the user of the strength of each piece of knowledge intuitively and let the user know what requires immediate review
- The app must allow practicing just what needs reviewing **and** allow the user to customize a review session by picking from multiple topics
- The app will test users on concepts, not individual words
- The app will only cover noun endings to keep the scope manageable

For the proof-of-concept I decided to design and build an Android app. The decision to use Android was multifold. First, I have only ever owned Android devices. Second, Android has the largest share of the market. Third, Android development is based on Java with which I was already familiar.

Interface

For the layout of my app I looked for inspiration from my research into existing solutions. The best mobile language learning app I have found, and one that I use personally, is Duolingo. Though not mentioned above, Duolingo does use SR for its app, but one that is currently based on machine learning and not the SuperMemo algorithm. More importantly for designing my own app, Duolingo's interface is highly intuitive and conveys a great deal of information to the user without cluttering the screen. Below you can see an example of Duolingo's home screen (Figure 4).

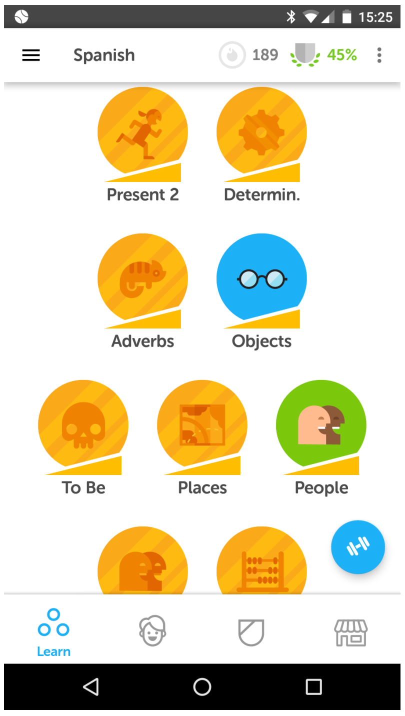

Figure 4: Duolingo's home screen

Duolingo lets the user immediately know the status of each lesson's memory strength. Items in gold are considered strong and require no review at this time. Items in full color are weakening and the user can clearly see the memory strength by looking at the bar at the bottom of each image. Clicking on any of the icons will give the user the opportunity to review each lesson. Once complete, the practiced lesson will increase in memory strength.

For my app I wanted to use a similar layout with visual clues. The home screen of my app would have an icon for each person/number pairing organized by noun declension and gender. Figure 5 shows my initial mock-up for my app's home screen. The user can press any of the options and be taken immediately to a quiz on that one topic.

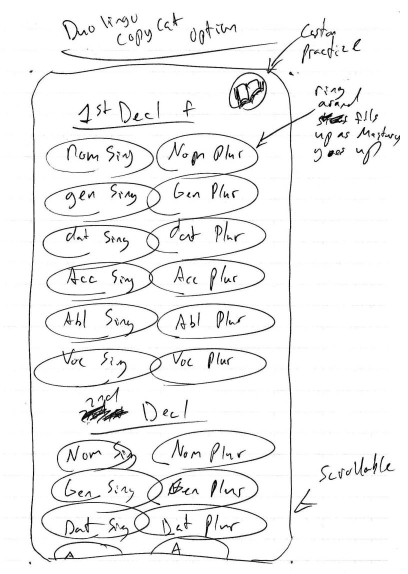

Figure 5: Home screen mock-up

When designing how the user could select a custom practice session, I found inspiration in another Latin app called *Vice Verba*. While this app didn't meet my criteria for a successful study tool, it did offer a wonderful interface. The app only covers five Latin verbs but allows the user to create a custom quiz. Figure 6 below shows *Vice Verba's* user interface. The user simply touches the tenses, moods, and voices he or she wishes to practice and *Vice Verba* designs a custom quiz covering the selected material.

I realized that this type of interface would work equally as well with nouns as verbs. By avoid drop-down menus or other awkward UI designs, my app would be easy to use. Below in Figure 7 you can

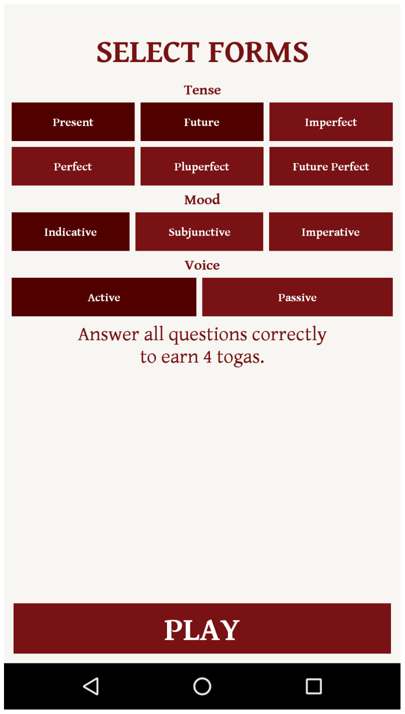

Figure 6: *Vice Verba's* Custom Quiz Selection

see how I used *Vice Verba's* interface to model my own for Latin nouns.

As mentioned before, many applications didn't allow easy input of vowels with macrons. So when designing my app I decided I wanted to make this as painless as possible. One solution I saw in another app was to have the user input a capital letter whenever a macron was required. While fairly ingenious, I felt there needed to be a better solution.

Many people find that they can tell if they have spelled something incorrectly because it just does not "look right." This is because humans are very visual creatures and we can recognize words and phrases by their "shape." Therefore I felt it was important to allow the user to type in the Latin noun with the macrons as he or she could reinforce that visual memory of where they belong in each word. To that end I designed an interface for each quiz question that had a permanent row of buttons for each vowel with macron. Figure 8 shows my mock-up.

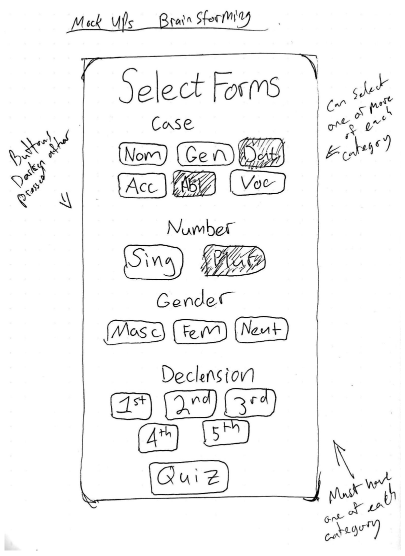

Figure 7: Custom Practice mock-up

Implementing Spaced Repetition

When researching how different tools have implemented SR I kept finding references to SM-2. Without question, SM-2 is the single most popular method of implementing SR in software. While this is the oldest version of SuperMemo's algorithm, it is very easy to understand and code into software.

SM-2 uses only one metric for determining how well a user knows the material: *quality rating* (q). Later versions of the algorithm have evolved to incorporate many variables with matrix multiplication. For my first foray into SR I wanted to make sure I used an algorithm that I could both fully understand and had a proven track record of success.

I chose to implement a custom version of SM-2 in my app. Minor changes were required to meet one of my criteria: *interval* calculation without user input. I didn't want to have to ask the user to select how well he or she knows the answer. The reasons were twofold. First, most people have a very bad understanding of their knowledge level. Secondly, asking the user for a *quality rating* breaks up the practice session and could be seen as an annoyance.

What allowed me to do this was the design calling of the app to quiz the user on concepts and not individual nouns. Anki and Cerego both track individual pieces of knowledge and a rating for each

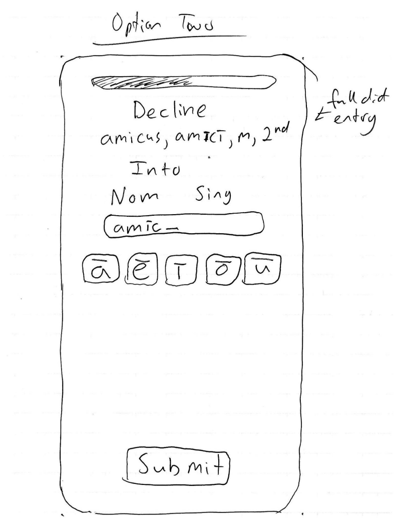

Figure 8: Quiz Question mock-up

piece is required for their algorithms to work. For my app, there are multiple noun forms for each concept.

For example, if a user chooses to practice Nom/Sing 1st Declension noun endings, the app will create questions that use a wide range of nouns with those endings to build the quiz. My app's algorithm will start each quiz with a *quality rating* of 5 (the highest possible). Each time a user inputs the incorrect answer the app deducts 0.5 from q to a minimum value of 0. My algorithm would then run the SM-2 equations to calculate the appropriate *interval*.

In review sessions that encompass multiple concepts, each question is connected to a unique q value shared across all questions with the same concept. In this way answering incorrectly a question about Nom/Sing 1st Declension endings wouldn't affect the *interval* calculations of the Nom/Plur 1st Declension endings concept that shows up in the same quiz.

After each review session the app would update the internal database to reflect the changes to each concept's *EF* and *interval*. The home screen would then display visually with color cues the mastery level for each concept. As the days go by, the app would update the colors for each concept based upon how close the knowledge is to the next scheduled review session.

Resulting Product

Using the above research and design parameters I was able to successfully create an Android app that leverages SR. Furthermore all the interface layout decisions were tailored specifically for Latin learners. The app is easy to pick up and immediately start using.

As you can see, the final app stayed fairly faithful to original mock-ups. The home screen in Figure 9 shows how I used the individual button icons as the visual clues for the user. As each concept's strength decays, the laurels both change from green to red, but also the amount of wreath visible shrinks. The user is also able to scroll up and down to view the other declensions or choose "Custom Practice."

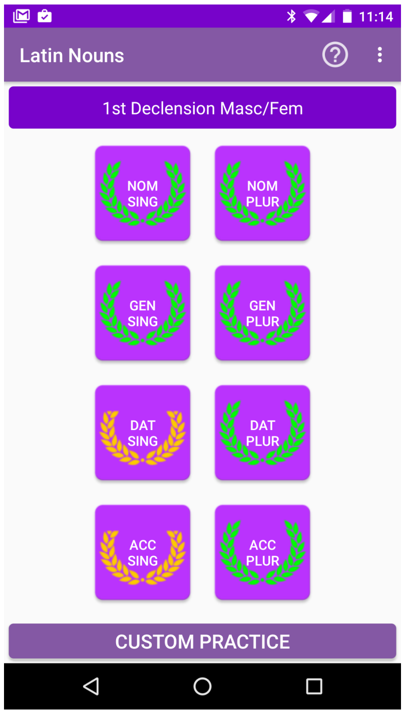

Figure 9: Final home screen

Figure 10 shows the app's final custom practice screen. As with *Vice Verba*, the user is able to use the touch screen to highlight which concepts to use in the review session. The user must select one from each category or the quiz will not start. Also, since some combinations of case/number/gender/declension do not actually exist in common Latin, the user is again prevented from starting the quiz.

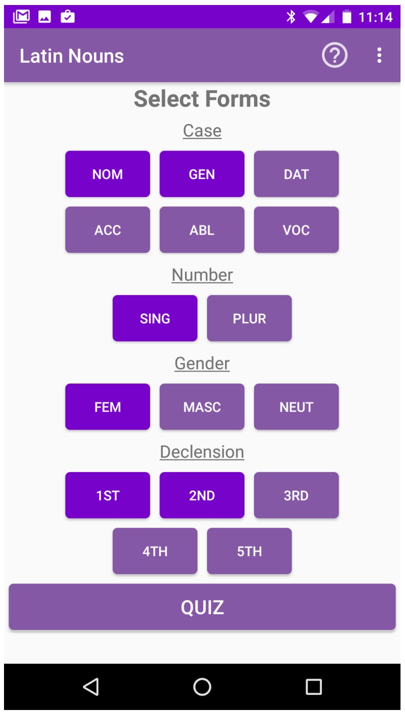

Figure 10: Final Custom Practice screen

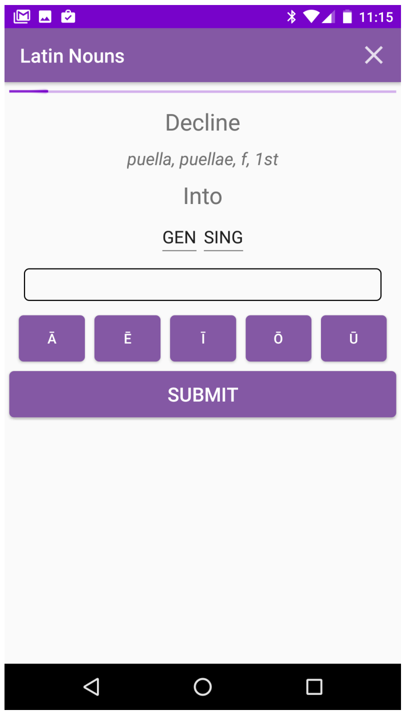

Figure 11: Final Question screen

Finally, Figure 11 demonstrates what a typical question looks like in the app. Note the special macron buttons that are always available. On most devices the onscreen keyboard does not cover the buttons so Latin input is easy.

My app is currently in alpha testing on the Google Play Store under the name *Latin Nouns*. It was made available to both my graduate school peers and to a class of Latin I students at Virginia Tech. Unfortunately not many people have installed the application and none have provided any feedback.

In order to determine if the app is useful and if my modifications to SM-2 are valid it requires testing. I cannot very well test it myself, because I already know the material and therefore cannot test how well it predicts me forgetting the material, which is not likely after all these years.

The Future

It is very possible my app has a bright future if I can find testers. One solution to this problem is to provide it to more students by contacting more universities and potentially high schools. Another thing that could be suppressing the download numbers is the alpha status. On the Google Play Store, apps in alpha testing can only be found by following a custom url and do not show up in any searches. Fully publishing the app on the store would make it available to more people instantly, but the general public is less likely to provide the type of feedback needed to adjust my algorithm.

Assuming for now that Latin Nouns is an effective tool for practicing Latin, it should be fairly easy to expand the scope of the app. Adding verbs or adjectives to the app would only require updating the databases with new concepts and adding a menu. The same algorithm can be applied to any type of knowledge. Likewise the Latin specific interfaces can be modified to incorporate the needs of other parts of speech.

I would also like to add a way to practice vocabulary using the same algorithm. This would involve adding definitions to the application and also allow users to build vocabulary lists either in-app or by loading and external file. This could be very helpful to instructors because they could provide their students with tailored vocabulary lists for the individual curricula.

The ultimate goal is to create a one stop Latin practice tool. It will continue to remain very general without targeting any particular textbook or author. But, it should include options to help students or instructors to customize the experience to meet specific needs. Over the coming year I hope to move closer to this goal while expanding the number of testers.

Bibliography

A Dictionary of Psychology. 2008. Jost's Law.

Bahrick, Harry P., Lorraine E. Bahrick, Audrey S. Bahrick, and Phyllis E. Bahrick. 1993. "Maintenance Of Foreign Language Vocabulary And The Spacing Effect." *Psychological Science* 4 (5): 316–321.

Duolingo: Language Courses for English Speakers. Accessed April 29, 2017. Language Courses for English Speakers. https://www.duolingo.com/courses.

Godwin-Jones, Robert. 2010. "From Memory Palaces To Spacing Algorithms: Approaches To Second-Language Vocabulary Learning." *Language Learning & Technology* 14 (2): 4?11 (Jun).

Hart, Robert S. 1995. "The Illinois Plato Foreign Languages Project." *CALICO Journal* 12, no. 4.

Kukulska-Hulme, Agnes. 2009. "Will mobile learning change language learning?" *ReCALL* 21 (02): 157.

Settles, Burr, and Brendan Meeder. 2016. Pages 1848–1858 in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Volume 1 of *Long Papers*. Association for Computational Linguistics.

Stockwell, Gleen, and Philip Hubbard. 2013. Some Emerging Principles for Mobile-Assisted Language Learning.

Wozniak, Piotr P. May 10, 1998. Accessed April 29, 2017. "Application of a computer to improve the results obtained in working with the SuperMemo method." *SuperMemo 2: Algorithm*, May. https://www.supermemo.com/english/ol/sm2.htm.