Bootstrapping an Ubiquitous Monitoring Ecosystem for Accelerating Vocabulary Acquisition

Mircea F. Lungu
SEARCH @ Johan Bernoulli Institute
University of Groningen
Netherlands

ABSTRACT

Learning the vocabulary of a new language is a very slow and time consuming process which can take many years of dedicated study. Free reading is known to be important for improving vocabulary and so are optimally timed repetitions of learned concepts. However, these two have not been put together until now.

This paper presents the architecture of a monitoring ecosystem of applications which tracks the reading activities of a learner and builds a model of their evolving knowledge. Based on this model it can steer their future reading and studying sessions in such a way as to accelerate the speed with which they acquire new vocabulary.

The paper describes several requirements for such an ecosystem, together with a prototpye implementation, and component applications. Finally a series of open questions that highlight opportunities for future research are discussed.

Keywords

software ecosystems; HCI; applied linguistics;

1. VOCABULARY ACQUISITION

At any given moment millions of people are learning the vocabulary of a new language. The first steps in the acquisition of the new language are usually full of enthusiasm, but often the learner gives up once he realizes the magnitude of the task.

Indeed, once a learner has acquired the basic vocabulary of a foreign language, they are still many thousands of words away from actually mastering the new language. To improve their vocabulary they must constantly expose themselves to contexts in which the learned language is used at a level that is not too difficult but not too easy either – that is, *they must study in the zone of proximal development*. Studying language textbooks is the traditional approach but, very often, textbooks being written for everybody are not particularly interesting for anybody.

ECSAW '16, November 28-December 02, 2016, Copenhagen, Denmark

ISBN 978-1-4503-4781-5/16/11...$15.00

DOI: http://dx.doi.org/10.1145/2993412.3003389

Amazon has made a first step towards allowing the learner to read engaging texts by integrating translations and basic vocabulary exercises with their proprietary eBook reader device – the Kindle. Besides the limitations of being locked to a given dvice, this solution suffers also from the fact that the words being learned are locked in by Amazon. We will see later that this limits the possibilities of accelerating the speed with which vocabulary acquisition happens. Another solution that the readers have is using Google Translate on the web. However, just as with Amazon, the translations that the user makes online are still not available outside the Google ecosystem. Thus the possibility of other applications benefiting from the knowledge of what the user is learning is also absent.

On the side of vocabulary rehearsal applications a plethora of solutions exist, but most of them are *data silos* – they do not interact with data from reader applications.

2. AN OPEN MONITORING ECOSYSTEM

To address the disadvantages of the locked down nature of the existing infrastructures, we propose as a solution an ecosystem consisting of a federation of applications that support a learner in accelerating vocabulary acquisition. Such an ecosystem should be built on the following principles:

Free Reading. The learners are able to read the materials they find interesting with the applications they prefer to use. When a learner encounters a word he does not understand, a high-quality, contextual translation will be offered.

Ubiquitous Monitoring. All the encounters of a learner with foreign materials are monitored in order to build an evolving model of the current state of the knowledge of the learner. The applications that are part of the ecosystem report a users interactions with the texts to a central repository.

Accelerated Learning. Based on the learner model intelligent agents recommend further reading materials that are both likely to be interesting to the reader and at the same time likely to maximize the retention of the most important studied words. The original context of the words can be used to accelerate retention.

Openness. The ecosystem is open to any reader or trainer application, be it open source or not, as long as it contributes back to the ecosystem. The data about a given leaner belongs to the learner themselves: they decide which applications have access their data.

This paper presents an ecosystem which has as a goal supporting a learner in vocabulary acquisition. However, an **ubiquitous monitoring ecosystem** can have other goals as long as it represents a *federation of applications that monitor a certain aspect of a user's interaction with information to build an evolving model of the user knowledge.*

It might appear intuitive for the reader that when the goal is accelerating the velocity of knowledge acquisition, such an ecosystem will present a network effect: the more contexts in which the reader is supported by applications from the ecosystem, the better the learner knowledge model that can be built, and the better the provided user experience.

3. A PROPOSED ARCHITECTURE

In this section we propose a high-level architecture of a ubiquitous monitoring ecosystem dedicated to accelerating vocabulary acquisition. Figure 1 shows some of the main actors and components in our ecosystem.

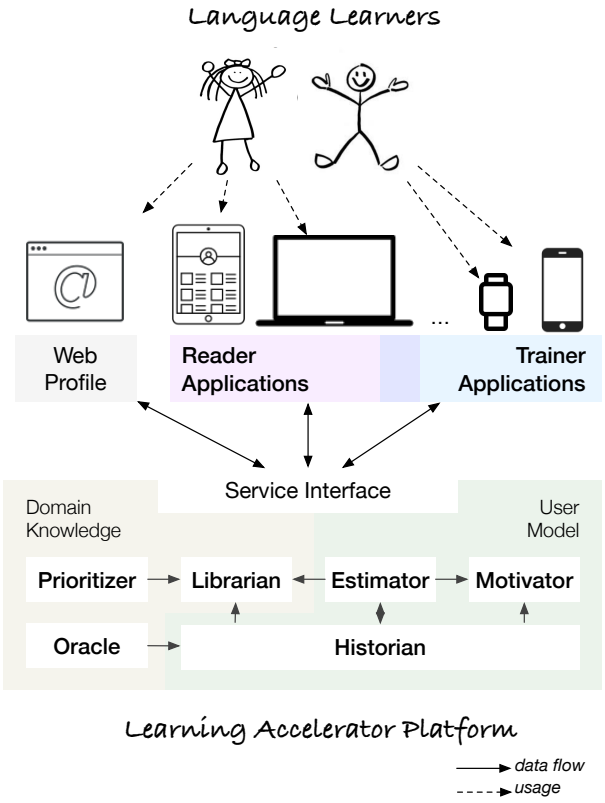

Figure 1: A very high-level view of the architecture of an exemplar ubiquitous monitoring ecosystem

Only one category of actors is explicitly depicted – the *language learners*. The *platform developers* and the *application developers* are missing from the picture. The latter are the makers of two types of applications:

Reader Applications facilitate the lecturing of materials in foreign languages. These applications have ideally two properties in common: 1) the usability of obtaining translations for unknown words is high (this is a quality attribute); 2) they report back to the platform information relevant for infering the current user knowledge[1] (this is a functional requirement).

Trainer Applications provide exercises to accelerate the retention of individual words. Trainer applications have in common two functional properties: 1) they should request from the platform a list of words to be studied and 2) they must provide back to the platform information about how well the learner behaved with respect to a given word[2].

Note that the Reader and the Trainer applications need not necessarily be disjoint applications; instead one application could provide both functionalities.

The *learning accelerator platform* sits at the core of the ecosystem since it stores and orchestrates the exchange of information between the various actors. It consists of six main components roughly divided into two categories: those related to domain knowledge and those related to user modeling. The components are:

1. **The Historian** is a data warehouse that sits at the core of a monitoring ecosystem. It records all the interactions of a user with knowledge based on the reports of Reader and Trainer applications. It stores the data in either a relational or a noSql database, depending on the type of data.
2. **The Oracle** is a service that *has all knowledge about the units of knowledge in the domain.* In our case it has knowledge of translations between many pairs of languages. It notifies the historian of every request it receives. It uses adaptive strategies to choose between different backends that have different properties in terms of cost and quality.
3. **The Prioritizer** is a data mining focused component which aims at *ranking the information in the domain* based on a global view of its relative importance. It can be built based on statistical analysis of the learning patterns of the various leaners or based on a generic study of corpora in the target language.
4. **The Estimator** is a machine learning agent that *estimates the current knowledge of the learner* based on user-specific information received from the Historian and language-specific information received from the Prioritizer. It decides what are the most likely items that must be studied by the learner.
5. **The Librarian** *provides reading recommendations wich are interesting to the learners while at the same being at the appropriate difficulty level.* It is a web crawler that uses natural language processing techniques but personalizes its results based on information from the Estimator.
6. **The Motivator** is an agent that *uses gamification techniques to provide feedback that would keep the learner motivated.* It uses information 1) from the Historian to report on a learners *engagement* and 2) from the Estimator to provide feedback on the actual vocabulary acquisition *progress.*

[1]e.g., the looked-up words, their context, reading speed, etc.
[2]e.g., the correctness of an answer, the time to answer, etc.

4. BOOTSTRAPPING THE ECOSYSTEM

In order to obtain a *minimal viable ecosystem* several of its key components must be in place. This section presents in a quasi chronological order several milestones in the evolution of the ecosystem over the recent years. Where appropriate we take the chance to step back and highlight general open questions that might be faced by future builders of monitoring ecosystems.

4.1 A Minimal Viable Ecosystem

The first milestone was releasing four main components of the ecosystem:

1. A *reader application* implemented as a Chrome extension.[3] It allows a learner to read any text and obtain in-place translations on any website as long as it is visited within the Chrome browser.

2. An initial version of the learning accelerator platform, containing the Historian, a Translator that relies on external services, and a very basic version of the Estimator, exposed via a REST API. [1].

3. A basic *Web Profile* application[4] that provides account management, and a simple interface to present the reading history of a learner.

4. A very simple *trainer application* which asks the learner to recognize a given word within its context. The words and contexts are selected from the past readings of the learner.

The web application and the REST API were deployed together on the same server as Python-based applications and the separation between them was not well enforced. This came to hunt us later when we discovered that as we were extending the REST API for other applications, we were duplicating functionality already existent in the web profile application. Surely a little technical debt can speed up the initial bootstrapping phase, but we think that it would have been better to treat internal applications as future third-party applications already from the begining.

The Chrome extension as a reader application posed two main limitations: 1) not everybody is reading his foreign language texts on their computer and, 2) some users are circumspect when it comes to installing third-party browser extensions. In truth, this circumspection is well founded since a browser extension has access to all the information on the pages the user visits.

In our case, we made our extension be active only on those pages where the user explicitly activates it. However, the only way a user can be completely confident that no information is leaking is reading the code. And although the code is open-source, this is not a realistic request from a language learner. A first question about privacy in a monitoring ecosystem emerges:

Can a monitoring ecosystem ensure and prove to its users that individual applications do not have access to more data than they are allowed to?

---

[3]Available at: https://zeeguu.unibe.ch/chrome

[4]The web application is available at https://zeeguu.unibe.ch

4.2 Adding a Reader for Android

The Zeeguu Reader for Android was implemented as a bachelor project by Schwab [2]. The application is written in Java and functions as an RSS feed reader. The architecture of the application makes a clear distinction between:

1. The *Zeeguu Android Library*[5] which eases the communication with the REST API and is released as a separate open-source component, and

2. The actual reader application which focuses on the usability of offering textual translations in context

The separation has proven to be a good choice since a second Android application – a Dictionary implemented by Giehl [3] could readily benefit from the API component.

During usability studies, however, we learned that a subset of the test users found it inconvenient to have both the Reader and the Dictionary applications installed. They would have preferred a single application. This raises another open question:

When is it better to fuse multiple applications into a single one and how to balance the convenience of a single application with the scalability of dedicated applications?

One of the features of the Android Reader application was ranking news items based on their difficulty with respect to the current estimated knowledge of the learner. Since such an algorithm would very likely be required for other applications in the ecosystem, we decided to move this functionality in a separate component on the server (the Librarian) and expose it using an API.

The Android Reader relies on Feedly, a third party API, to track news feeds. This turned out to be very cumbersome for learners since they required to create a new account to a different service, before using the application. This problem was solved in the case of the iOS reader.

4.3 Adding a Reader for iOS

Oosterhof [4] implemented a reader for iOS. The application written natively in Swift eliminates the explicit dependence on third party services like Feedly. Instead the burden of extracting feed information from pages and monitoring a users preferred feeds has been moved into a separate agent inside of the platform. We discussed earlier how the Librarian was also migrated to the platform from an app. This illustrates our assumption that having all the information about a learner stored in a single central place simplifies data processing and user modeling. However, a more distributed design cold also be investigated. Which brings us to the question:

Is it practical to have the learner modeling information distributed across applications or is the centralization of the learner model the dominant strategy in a monitoring ecosystem?

The architecture of the iOS application made again a clear separation between a component which was to interact with the API[6] and the actual GUI of the application. Since for

[5]https://github.com/linusschwab/zeeguu-android-library

[6]https://github.com/mircealungu/zeeguu-ios-library

every new language the developer must handwrite a new API interface to communicate with the core services, we realized that we missed the opportunity of automatically generating the APIs for different languages. However, as far as we are aware, no such generator exists for the Python technology stack used.

4.4 Adding a Smartwatch Trainer

According to current estimates, the wearable market[7] will pass 111 million shipped devices in 2016, up from 80 million shipped in 2015.

The ease with which a user can consult his smartwatch makes it an interesting platform for a learning strategy called *micro-learning* known for quickly closing skill and knowledge gaps [5]. Having a trainer on the smartwatch would make it easy for the learner to take advantage of and study during dead moments of the day (e.g. waiting for the barista to prepare a cappuccino).

A trainer, dubbed *Time to Learn*, was implemented for the Gear S2 smartwatch by Haan and Nienhuis[6]. The Gear S2 device runs on the Tizen mobile operating system and applications for it are written using HTML5 and Javascript.

To support micro-learning, the information on the smartwatch should be readily available when a learner looks at the watch. Thus, Time to Learn is implemented as a watch face which is divided in two: the top half presents the usual watch information, while the bottom part represents a word recognition challenge for the learner. The word to be displayed is selected based on an Estimator recommendation.

After every challenge, the learner provides feedback on whether he knew a given word or not. This feedback is sent back to the Historian so it can be used in the future by the Estimator.

On the platform, the Estimator has to take into account the existence of the smartwatch events explicitly and treat them in a different way than the events from the web based trainer. Theoretically, the Estimator could be implemented with machine learning technologies, in such a way as to be agnostic about the existence of the indiviudal trainers. However, this direction needs to be explored further. This leads us to raising a question about the *openess* principle (from Section 2):

Is it possible to let new applications join a monitoring ecosystem without having a central component that is aware of the existence of the individual applications?

---

5. REFLECTIONS

Based on early user studies and the feedback we received regarding the individual applications we can say that such an ecosystem shows promise. But work still has to be done to unequivocally show the benefits it brings to the learners.

Until now, all the applications that were added, have been in one way or another supported by the creator of the ecosystem. Although all the actors could benefit from joining such an ecosystem, the challenge of incentivizing external application developers actually join remains a future challenge. We will work on defining policies that will benefit all the players [7].

[7]Which includes fitness trackers and smartwatches, so the number of smartwatches is likely smaller

Also, this is not the first ecosystem that aims to track the interactions of a user with data and build a better user model based on this; many commercial companies do it too. However, the goals of these companies are usually driven by economic reasons, while the goal of our ecosystem is mainly supporting future research and education. Indeed, one type of stakeholder that we did not have space to talk about in this paper are researchers who could have access to real world, longitudinal data.

It would seem thus, that since economical reasons do not drive the evolution of the ecosystem, we should be confronting with different problems than industrial enterprises. However, the economical sustainability of such a platform when hosted in academia is not necessarily straightforward. The hosting costs of the core components are at the moment not significant but they could become large in the eventualily of a massive growth.

6. CONCLUSION

The paper presents a proposed architecture and reports several lessons learned while bootstrapping an ecosystem designed for accelerating vocabulary acquisition through free reading and optimally timed repetitions. The paper also lists several open research questions which are probably relevant for other types of monitoring ecosystems.

Acknowledgements The author would like to thank Anca Lungu, Sara Mahdavi-Hezavehi, Jens Knodel, Paris Avgeriou and the anonymous reviewers for feedback on earlier versions of this paper.

7. REFERENCES

[1] M. Lungu, K. Sethi, S. Marti, and L. Schwab, "The Zeeguu API - Modeling Learner Progress to Accelerate Vocabulary Acquisition," Jul. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.58569

[2] L. Schwab, "Using RSS feeds to support second language acquisition," University of Bern, Bachelor's thesis, Jun. 2016. [Online]. Available: http://scg.unibe.ch/archive/projects/Schw16a.pdf

[3] P. Giehl, "Zeeguu translate application — extending the Zeeguu platform to the Android device," University of Bern, Bachelor's thesis, Aug. 2015. [Online]. Available: http://scg.unibe.ch/archive/projects/Gieh15a.pdf

[4] J. Oosterhof, "Making reading in a second language more enjoyable," Aug. 2016, bachelor's Thesis, University of Groningen.

[5] D. Dearman and K. Truong, "Evaluating the implicit acquisition of second language vocabulary using a live wallpaper," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1391–1400.

[6] R. Nienhuis and N. Haan, "Time to learn – A new way of learning with the use of a smartwatch," Aug. 2016, bachelor's Thesis, University of Groningen.

[7] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *Presented at the 31st International Conference on Software Engineering - Companion Volume, 2009. ICSE-Companion 2009, IEEE*, 2009, pp. 187–190.