

The African Anaphora Project

Ken Safir, Andrei V. Anghelescu, Sarah E. Murray, and Jessica Rett

Rutgers University
Department of Linguistics
18 Seminary Place
New Brunswick, NJ 08901
afranaph@rci.rutgers.edu

Abstract

The goal of the African Anaphora Project is threefold: to elicit a research-directed database of African language data, which is collected and analyzed with native speaker linguist consultants; to organize and present this collected data in a manner such that it is as widely accessible as possible; and, to provide a forum where project directors, consultants, linguists, and in general anyone interested in African languages or linguistics can share an interactive community research space. The achievement of these goals necessitates the development of an interactive, dynamic site capable of allowing many users with many different objectives to access, input, edit, search, and browse data. Such an implementation requires sophisticated language resources, such as a site management component, a data storage component, and a query component, as well as a tool to input and display Unicode correctly, and tools to export the data in a variety of formats. It is our hope that our project design and our technical implementation will be generalizable to other projects that seek to collect complex linguistic data online and make it available to online users.

1. Introduction

The African Anaphora Project (NSF grant BCS-0523102, Ken Safir, Principal Investigator) was established to develop in-depth descriptions of a wide range of African languages in order to facilitate linguistic research into the nature and distribution of anaphora. Anaphora, in the sense intended here, is the phenomenon where one linguistic form, such as a pronoun, reflexive or reciprocal, refers back to a previously mentioned form in the sentence or discourse. This phenomenon is common to every human language; the African Anaphora Project (=:AfrAnaph) aims to explore anaphora — its forms, distribution and interpretive effects — for every African language with a native speaker consultant who is willing to help us by filling out an in-depth anaphora questionnaire (=:AQ).

Although our project is grounded in a generative framework, and our focus has been the phenomenon of anaphora specifically, it is our intention to insure that the data we collect will be accessible to anyone interested in learning about these languages or in more general and/or theoretically oriented typological research.

Our theoretical bias is toward nativist accounts of language competence, hence we posit a universally available language forming capacity in human beings that operates no differently for speakers of African Languages than for speakers of any other language. From this point of view, even if the world's languages vary enormously, thanks to relatively small formal differences with large effects, or because lexicons and phonological forms must differ, there are commonalities which may be extrapolated through careful study. In other words, there is a core plan common to all the world's languages that can be most profitably studied by closely examining how they are similar and how they vary. For example, certain constructions are unavailable in every human language; accordingly, we are just as interested in ungrammatical forms as we are in grammatical ones.

Anaphora is an interesting object of study because of its variation cross-linguistically; languages vastly differ in what forms can be used as anaphors and how the

meanings of these forms are distributed. Additionally, its study is of crucial importance because it has provoked vast debate in the generative project, inspiring slews of different theories which account for varying ranges of data. We desire that through careful elicitation and study, the AfrAnaph project will help to shed some light on this debate and inform the current and future theories.

However, the goals of the AfrAnaph project extend beyond furthering generative research into this specific phenomenon. We fully hope and intend that researchers with goals and perspectives different from ours may find our data pertinent and accessible. Thus, this project was designed with broader interest in mind, and both the range of data and the technological contributions will reflect this.

This project is feasible at this point in history not only because there are an unprecedented number of trained African linguists who are potential participants in our project, but also because digital technologies and the resources of the Internet make it possible for more efficient remote participation. However, the technological challenges which Africa is presented with are also a challenge for AfrAnaph, and every effort has been, and will continue to be, made to ensure that even those with limited access to technological resources will be able to contribute to and access our database.

Since the beginning of the project three years ago, we have differentiated several sub-tasks to achieve our goals. They include a) improving the efficiency of data collection; b) standardizing the presentation of data; c) making the collected data searchable for the research community; and d) implementing a forum where both project participants and the general research community can discuss our data and propose new research initiatives.

These tasks are currently being realized with the development of a new website. Making use of the transactional object database of the open source web application server Zope (<http://www.zope.org>), we are designing Unicode-compliant dynamic HTML templates for use both by consultants — to input the data in a standardized fashion — and by researchers — to search and browse the data. Additionally, to accommodate

possible technological difficulties, we will provide a number of offline resources, like exporting of data and snapshots of the database.

Section Two of this paper discusses the background of the African Anaphora Project as well as the specifics of its goals. Section Three discusses how we have begun to implement these goals, the software we are using—Zope—and how it both helps to achieve the specific project goals as well as the goals of the larger language documentation project. Section Four is a brief summary of where the AfrAnaph project is today and where it will be in the near future.

2. Project Background and Goals

Our data collection begins with invitations to native-speaker linguists of African languages to fill out a comprehensive Anaphora Questionnaire. In follow-up interactions with consultants, we analyze AQ responses, expanding and elucidating whenever issues of particular interest are uncovered. Previously, the AQ was available online for download; most consultants completed it offline on a word processor and then emailed it back. The data would then have to be transposed into a form presentable on the website and consistent across surveys. However, we have recently developed a dynamic input interface through which consultants can directly input the data into our database, returning later to finish the questionnaire or to edit the data. This method is much more accurate and efficient. Once the data is approved, it is accessible to anyone who accesses the AfrAnaph webpage. The technical details of this development will be discussed in Section 3.

In addition to the language data, each language has its own "case files" containing the results of the questionnaire (which is supplemented by follow-up data elicitation focusing on unique and rich aspects of that language), a short grammar sketch of the language, a sketch of the anaphora system (highlighting aspects of anaphora in the language that appear to raise interesting theoretical or analytic issues), a bibliography, and links to related linguistic and cultural resources. In the future we hope that each case file will also contain audio files, papers available on line, responses to additional questionnaires on other topics, and language-specific bulletin boards and venues for online discussion between researchers. Case files for Berber, Bukusu, CiNsenga, Urhobo, and Yoruba are already complete, and case files for 15 more are in development. Completed case files can be viewed on our website, <http://www.africananaphora.rutgers.edu>.

Our previous method of eliciting data was to send an informant a questionnaire, which he or she would fill out, and then send back to us. After the questionnaire had been filled out, a long process with a fair amount of back and forth to clarify the responses and questions would have to be undertaken. Additional follow-up questions were asked, which were designed, for example, to explore questions that can be profitably asked for those domains where one language has a more articulated set of distinctions than others.

This method of elicitation is clearly limited. Though it has allowed us to collect invaluable data from a number of languages, there is clearly room for improvement. The goals for the new website have been prompted by challenges we've faced eliciting, recording and publishing

data over these past three years. Most languages we study have a complex alphabet as well as complex tonal systems. Faithful elicitation and reproduction of this linguistic data is a challenge, especially when the informants don't have access to PDF writers and readers. This required that our informants and interested researchers download a Unicode font, something that was often a challenge for many informants without access to optimal or consistently available internet resources.

Additionally, since each informant was completing his or her AQ on his own word processor, not all AQs had a consistent format. The data, often 70-80 pages, was not at all indexed, much less cross-indexed, and was cumbersome to navigate.

The goals then for the new website are to have a standard method of data entry, so that all of the data will be formatted correctly, will display correctly (with all tones and diacritics), and will be easier to navigate, for both the consultants and project directors during follow-up sessions, and for researchers browsing and searching the data.

It is clear that improving data elicitation and recording will save time, make the data easier to search and study, and improve the accuracy of data recording. At the same time, the new website will provide a venue for research discussion and debate, between the project researchers and informants and the general research public.

We are revising our offline input template for the AQ response and aligning it with the online data entry option using our dynamically generated interface pages. The former option insures that we do not exclude potential consultants whose access to internet and software resources is limited. For researchers, the data will be exportable in a number of formats, including HTML, XML, and PDF. Website discussion boards will be provided to meet objective (d) above. As our methods and our researcher network evolve, we hope they can be applied to a wide variety of linguistic phenomena, while creating a community space for research into African languages.

3. Implementation

As stated above, the goals of the AfrAnaph project necessitated the development of a language resource specifically designed to accommodate it. In creating such a resource, we believe that we have also developed a tool which is generalizable—one that can be easily modified for use by many similar projects.

Essentially, what we are developing is a dynamic version of the Anaphora questionnaire which we have been using (<http://www.africananaphora.rutgers.edu/NSF-ques.pdf>). This online version of the questionnaire is broken up into subsections organized by topic, and the consultants may fill out any question in any order, save these responses, and return at a later time to finish other questions or edit their responses. This data will be kept private until approved for public viewing, a task done by project director under the consultant's approval. These responses will then be available to anyone who searches or browses our site.

The answers to the questions can be viewed individually, or the whole questionnaire for a language may be viewed. Additionally, researchers may choose to

view the answer to one or several questions from several languages, in order to compare the forms.

We have designed HTML templates for use both by consultants—to input the data in a standardized fashion—and by researchers—to search and browse the data, by any combination of language, language family, grammatical phenomenon, morphemic gloss, or substring. Consultants will be able to input their data in the IPA, complete with diacritics, through use of a customized keyboard or keyboard shortcuts, and both methods can be personalized by individual informants for the needs of their language. Additionally, our morphemic glossing convention, based mostly on the Leipzig standards (Bickel et al, 2006), will make searching easier and more efficient, while minimizing, if not eliminating, the need for tagging.

This remainder of this section presents the technical details about the implementation of the Anaphora online database, as follows: a general introduction to online databases, the technical solution adopted, the hardware and software requirements, the data input tools, and offline access. We conclude with a description of the offline facilities.

3.1. Online databases

The Anaphora server implements a web-based content management architecture that allows end users to collaborate towards building a database of answers to a pre-defined questionnaire.

The database is implemented using the Zope Object Database (ZODB) framework, which follows the object oriented database paradigm. In contrast to the more traditional relational database systems (RDBMS), the data is not represented as a set of tables with relationships designated between them. In fact, the representation of the data is a natural description of the information stored in the database. For example, a questionnaire is structured into sections, which together compose the questionnaire. According to an object oriented representation, the questionnaire would then be represented as a container filled with objects representing sections. Analogously, sections are objects, which may contain other objects (e.g. subsections, questions, etc).

The object oriented representation of such a structure has several advantages over a representation using a relational database model. Conceptually, it is more intuitive, because it follows closely the structure of the questionnaire to be stored. Each item in the questionnaire (e.g. section, subsection, question, etc.), is an object with properties and methods associated to it. This allows for great flexibility in the design of the functionality presented to the user, since the programmer can conveniently write code for any particular class of objects and this code is stored in the database as part of the object itself. The hierarchical representation of data, closely tailored to the purpose of the database (to store and query the answers to a questionnaire), allows for very fast searches, using tree searching algorithms. Such searches are an order of magnitude faster than searches in tables of data when using tree agnostic relational database systems (most common systems available today).

The main interface to the database is network oriented, using a web browser front end over an encrypted communication channel. The user is presented a dynamically generated website, where each page contains

the information associated with the particular object displayed, as well as links to the actions allowed for that context. For offline processing purposes, the data may be exported in a text format, described using XML.

3.2. The Implementation of the Anaphora Database

The server is built on top of the Zope 3 framework, which uses Python as a programming language and XML for configuration files. The Zope 3 framework provides utilities for various common tasks, such as template-driven dynamic HTML generation, user authentication and data storage. The remaining tasks, such as data querying, have been written from scratch, taking advantage of the object oriented model provided by the Zope Object Database (ZODB).

The server, a *site* in Zope parlance, is structured in the following main components:

- a site management folder, responsible for user registration and authentication
- a data storage component, containing trees of objects, as follows:
 - user annotations (e.g. user profiles)
 - questions
 - answers to the questions
 - general information about the languages present in the database
- a query component, which handles queries to the database

3.2.1. The data storage component

The site is a hierarchical structure of objects, with the root of the site containing the questionnaire, language information folder and the site management folder. Each of these components is a container on its own right.

The questionnaire is structured in sections (or question groups), each of which may contain questions and subsections. The only restriction is that a question is not a container, in the sense that it may only store its text (description) and information about the type of answer (e.g. yes/no, string). As such, the questionnaire may be arbitrarily deep, although for practical purposes one should devise it so that it doesn't contain more than four levels.

A question is a structure containing the following:

- description: a free-form optional text (HTML tags are honored), containing some information about the purpose of the question.
- text: a mandatory text (HTML tags are honored), containing the actual question.
- answer type: the type of allowed answer. This may be one of the following:
 - string: a one line, free form string. No HTML allowed
 - boolean: yes/no
 - single-choice: choices are provided by the editor of the questionnaire (HTML tags are honored)
 - multi-choice: choices are provided by the editor of the questionnaire (HTML tags are honored)

For efficiency, the user input is converted to the appropriate data type: yes/no become booleans, single choice answers become numbers, and multi-choice answers are stored as sequences of numbers. The

reasoning behind this choice is that it makes searching these data types efficient.

In general, objects may contain HTML tags in their descriptions and contents. Since some of these are dynamically inserted in the HTML page generated on the server, this leaves some of the server code vulnerable to malicious HTML code attacks. For this reason, only the administrator and the editor of the questionnaire are allowed to input such code. The data input by all other users is interpreted literally, and the HTML code is displayed as plain text, with tags included.

The language information folder is a list of objects containing general ethnological information about the languages represented on this site.

3.2.2. The site management folder

The site management folder contains the information associated with the registered users. This information contains the user profile, the data the user contributed to the database and the workspace of the users (e.g. query objects saved by the user). The workspace and the personal information are accessible only to the user, while the contributed data is public (the user may opt to not be publicly associated with the data).

The access to the site is built around the notion of "connection", where users authenticate using a login name/password combination. The action of authentication creates a *user authentication token*, which is transiently stored in the browser and sent with every request. The absence of such a token is interpreted as lack of authentication, the user being then assigned the status of guest by default. The set of resources and actions available in a context may be restricted using security policies. These policies are defined by a privileged user (root). A policy defines what classes of users may access a resource (e.g. view the content of an object or call a method of an object). Subsequently, users are assigned to one or more classes. The granularity of such a system of permissions allows to specify permissions for each object individually, or to define classes of objects and their interactions.

A user is represented as an object that stores the following information:

- authentication data (login name, password)
- personal information
- data contributed by the user
- a user workspace, which contains queries saved by the user

The login name is chosen by the user, and it must be unique in the database. This is a somewhat artificial constraint, since Zope does allow for users with the same login name, but is implemented only for public identifiability concerns, of particular relevance in the forums. As the number of users is expected to be small (by modern standards), this constraint will not be a concern. However, it is important to note that every user is assigned an unique numerical id, which is used for indexing purposes by the system.

This id is also used when exporting the data to XML. In UNIX tradition, the password is never stored on the server: instead the server generates a hash using the Blowfish algorithm, and stores this hash. At every login, the hash is re-calculated from the password input by the user, and the authentication is granted only when the tested hash is identical to the one stored on the server. The

Blowfish algorithm allows for passwords of arbitrary length, and it is sufficiently expensive to offset dictionary based attacks, and should provide our users with a requisite level of security.

The minimum of information requested from a user is a login name and a password. There are no requirements of personal information; this option is left to the user, who may also specify the information as private (to be viewed only by the project directors and administrators) or public (able to be access by either other registered members of the site or to all users).

Each user will have a certain set of permissions, which will allow the possibility of entering data, editing it, or simply browsing and searching. By default, registered users are not allowed to input data. To achieve the status of contributor, the user must request such credentials from the administrator of the site. This can be handled entirely online, and will provide safeguards to ensure the accuracy and the integrity of the presented data.

For the user who does not wish to, or for some reason can not, register, the database may be browsed without registration. For users who do not desire to input data, the major benefit to registration is the workspace where the user may save queries.

3.2.3. The query component

Given the multi-variate nature of the data, with diverse features (e.g. language, type of answer, goal of the question asked, user), the most natural and flexible approach to querying it is to represent the available data as an amorphous block.

The user may then use a drill-down method, imposing successive constraints on the data features to reduce the number of matching entries to a set that meets a goal. Each such simple constraint limits the values of one feature of the data. The entire filter is the conjunction of the simple constraints it contains.

As such, the result of applying the filter on the data is the set of entries that meet all the simple constraints specified in the filter. Formally, this can be defined by the following grammar:

```
< Filter > ::= < SimpleCondition > [ ^ < Filter > ]
< SimpleCondition > ::= < FeatureName > < operator >
                                     < value range >
< FeatureName > ::= < string >
< operator > ::= [not](< | > | = | ~)
< value range > ::= (< number > | < string > | < regular
                                     expression > )
```

Since this is a multiple conjunction, the simple conditions are commutative, so they can be applied in any order. This is important for query optimization, because the condition can then be applied in the order that minimizes the number of operations (i.e. data transferred, which is the bottleneck).

Also, since simple conditions are objects, they can be stored in the database (the workspace of the user), with the entire query being just a list of simple conditions.

In what concerns the performance of the query, the implementation is non-trivial. In particular, the user inputs the constraint as an HTML query (that is, a formatted string), which doesn't match the structure or the data types of the objects in the database. Questions in the database can have one of the following types of answers: free form

strings, booleans (yes/no), and numerical values (e.g. answer no. 3 in a list of five options). As comparing strings is a slow process (even with the fastest matching methods available, it still is linear time complexity), the constraints are first compiled into a more efficient representation.

When a constraint is created, the interpreter extracts the properties of the feature to which the constraint applies, and finds the data type stored in this feature. Then, the value range introduced by the user is converted to the data type of the feature, allowing for faster comparisons (constant time) for the nonstring data types.

Since a query filter is evaluated as the conjunction of all simple constraints it contains, the order of applying the constraints does not change the result. As such, the constraints known to be calculated faster (e.g. numerical comparisons) are applied first, reducing the number of expensive comparisons, because those would be applied to already filtered data sets. This is implemented by a simple search procedure, applied every time the filter is run: select all numeric constraints, and apply them in the order found in the list, then repeat for the string constraints. This step could be made marginally more efficient by maintaining a sorted list of constraints, with those affecting numerical features moved to the top. However, this would be a usability nightmare, since the constraints would end up in a different order than input by the user. As such, this minor performance hit is by far the better choice.

The constraints applying to string features may also be regular expressions. This takes advantage of the facilities provided by Python for regular expression matching, by using its `re` module. Since a regular expression is compiled into a finite state automaton, the structure used to store such an automaton could become somewhat large. For a small number of concurrent users, this will not be a problem. However, it is unclear how it would scale when the number of concomitant users reaches hundreds.

3.3. Language-Specific Input Tools

One challenge for online data collection and presentation is that of character input and display. A common solution to this problem is the use of the character encoding standard Unicode, which can represent the vast majority of scripts, and allow the simultaneous display of many different scripts. However, the problem of data entry still remains: a goal for any online elicitation should be the use of a tool which creates a reasonably simple method of entering characters, eliminating the need for a complex system of keyboard shortcuts. One such available tool is CharWrite (E-MELD 2005), developed by the LinguistList, which allows the user two options for data entry. The first is the user can type in an ASCII character, and then right click to bring up a table of similar characters. The second option is to double left-click on the input field to bring up an interactive IPA chart, which provides the user a workspace in which they may create strings of IPA characters, and then send them back to the input form.

3.4. Software and hardware requirements

The software requires Zope 3.2.0 or newer, and Python 2.4.2 or newer. It has been developed and tested on a SuSE Linux platform (versions 10.1 x86 64, 9.3 i386), but

it should run without changes on any platform that provides the required versions of a Python interpreter and Zope. Additionally, the Python cryptography toolkit (Kuchling 2005) — which provides the Blowfish algorithm for password hashing — is required.

In terms of processing power, a commodity PC should provide sufficient power for handling on the order of 100 concurrent connections.

3.5. Offline Access

We are also revising our offline input template for the AQ response and aligning it with the online data entry option using our dynamically generated interface pages. Basically, we can create an offline snapshot of the questionnaire which can then be downloaded (or burned to CD and mailed) and filled out at the consultant's leisure. This insures that we do not exclude potential consultants whose access to internet and software resources is limited. For researchers, the data will be exportable in a number of formats, including HTML, XML, and PDF, and snapshots of the database will also be available. All data representation will be made consistent with all data-sharing protocols such as those suggested by the Linguistic Data Consortium. For example, we want the data we collect to be converted into a standard method of data representation which will allow the data to be used by tools developed in later phases of the project or by independent researchers. Similarly, bibliographic entries will be made consistent with existing protocols.

All data representation will be made consistent with all data-sharing protocols such as those suggested by the Linguistic Data Consortium. For example, we want the data we collect to be converted into an XML data format, which is a standard method of data representation, one that will allow the data to be used by tools developed in later phases of the project or by independent researchers. Similarly, bibliographic entries will be made consistent with existing protocols.

4. Summary (and Future Extensions)

There are substantial plans for the future of the AfrAnaph project. We plan to develop a French language version of the AQ which will facilitate elicitations for those whose second language is French. Additionally, we will devise new questionnaires for other aspects of anaphora, including a logophoricity questionnaire (developed by Oluseye Adesola and Ken Safir) and a specialized reciprocals questionnaire, but we will also commission questionnaires that explore other well-studied, compact empirical domains that have been known to vary in interesting ways, such as the nature of questions or the nature of specialized focus constructions (in the languages that have them). We expect our research platform to be flexible enough to serve the interests of anyone who has a good idea about what can insightfully be investigated using our resources, including phonological or semantic phenomena. We will add audio files to each case file so that an interested participant can hear different forms being spoken. In addition to important information about intonation that can influence anaphoric interpretation or the acceptability of phrases, well-developed audio files may also be a resource for

phonologists interested in some of the languages in our case files.

In addition to the Listserve that we will develop on our current site, we hope to open a chat room for the members of the community we serve, in order to facilitate interaction and draw together researchers with common interests.

As our usership grows, we will have a bulletin board space and a newsletter space reporting what's new on the site and who is up to what. As our project grows and new work employs some of our data base, occasional papers will be published on the site as a series of technical reports connected with our project.

We hope to develop a software library of open source materials that our users can download for their projects, including tree diagram programs, fonts, other forms of graphic representation, etc., and perhaps more ambitious analytic tools for speech recognition and data analysis, as well as links to other site which have similar resources available. Participants in the project would make themselves available as references so that someone unfamiliar with a particular program could ask the listed reference person for advice on how to download it, install it and use it. However, all informants will have the option of anonymity if they so choose.

We will also develop a detailed case file for each language, which will include a grammar sketch, links to linguistic and cultural resources for the language, and a list of linguist consultants who are willing to be contacted by other researchers with questions about their language. In this way it may eventually be possible to have several consultants available for a novel project on a language specific basis. This will also serve to form links between researchers on a given language, organized around the development of the case file for the language they speak.

Our central business, however, is the careful and directed collection of linguistic data that is likely to be of use to researchers interested in specific questions which require sophisticated cross-linguistic data. A great many more languages need to be explored if we are to develop resources for even a partially representative sample of the languages of Africa on our site, a project that can only be result of many years or work with new and current consultants. On the other hand, it is also part of our ambition for the project that the technical implementation of our database, with its online access, data input, static presentation and data search and manipulation functions, will prove useful to other projects that strive to make linguistic resources available to anyone interested in linguistic research, especially those with limited technological access.

5. References

- Bickel, Balthasar, Bernard Comrie, and Martin Haspelmath. 2006.
<http://www.eva.mpg.de/lingua/files/morpheme.html>
- Kuchling, A.M. 2005. Python cryptography toolkit v2.0.
<http://www.amk.ca/python/code/crypto>.
- E-MELD. 2005. *E-MELD School of Best Practice: CharWrite Index Page*. Online:
<http://emeld.org/school/toolroom/software/charwrite-index.html>

6. Acknowledgements

This project was initiated with NSF grant BCS-0303447 and is currently funded by NSF grant BCS-0523102. Many thanks go to all of our past, current, and future informants; without their participation, there would be no project. Additional thanks go to Oluseye Adesola, the Assistant Director of the project, and an anonymous editor for helpful comments. All errors are the authors'.