The FieldLinguists' App

1 Project Abstract

The FieldLinguists' App is an OpenSource database that allows language researchers to securely enter, store, organize, annotate, and share linguistic data. The application is accessible everywhere; it runs on three different systems (Mac, Linux, and Windows) and is suitable for both online and offline use. Furthermore, the application will be created with collaborative goals in mind; data will be syncable and sharable with other researchers. Researchers can form teams that contributes to a single corpus, where team members can modify and discuss the data from within the application. The system will also have a simple and friendly user interface, allowing participants to drag and drop files (audio, video, text), or record it directly into the database when using the Android app. In addition, the application will have import and export capabilities for multiple file types. Most importantly, the application is designed intuitively and theory free, so it is not necessary to be a field linguist or programmer to figure out how it works.

2 Statement of Need

The statement of need should describe the problem that the project will attempt to address. Also, describe the population that will be served.

3 App Description

The FieldLinguists' App will enable those interested in language research, preservation, and documentation to securely enter, store, organize, annotate, and share linguistic data. Moreover, the application will be easily customizable to fit specific needs. To accomplish these tasks, the database will be equipped with a variety of features. The following functional requirements are based on a few important considerations where most existing field linguistics/corpus linguistics databases applications fall short.

- **Modern and Intuitive**
  - **Simple** The system will be designed to replace Word Documents or LaTeX documents which is a very common way fieldlinguists store data because it requires no training, doesn't require a complicated set-up for data categories, and takes no time to add new categories.
  - **Attractive** The system will have a modern design like many of the popular websites such as google and be customizable so that the user include a picture of where they are doing research as background.
- **Powerful**
  - **Smart** The application will guess what users do most often, and automate the process for them. Most importantly, the system will have automated predictable glossing information .
  - **Searchable** The application will be designed for search as this is one of the most fundamental tasks a language researcher must be able to do. The search will go far beyond traditional string matches and database indexes; it will be able to display data in context.
- **Data-Centric**
  - **Atheoretical** The application will not include categories or linguistic frameworks or theoretical constructs that must be tied to the data. The application allows an analysis to develop organically as data collection proceeds as opposed to imposing a particular construct upon entry. Researchers can set and change their own categories to the data whenever they choose to.
  - **Collaborative** The system will have users and teams, and permissions for corpora. Permissions will ensure that data can be safely shared and edited by multiple users. Moreover, the corpus will be versioned so that users can track changes and revert mistakes.
  - **Sharable** The application will allow researchers to share their data to anyone interested in their work.
- **Accessable**
  - **Cross-Platform** The application will run on Mac, Linux, and Windows computers. The application will be installable as a Chrome extension and available on any device that runs a browser.
  - **Portable** Touch tablets are one of the easiest tools to carry and use in field; they have a long battery life; they can play videos or show images for the consultant

to elicit complicated contexts; and they permit recording audio and video and direct publishing to YouTube and/or other services. Furthermore, Android tablets are particularly easy to program and integrate the microphone directly into the database.

  - **Work offine** Running a webapp offline will have considerable consequences for how data is stored, how data is retrieved, and how much data can be used while offline. Most browsers have limits on the amount of data a webapp can store offline. By delivering a version of the app in a Chrome Extension, which has permission to have unlimited storage, researchers will be able to have a significant portion of their data at their fingertips, regardless of the location.

- **Open**

  - **OpenData**. Corpora often contain sensitive information, informant stories and other information which must be kept confidential. Having confidential data in plain text in a corpus forces the entire corpus to be kept confidential. Instead, the system will encrypt confidential data and store the data in the corpus encrypted. To access the plain text the user will have to log in and use a password to decrypt the data. This design has important ramifications for exporting data, and for editing the data outside the application.
  - **OpenSource**. Being OpenSource allows departments to install and customize the database application to tailor specific needs without worry that the company behind the software will disappear or stop maintaining the software. In addition, OpenSourcing the software on GitHub will allow linguists with scripting or programming experience to contribute back to the software to make it more customized to their needs, language typologies, or linguistics research areas.
  - **Unicode**. Encoding problems and losing data should be behind us in the days of unicode. However, many existing fieldlinguistics databases were built in programming languages that did not support unicode, so the unicode support is dangerously fragile.

4 Goals & Objectives

Describe the project objectives in measurable terms that address the academic and technological needs of language teachers, linguists, etc. (it's for linguists but other people can benefit from it as well i.e. help create dictionaries for endangered languages, which will benefit the communities of these languages; L2 acq. teachers).

5 Staff & Organizational Information

iLanguage Lab is a Montreal based company that develops tools in the form of experimentation and data collection apps for Android and Chrome in collaboration with researchers at UdeM, UQAM, McGill and Concordia. Previous research applications includes the Bilingual Aphasia Test, AuBlog, OPrime and SpyOrNot. Furthermore, iLanguage Lab has a background in assisting researchers obtain results and publications. The AuBlog application was employed to investigate evolving information structure and audienceless vs. audience oriented prosodies and culminated in a poster presented at Experimental and Theoretical Advances in Prosody Conference. The Bilingual Aphasia Test led to a presentation at the Academy of Aphasia 49th Annual Meeting on Aphasia Assessment on Android: recording voice, eye-gaze and touch for the BAT and a publication in the Academy of Aphasia.

5.1 Gina Cook M.A.

Gina Cook received her Masters in Field Linguistics & DESS in Computer Science and has worked as a computational linguist for companies such as Nuance and Idelia. She founded iLanguage Lab with a vision to develop computational tools to help researchers as opposed to consumers. She is an active contributor to OpenSource projects on GitHub focusing on integrating existing OpenSource libraries for Speech Recognition, Natural Language Processing, Eye Gaze analysis and Acoustic analysis into Android tablet applications.

5.1.1 Publications

- "Aphasia Assessment on Android: recording voice, eye-gaze and touch for the BAT." (with A. Marquis & A. Achim). Poster at Academy of Aphasia 49th Annual Meeting, Montéal, Québec. October 2011.
- "Eliciting evolving information structure and audienceless vs. audience oriented prosodies: experimentation on Android tablets." (with S. Kattoju). Poster at ETAP2 Ð Experimental and Theoretical Advances in Prosody, Montréal, Québec. September 2011.
- "PDFtoAudioBook Android app" (Java, XML).Canadian University Software Engineering Conference (CUSEC) DemoCamp, Montréal, Québec. January 2011.
- "Word features and word concatenation."Sixth Interdisciplinary Graduate Student Research Symposium, McGill University, Montréal, Québec. March 2009.
- "The Structure of Long Distance Agreement in Hindi/Urdu." Invited Lecture in Advanced Syntax, Concordia University, Montréal, Québec. November 2007.

- "The Phonological/Phonetic status of Productive Palatalization in Romanian." (with L. Spinu). Presented at the Seoul International Conference on Linguistics, Seoul National University, Seoul, South Korea. July 2006.

5.2 M.E. Cathcart Ph.D.

M.E. Cathcart completed her PhD at the University of Delaware with a dissertation grant funded by the National Science Foundation (NSF) for her field work in Cusco, Peru on Quechua. In addition, she also has a background of coursework in computational linguistics, at the University of Delaware and at the Linguistic Society of AmericaÕs Summer Institute.

5.2.1 Publications

- "Affected Arguments Cross-linguistically." S. Bosse, B. Bruening, M.E. Cathcart, A. E. Peng, M. Yamada. In: Tadic, M. Dimitrova-Vulchanova, M., Koeva, S. (eds.): FASSBL 6 The Sixth International Conference on Formal Approaches to South Slavic and Balkan Languages. 2008 (Proceedings) pp. 41-47.

- "A New Grammatical Category: Impulsatives." Penn Linguistics Colloquium, Philadelphia March 2010 ?Eliciting data for dissertation on Impulsatives: functional morpheme in context

- "The Syntax and Semantics of Desideratives in Albanian." Georgetown Linguistics Society, Washington, D.C. February 2010

- "Bi-Eventivity & Affecting Arguments." S. Bosse, B. Bruening, M.E. Cathcart, H.-j. Cheng, A. E. Peng, M. Yamada. Formal Approaches to South Slavic and Balkan Languages, Dubrovnik (Croatia), 25-28 September 2008.

- "Bi-Eventive Affect." S. Bosse, B. Bruening, M.E. Cathcart, H.-j. Cheng, A. E. Peng, M. Yamada.TEAL, Potsdam (Germany), 10-11 September 2008.

5.3 Theresa Deering M.A.

Theresa Deering has a Bachelor's in Computer Science from Malaspina and a Master's in Computer Science from McGill University. Her thesis focused on the Least-Used Direction pivot rule for the Simplex Method of solving linear programs.

- "The Least-Used Direction Pivot Rule on Acyclic Unique Sink Orientations." Master's Thesis. McGill University, Montréal, Québec. July 2010.

- "Worst-case Behaviour of History Based Pivot Rules on Acyclic Unique Sink Orientations of Hypercubes." Y. Aoshima, D. Avis, T. Deering, Y. Matsumoto, S. Moriyama. Submitted to AAAC. October 2011.

6 Budget

The FieldLinguists' App is composed of eight modules and thus the cost is divided into eight major components. In addition, a separate price is given for software architecture and for 1 year of user support and project growth; needed to make a longterm viable and useful tool that fieldlinguists can adopt for their labs or for their field methods courses. The collaboration module is used to permit collaboration with teams and users. The Corpus Module is used to sync, share, edit, tag, categorize and open data. The Lexicon Module is used to house, and read lexicon entries to be used for the glosser. The Dictionary Module is used to share the lexicon in the form of a WordNet/Wiktionary dictionary with the language community as required by some grants. The Glosser Module is used to automatically gloss datum, smarter than the standard lexicon. Finally, the Aligner Module is used to create TextGrids from the orthography and the audio files, used for prosody and phonetic analysis.

The cost is calculated by determining the time in hours and multiplying by $42-58, the average rate of a software developer in Montréal.

Module	Weeks	Price
Software Architecture	0.5	$1,555.20
Collaboration Module	2.5	$5,728.32
Corpus Module	4.2	$9,201.60
Web Spider Module	2.5	$2,177.28
Phonological Search Module	3.0	$2,177.28
Lexicon Module	2.0	$7,340.54
Dictionary Module	22.3	$18,781.63
Glosser Module	19.7	$17,770.75
Aligner Module	10.2	$ 9,787.39
User Support	21.1	$30,246.70
TVS and TPQ		$10,833.32
Total	88	$83,176.04

Table 1: Project Summary

6.1 Collaboration Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Collaboration API on central server	30	Software Engineering
Users Model	15	Javascript
Informant Model	15	Javascript
Team Model	15	Javascript
Bot Model	15	Javascript
User Activity Model	8	Javascript
Team Feed Widget	25	HTML5
User list item Widget	16	HTML5
Team Preferences Widget	8	HTML5
User Profile Widget	8	HTML5
User Tests	30	Javascript
Informant Tests	30	Javascript
Team Tests	30	Javascript
Android Deployment	15	Java
Chrome Extension Deployment	20	Javascript
Heroku Deployment	5	Integration

Table 2: Module Details

6.2 Corpus Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Corpus API on corpus server	20	Software Engineering
Corpus Model	8	Javascript
Session Model	8	Javascript
Datum Model	8	Javascript
Datum status model	8	Javascript
DataList Model	8	Javascript
Confidential datum encrypter	16	Javascript
Audio upload and play logic	8	Javascript
Corpus DB implementation on Android	20	Java
Corpus DB implementation on Chrome	20	Javascript
Corpus DB implementation on Node.js	20	Javascript
Corpus versioning Logic	25	Javascript
Corpus Preferences Widget	6	HTML5
Session Preferences Widget	6	HTML5
Datum Preferences Widget	20	HTML5
Datum Status Preferences Widget	16	HTML5
DataList Preferences Widget	6	HTML5
Corpus sync logic	10	Javascript
Corpus diff Widget (to show before sync)	10	HTML5
Insert Unicode Character Widget	10	HTML5
Corpus Details Widget	6	HTML5
Session Details Widget	6	HTML5
Datum Details Widget	20	Javascript
DataList Widget	30	Javascript
Global Search logic	30	Javascript
Power Search logic	80	Javascript
Corpus Tests	5	Javascript
Session Tests	10	Javascript
Datum Tests	10	Javascript
Datum Status Tests	10	Javascript
DataList Tests	20	Javascript
Heroku Deployment	5	Integration

Table 3: Module Details

6.3 Web Spider Module

Iteration	Hours	Technology
Corpus Visualization Widget	40	HTML5
Web Spider Training Logic	60	Java

Table 4: Module Details

6.4 Phonological Search Module

Iteration	Hours	Technology
Phonology Ontology for phonological search	60	Java
Lexicon Visualization Widget	40	Javascript
Lexicon Editing Widget	20	Javascript

Table 5: Module Details

6.5 Lexicon Module

Iteration	Hours	Technology
Software Architecture Design	20	Software Engineering
Lexicon API on Lexicon server	20	Software Engineering
Lexicon Model	6	Javascript
Morpheme Model	6	Javascript
Allomorph Model	6	Javascript
Gloss Model	6	Javascript
Orthography Model	16	Javascript
Lexicon DB implementation on Android	20	Java
Lexicon DB implementation on Chrome	20	Javascript
Lexicon DB implementation on Node.js	20	Javascript
Lexicon versioning Logic	10	Javascript
Lexicon Preferences Widget	6	HTML5
Morpheme Tests	6	Javascript
Allomorph Tests	6	Javascript
Gloss Tests	6	Javascript
Orthography Tests	8	Javascript
Lexicon Analysis Widget	10	HTML5
Lexicon sync logic	10	Javascript
Lexicon diff Widget (to show before sync)	10	HTML5
Lexicon Details Widget	6	HTML5
Lexicon Tests	12	Javascript
Heroku Deployment	5	Integration

Table 6: Module Details

6.6 Dictionary Module

Iteration	Hours	Technology
Software Architecture Design	40	Software Engineering
Dictionary API on Lexicon server	30	Software Engineering
Semantic Model	60	Javascript
Syntactic Model	60	Javascript
Citation Model	60	Javascript
Synonyms Model	60	Javascript
Dictionary DB implementation	80	Integration
Dictionary Training Logic	80	Java
Web Spider Training Logic	100	Java
Dictionary Preferences Widget	8	HTML5
Dictionary WordNet Analysis Widget	120	HTML6
Dialect Profile Widget	8	HTML5
Semantic Tests	30	Javascript
Syntactic Tests	30	Javascript
Citation Tests	30	Javascript
Synonyms Tests	30	Javascript
Spider Tests	30	Java
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 7: Module Details

6.7 Glosser Module

Iteration	Hours	Technology
Software Architecture Design	40	Software Engineering
Glosser API on Lexicon server	30	Software Engineering
Morpheme Model	15	Javascript
Allomorph Model	15	Javascript
Gloss Model	15	Javascript
Orthography Model	30	Javascript
Glosser DB implementation	80	Integration
Glosser Prediction Logic	80	Java
Glosser Machine Learning Logic	80	Java
Glosser Training Logic	80	Java
Web Spider Training Logic	80	Java
Glosser Preferences Widget	8	HTML5
Morphological Analysis Widget	40	HTML6
Dialect Profile Widget	8	HTML5
Morpheme Tests	30	Javascript
Allomorph Tests	30	Javascript
Gloss Tests	30	Javascript
Orthography Tests	30	Javascript
Spider Tests	30	Java
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 8: Module Details

6.8 Aligner Module

Iteration	Hours	Technology
Software Architecture Design	10	Software Engineering
Aligner API on Lexicon server	10	Software Engineering
Dictionary Model	15	Javascript
Aligner DB implementation	80	Integration
Aligner Machine Learning Integration	80	Java
Aligner Preferences Widget	8	HTML5
Audio Waveform Visualization logic	30	Javascript
Audio Spectrogram Visualization logic	?	Javascript
Transcription User Interface	80	HTML5
TextGrid export	20	Javascript
Dialect Profile Widget	8	HTML5
Orthography Tests	30	Javascript
Training Tests	30	Java
Heroku Deployment	5	Integration

Table 9: Module Details

6.9 User Support

Iteration	Hours	Technology
Sample data	30	Linguistics
Integrate software with sample data	30	Javascript
Screencasts on how to use the app(s)	24	Quicktime/YouTube
Screencasts on how to modify the code	40	Quicktime/YouTube
Server maintenance	20	Integration
Monitor server costs and develop pricing plan	100	Business
Answer user emails	250	Support
Read twitter feeds and facebook channels	100	Support
Help IT/developers install and set up the server on their department servers	50	Support
Upgrade javascript/android libraries	40	Javascript
Amazon EC2 server CPU+Memory+Bandwidth		Server
Release new versions	160	Javascript

Table 10: Module Details