

# The e-Linguistics Toolkit

Scott Farrar and Steven Moran  
University of Washington  
Department of Linguistics  
Box 354340, Seattle, WA 98195  
{farrar,stiv}@u.washington.edu

## Abstract

*In order to achieve the objectives of the e-Humanities framework, it is necessary for individual humanities fields to take charge and to create their own specific techniques that apply to their own unique varieties of data. The e-Linguistics Toolkit is presented to aid the move of linguistics into this new digital era. We show that achieving this requires data interoperability in terms of encoding, format, and content. Once data are made interoperable, data manipulation in the form of validation and merging is necessary. As one way to ensure immediate usefulness to the ordinary working linguist, the toolkit offers data transformation services among various working formats and lays the foundations for intelligent search.*

## 1. Introduction

The field of linguistics, from natural language processing to typology, rests on its foundations of being data-driven. Data are collected and analyzed, scrutinized, and re-analyzed. Until lately, this task had to be carried out manually, often making broad comparative work or data processing on any large scale impossible or contentious at best. With the rise of the Web as an important source and medium for storing linguistic data, the situation has decidedly changed. With the Web, there has been much interest in combining pre-Web computational approaches with Web-based technologies. We refer to the processing of linguistics data using computers (especially Web-based computing) as **e-Linguistics**.

While situated squarely under the rubric of e-Humanities and thus inheriting all the associated challenges, e-Linguistics also faces the additional difficulty of combining methods in natural language processing with the accumulated knowledge of linguistics. Traditional linguistics has influenced NLP. But ironically, linguistics as a whole has not benefited from the advances in NLP. In order to con-

solidate these techniques and, at the same time, to achieve broad-based linguistic data interoperability, we have developed the e-Linguistics Toolkit (ELTK) which is accessible at <http://e-Linguistics.org/>. The toolkit addresses these major issues:

- data interoperability in terms of encoding, format, and content
- data manipulation (validation and merging)
- data transformation to various working formats
- leveraging NLP techniques to serve the whole of linguistics

In this paper we introduce the e-Linguistics Toolkit by describing its major functionality and justifying our approach.

## 2. Problems and Possible Solutions

As in most of the humanities, a primary source for data used in scholarship has been published print materials. A long tradition of citation and *de facto* scholarship standards have existed and have contributed to our successes. But recently, since the Web has become a major source and forum for data, the situation has changed dramatically. Individual researchers are self-publishing an extraordinary amount of data on the Web. Print materials of the past 100 years and older are slowly being digitized (usually scanned) and posted to the Web, either as part of digital libraries or smaller-scale projects. By embracing the Web as the primary forum for data exchange and its emerging standards as part of our overall methodology, it is possible to scale the scientific techniques developed over decades of linguistics research to get at the knowledge of language. But there are problems.

First, there is the problem of format. By format we refer to character encoding and data structure format. Data are rarely fully compatible with other data. For every piece

of software, there is a unique output format that is likely incompatible with any other application. As a solution to the format problem, projects such as the Text Encoding Initiative (TEI) [15] and the E-MELD project<sup>1</sup> have responded by recommending certain structured languages (SGML, and now XML) to be used in conjunction with Unicode.

The second issue concerns content annotation. Due to the variability of linguistics terminology and to the many theoretical traditions, annotations are rarely fully compatible. Terms do not have the same meaning across traditions, and data structures are often dissimilar. The content problem is perhaps more challenging than that of the format problem, though it is also being addressed by other efforts, e.g., [5]. The content problem can be illustrated by the following example. Consider an information extraction system whose task is to discover and aggregate instances of linguistics data (e.g., lexical entries) found on the Web. Discovery is difficult enough, but assume that a number of data instances have been successfully identified and the task becomes that of aggregation. This involves combining entries that are the same, or similar, such that the resulting system contains no duplicate entries. Even if the structure of the entry is consistent across data instances, lexical entries may differ according to terminology, such as the part of speech for the headword. In general, it is very difficult to determine that **n.**, **Noun**, and **cn.** refer in fact to the same concept.

Third, even if the format and content issues are solved, data are copied and sometimes re-copied to suit various new purposes. This path is imperfect. For instance, consider the case of Wikipedia, the popular Web encyclopedia site. While members of the Wikipedia community, by and large, place a high priority on clarifying provenance and accuracy, errors are bound to creep in: often intentional but well-meaning alterations to data can be seen as errors. Even more serious are the ever growing Web search engines that mine data and re-purpose them for presentation or optimization. As an example of this is the ODIN project [11]. Due to the semi-supervised algorithms used to identify and process the data, e.g., language identification (the assigning of a language code to each bit of data), there are bound to be errors. This does not, in general, distract from the tool's usefulness, but might give reason for researchers to worry, especially if *their* data are misconstrued in any way.

Finally, any new approach or technology requires critical mass. If too few in a community use the technology, then it will usually fail. TEI recommendations (using SGML) never caught on with the ordinary working linguist, likely due to the unavailability of tools to produce it. The situation with recent best-practice XML recommendations has been only slightly better.

It has been evident, even with the successes of the TEI, E-MELD, and other programs, linguists simply are not

abandoning their print-based ways. There are many good reasons for this. We demonstrate with the e-Linguistics Toolkit that it is a first step in convincing the ordinary working linguist to adopt the stance taken by e-Humanities and to produce data that is maximally interoperable. We are confident in this, because critical mass can be achieved by providing useful and novel services to linguists, convincing them that using these tools is a path towards a better understanding of their field.

### 3. The e-Linguistics ToolKit

In this section we introduce the e-Linguistics ToolKit (ELTK) as one possible solution towards field-wide data interoperability and as an example of techniques in e-Humanities. The toolkit is accessible at <http://e-Linguistics.org/> and is implemented in Python. The ELTK currently consists of a set of Python modules usable by interested programmers to create e-Linguistics applications. In current form, the ELTK consists of three main modules with three distinct functions: (1) **readers** for the migration of data in various working formats, (2) **ontology** for the storage and merging of transformed data, and (3) **display** for transforming data into various presentation formats.

#### 3.1. Readers

The first module can be used to transform legacy data and data in various working formats into an interoperable store. The transformation is according to encoding, form, and content. On the path to interoperability, all data are eventually transformed into Unicode. There are utilities to convert special ASCII character conventions used in Praat<sup>2</sup> (e.g., Praat internal codes) and Elan<sup>3</sup> (e.g., XSAMPA) to Unicode IPA. Though not fully implemented, the ELTK will be able to detect the semantics of characters. This is a problem when processing data in arbitrary, often unknown encodings. For instance, the character *p* might be used by a Russian linguist in transcribing an */r/* phoneme.

In terms of form, the ELTK translates among various working formats such as Praat *TextGrid* files, Elan *eaf* files, and data in plain text files. At the moment Leipzig-style glossing in plain text format is recognized. Several fundamental data types are handled, including lexicons, interlinear glossed text, paradigms, feature structures, and bibliographic citations. For example, the ELTK uses an in-house built Web scaper to harvest linguistically-related bibliographic references from selected Web sites that allow Web crawling. Extracted citations are written to a Bibtext file, loaded into the BibtextReader, and are then transformed into an RDF store where each bibliographical entry

<sup>1</sup><http://e-meld.org/>

<sup>2</sup><http://www.fon.hum.uva.nl/praat/>

<sup>3</sup><http://www.lat-mpi.eu/tools/elan/>

is identified with a URI. We have chosen Bibtex because it is a plain text non-proprietary format for marking up bibliographic data. It is widely used and third party software makes it possible to include Bibtex bibliographies in popular word processors, such as Word. Also, an OWL specification for Bibtex already exists [8], which we make use of.

As for content, all linguistic units and labels are mapped to the GOLD [6] namespace in a supervised manner. For instance, *PST* and *Past* would both be mapped to *gold:PastTense*. Functionality is included to create mapping tables of annotation symbols and their corresponding GOLD concepts. As the content issue is perhaps one of the most problematic, how to map annotation symbols to concepts is still on-going research.

The ELTK uses the popular Natural Language Toolkit (NLTK)<sup>4</sup>; it is in fact modeled somewhat after the NLTK as its name suggests. The ELTK depends on the NLTK, for example, in order to parse the translation line in inter-linear glossed text. The ELTK also uses the ISO 639-3 / Ethnologue 15 language codes. We require that each data element processed by the ELTK be linked to a particular 3-letter code, identified by particular URIs.

### 3.2. Ontology

The functionality of the second module is to manage (store and merge) the transformed data. This module interfaces with both a database and various ontologies. For this purpose, the ELTK relies heavily on the RDFLib<sup>5</sup> Python library for manipulating RDF.

At the core of the ELTK is the idea of a Uniform Resource Identifier (URI) [3]. URIs are used to ensure that everything that a linguist refers to is uniquely identifiable. In keeping with this, the ELTK emphasizes URIs for everything: authors, bibliographical entries, data instances, as well as linguistic concepts and relations. Second, the ELTK uses the data model of the Resource Description Framework (RDF) [10]. These two technologies are at the heart of the Semantic Web, a much debated and often misunderstood enterprise. The major misunderstanding is that “the Web community will do it for me”. But as nearly a decade of Semantic Web buzz has taught us, no one is going to transform the current Web into semantically interoperable resources, that is, except for individual Web communities. We have embraced these technologies in part because they are being accepted by major players in software development such as IBM, Microsoft, and Hewlett-Packard [7]. But also, the simplicity of the data model appeals to our discipline, namely that many of our fundamental data structures

can be easily adapted to fit into the RDF subject-predicate-object model. But the primary reason why we believe the Semantic Web movement is ultimately achievable is that it emphasizes ontologies as the key to content interoperability. As such, we have designed the ELTK to be compatible with the General Ontology for Linguistic Description (GOLD) [6]. GOLD is just one possible ontology for the ELTK, and we hope that other ontologies will soon be available for the purposes of e-Linguistics.

### 3.3. Display

Finally, the third module is responsible for the creation of various output formats. These include various recommended best-practice XML formats such as [4]. But importantly, the ELTK is able to output working formats, e.g., in the form of *TextGrid* and *eaf* files, even when the input source has a completely different structure. The module can also output fundamental linguistic data types using various presentation formats including HTML, L<sup>A</sup>T<sub>E</sub>X, and PDF.

### 3.4. Design choices

The choice of a development language can ensure the success of such a project, but it can also be its demise. And since there is an increasing array of choices concerning a development language for the Semantic Web, scripting, and ontology programming, some comments regarding our choices are in order. We have chosen Python as the primary access language for the toolkit, but we utilize several Java libraries for the more intensive processing. To accomplish this, we use the JPy<sup>6</sup> package as integration code to give Python programs full access to Java class libraries.

There are two reasons why we chose Python. First, Python is one of the most accessible languages for beginning programmers (though a similar argument could be made for Ruby and possibly Perl). This is an important consideration, because we envision that our toolkit will serve the ordinary working linguist who, in general, is not trained for coding and often does not have the means to hire expert programmers. Second, Python is used in the related NLTK project (perhaps even for similar reasons) and we envision a high degree of complementarity with our two projects. At this time we have not chosen to incorporate our own toolkit with the NLTK because the goals of the latter are fundamentally different. Whereas the NLTK is designed for teaching and research in mainstream computational linguistics (which includes parsing, machine learning, and other core algorithms), our project is intended to leverage those techniques to the benefit of linguistics in general. As computational linguistics has seen huge successes over the last two decades, those successes have not, as a rule, benefited

<sup>4</sup><http://nltk.org/doc/>

<sup>5</sup><http://rdflib.net/>

<sup>6</sup><http://jpye.sourceforge.net/index.html>

the entire field of linguistics. The e-Linguistics Toolkit is intended to fill that gap.

As noted, Python affords a quicker development cycle for our purposes. For example with Python, it is easier to process text than with Java. It is also easier to write unit tests against our Python code. Granted, Python has its share of drawbacks. There are certainly performance issues to contend with, performance issues that are common to many interpreted languages. But for our purposes, performance is not as critical. More serious is that the development of key Python libraries lags behind that for similar Java libraries. This is likely due to Java's huge user base. Python is just now moving to a pure Unicode implementation, and key components such as RDFLib have not yet been refactored to accommodate this move. So, while using pure Python would be feasible, we also want to take advantage of cutting edge tools for Semantic Web development, something for which Java seems particularly well equipped.

It is clear that such a project could be carried out using pure Java, C++ or other commonly used, non-interpreted languages. After all, numerous Java libraries are available for Semantic Web development, such as Jena [7] and the OWLAPI [2]. The biggest reason why we have chosen an interpreted language such as Python is the possibility of **meta-programming**, or the ability to write code to manipulate code. But why is this useful? The biggest reason why meta-programming is required, or at least highly desirable, is to seamlessly access the domain model of OWL with Python. In fact, we do not simply access the data model, but import it into the Python programming environment such that OWL classes, properties and individuals are created and manipulated alongside Python classes, functions and instances. As inspired by [1], the OWL class hierarchy can be directly imported into the Python class hierarchy. The goal is to produce Python code that reflects the OWL domain model, something that seems quite natural for a Semantic Web effort and which may be characterized as a type of **ontology-driven software design** [9]. Ontology-driven software design and meta-programming itself are known to be quite difficult to achieve in statically typed languages such as Java [9].

In general OWL is conceptually similar to the object-oriented programming (OOP) paradigm, as used in Python. Both OWL and OOP allow for classes and subclasses, along with inheritance and limited multiple inheritance. Object composition and class instantiation are also similar. But OWL semantics is inconsistent with that of Python's in a number of key aspects. For example in most OOP languages, a class instance can only belong to a single class. That is, in Python, the expression `type(MyInstance)` can only yield a single class. This ensures the behavior of instances based on the associated methods and variables of the instantiated class. In OWL, however, a single individ-

ual (corresponding to an instance in OOP) can instantiate multiple classes in the same knowledge base. Thus, to provide a linguistics example, a particular language can be an individual of both `EndangeredLanguage` and of `Koiné` at the same time. In the ELTK we manage to integrate this facet of OWL semantics in a fairly seamless way.

Our implementation differs somewhat from the Seth library of [1] in that we leave most of the computationally intensive processing on the Java side. This includes RDF processing. While the ELTK does use RDFLib for some RDF processing, the parsing and writing of OWL files (regardless of the serialization) is handled via the OWLAPI. A key reason to push such processing to the Java side concerns the performance of Pellet [14], the reasoner that implements the tableaux algorithm, and KOAN2 [12] the framework useful for reasoning with large numbers of individuals.

## 4. Examples of Use

The Web has become the predominate resource for obtaining language data, whether it be secondary linguistic field data or structured corpora. Since the majority of the world's languages lack tagged corpora for NLP applications, many data mining and language identification algorithms have been developed to create under-resourced (or *low density*) language corpora [13]. These corpora are used to bootstrap algorithms for spelling and grammar checkers, automatic generation of word lists and dictionaries, and the creation of treebanks for further application development in NLP. Web-derived text corpora, however, also provide a rich resource for phonological investigation. Orthographic properties of some languages allow textual data to be used for phonological, and even morphophonological, research. For example, Zuraw uses a Web-derived corpus to investigate frequency affects of Tagalog's intervocalic tapping rule [16]. Moreover, Zuraw's study shows that a Web-based corpus was preferable to a traditional newspaper-derived corpus, because the informal writing of blogs and Web forums contained data that extended novel phonological situations in informal Tagalog.

With e-Linguistics, reader modules are used to validate and transform phonemic and orthographic inventories into IPA<sup>7</sup> Unicode. The ontology module maps these resources to a central ontology and merges them into an RDF store. The ELTK can then be used to load these resources into the Python programming environment to query across the data. For example, the ELTK can be used to infer the phonemic and orthographic relationships between disparate writing systems, making queries such as, 'show me all graphemes that are used to represent the phoneme /ɲ/'.

Taking Web-scale phonological research one step further, with e-Linguistics language data from languages with

<sup>7</sup><http://www.arts.gla.ac.uk/IPA/ipa.html>

phonemic writing systems can be migrated from their orthographies to an interoperable format such as the IPA. Once phonological data are rendered interoperable, cross-linguistic phonological questions can be asked at a featural level. Semantic information is embedded in each IPA symbol as an intersection of feature bundles, e.g. /p/ is voiceless, bilabial and plosive. Many phonological questions require large corpora of data, and phonologically-encoded Web-scale resources provide statistically testable data for investigating phonological patterns, variants and conditioning. For example, studying the frequency effects on phonological rule application across word boundaries, or statistically modeling the phonological-relatedness of several languages in a group, becomes possible when resources are available and interoperable. The ELTK makes this possible by transforming data into an interoperable format, and then merging it into an RDF store. It is our intent to expand the current ontological model to include several phonological theories. These theories may then be used to test competing phonological analyses across large amounts of merged linguistic data.

Next, consider the scenario of collecting interlinear glossed text with the goal of creating an RDF store. With e-Linguistics the various readers may be used to process files in common working formats: Praat, Elan or plain text for example. To accomplish this the contents of these files need to be converted to IPA Unicode. For this task we have created a specialized character converter that will convert these specialized encodings into Unicode. Once these data are interoperable in terms of encoding and format, the ELTK can be used to validate and merge these data into RDF. Once again, using the ELTK to load the merged data into the Python environment, questions can be programmatically asked across the data.

Finally, by using the ELTK to transform data into an interoperable format, the ordinary working linguist can easily output their data into other useful working formats for data analysis and presentation. In this manner, the ELTK provides a useful tool that also creates interoperable resources in line with e-Linguistics' goals and the e-Humanities framework.

## 5. Discussion and Outlook

The e-Linguistics Toolkit has been developed to aid the move of linguistics scholarship into a new digital era. We have argued that achieving this requires data interoperability in terms of encoding, format, and content. Once data are made interoperable, data manipulation in the form of validation and merging is necessary. As one way to ensure usefulness, the ELTK offers data transformation to various working formats. Finally, the ELTK leverages NLP techniques to serve the whole of linguistics.

In order to achieve the objectives of the e-Humanities framework, it is necessary for individual fields to take charge and to create their own specific techniques that apply to their own data. One useful outcome of such a research program is a cyberinfrastructure for linguistics. Only when such an infrastructure is in place can truly advanced applications be written, such as ontology-driven, or intelligent search. The e-Linguistics Toolkit is one step in this direction.

## References

- [1] M. Babik and L. Hluchy. Deep integration of python with web ontology language. In *Proceedings of the 2nd Workshop on Scripting for the Semantic Web*, 2006.
- [2] S. Bechhofer, P. Lord, and R. Volz. Cooking the semantic web with the owl api. In *Proceedings of the 2nd International Semantic Web Conference, ISWC, Sanibel Island, Florida*, 2003.
- [3] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifiers (URI): Generic syntax. Technical Report RFC 2396, IETF (Internet Engineering Task Force), Aug 1998.
- [4] C. Bow, B. Hughes, and S. Bird. Towards a general model of interlinear text. In *Proceedings of the E-MELD Language Digitization Project: Workshop on Digitizing and Annotating Texts and Field Recordings*, Michigan State University, 2003.
- [5] N. Calzolari, F. Bertagna, A. Lenci, and M. Monacchini. Standards and best practice for multilingual computational lexicons and MILE (the multilingual isle lexical entry). ISLE Deliverable D2.2-D3.2, ISLE Computational Lexicons Working Group, 2002. [http://www.ilc.cnr.it/EAGLES96/isle/clwg/ISLE\\_D2.2-D3.2.zip](http://www.ilc.cnr.it/EAGLES96/isle/clwg/ISLE_D2.2-D3.2.zip) (2006-07-09).
- [6] S. Farrar and D. T. Langendoen. A linguistic ontology for the Semantic Web. *GLOT International*, 7(3):97–100, 2003.
- [7] Hewlett-Packard. Jena2—A Semantic Web Framework, 2003. <http://hpl.hp.com/semweb/jena2.htm>.
- [8] N. Knouf. bibtex definition in web ontology language (owl) version 0.1. Technical report, MIT, 2004.
- [9] S. Koide and H. Takeda. Owl-full reasoning from an object oriented perspective. In R. Mizoguchi, Z. Shi, and F. Giunchiglia, editors, *The Semantic Web ASWC 2006*, pages 263–277. Springer, Berlin / Heidelberg, 2006.
- [10] O. Lassila and R. R. Swick. Resource Description Framework (RDF) model and syntax specification. Recommendation, W3C, Feb 1999. <http://www.w3.org/TR/REC-rdf-syntax/>.
- [11] W. D. Lewis. ODIN: A model for adapting and enriching legacy infrastructure. In *Proceedings of the e-Humanities Workshop held in cooperation with e-Science 2006: 2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam, 2006. Available at <http://faculty.washington.edu/wlewis2/papers/ODIN-eH06.pdf> (2006-10-29).

- [12] B. Motik and U. Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In *Logic for Programming, Artificial Intelligence, and Reasoning: Proceedings of the 13th International Conference, LPAR*, pages 227–241. Springer, 2006.
- [13] R. J. Rayid Ghani and D. M. G. Mining the web to create minority language corpora. In *Proceedings of the 10th international conference on Information and knowledge manage*, pages 279–286, Athens, Georgia, 2001.
- [14] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.
- [15] C. M. Sperberg-McQueen and L. Burnard, editors. *Guidelines for Electronic Text Encoding and Interchange, TEI P4*. Text Encoding Initiative Consortium, Oxford, Providence, Charlottesville, and Bergen, 2002.
- [16] K. Zuraw. Using the web as a phonological corpus: a case study from tagalog. In *Proceedings of the 2nd International Workshop on Web as Corpus*, 2006.