LANGUAGE DOCUMENTATION & CONSERVATION

Vol. 2, No. 1 (June 2008), pp. 160-165
http://nflrc.hawaii.edu/ldc/

Kirrkirr

created by Christopher Manning

Reviewed by JAMES MCELVENNY, *University of Sydney*

1. INTRODUCTION. Kirrkirr is a software tool for presenting electronic dictionaries marked up in XML. It provides an innovative interface that allows users to access the content of the dictionary by the semantic connections between words as well as by the more traditional method of looking up entries by headword. The user interface employs color and animation in a clever and appropriately restrained way that makes it engaging and quite intuitive to use (some studies on the usability of Kirrkirr are reported on in Corris et al. 2004).

Kirrkirr was originally developed to work with the Warlpiri dictionary (Laughren and Nash 1983), but it has been designed in such a way that it is possible to port any XML dictionary that conforms to a few structural restrictions into the program. This means that most electronic dictionaries can be ported into the program to take advantage of the interface it provides. The process of porting dictionaries into Kirrkirr is discussed below.

The development of Kirrkirr started in the late 1990s at the University of Sydney as part of a project exploring ways to improve the usability of dictionaries of Indigenous Australian languages for the speakers of those languages. New versions of the program continue to be developed by one of the leaders of the original project, Christopher Manning, who is now at Stanford University.

This review examines Kirrkirr from both the point of view of end users, who use Kirrkirr to search through a prepared dictionary, and dictionary makers, who want to port their dictionary into Kirrkirr.

2. KIRRKIRR FOR THE END USER. The main innovation in the interface is the semantic network view, shown in figure 1. This view displays a network where words in the dictionary that are semantically related are connected together by colored lines. Different colours indicate different semantic relationships. Words in the network jiggle in place (although the movement can be turned off if the user wishes). Users can interact with the network by dragging words around the display and by clicking on words to expand the network with their semantic connections.

Another innovative method that Kirrkirr provides for exploring the dictionary is the semantic domain view, shown in figure 2. In this view the user is presented with several nested ovals, each of which represents a semantic domain within the dictionary. Each oval contains the headwords of the entries that belong in the domain that it represents. Users can zoom in and out to see either many semantic domains or the details of which words are contained in each semantic domain.

Kirrkirr also allows users to search the dictionary by the more traditional method of looking up headwords. The left-hand pane of the Kirrkirr window, which can be seen in both figures 1 and 2, contains a list of all the headwords in the dictionary. Users can also type in the headword they are searching for in the search box above the dictionary entry display panes. Kirrkirr is designed primarily for bilingual dictionaries, and so it contains a function that allows users to search by the glossing language using a re-

Document URI: http://hdl.handle.net/10125/1796

Licensed under Creative Commons
Attribution Non-Commercial No Derivative License
E-ISSN 1934-5275

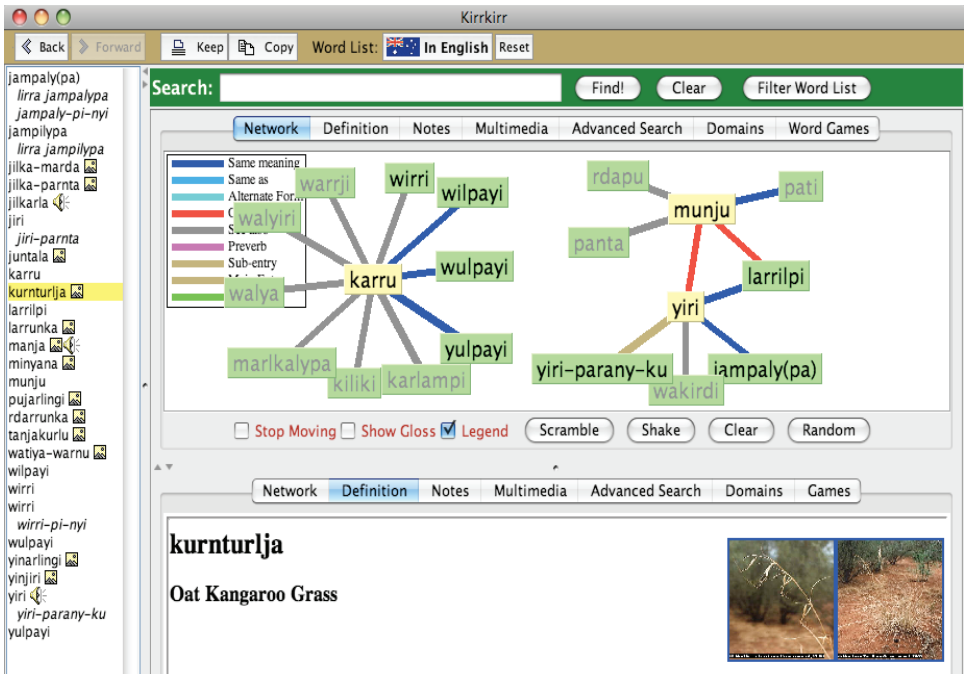

FIGURE 1. Semantic network view.

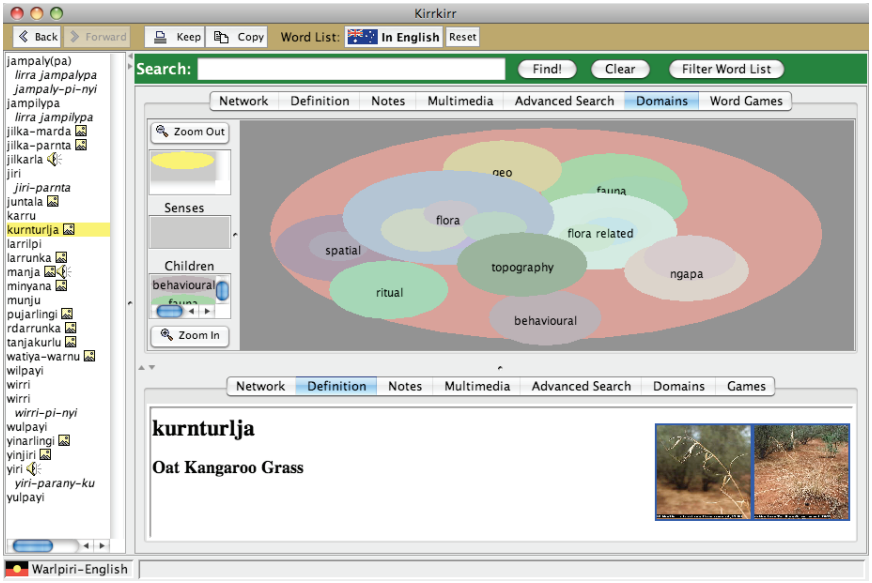

FIGURE 2. Semantic domain view.

verse index. The index pane with the reverse index for the Warlpiri dictionary loaded is shown in figure 3. Dictionary makers can use any field as the reverse index field when they port their dictionary into Kirrkirr, although usually it is the gloss field that is used.

Kirrkirr also has an "advanced search" option that allows users to search through other fields in dictionary entries apart from headwords and to use regular expressions and "fuzzy spellings" in their searches. Fuzzy spellings will return words that approximately match the spelling provided by the user rather than only returning exact matches.

The text of the currently selected entry is displayed in the definition view. The information that is displayed is generated from the data contained within the dictionary entry by transforming it into HTML with an XSLT style sheet. Kirrkirr allows more than one style sheet to be associated with a dictionary. Users can select different style sheets to control what pieces of information from entries they see and how that information is presented. For example, a dictionary may contain both definitions in the language it is describing as well as glosses in another language. If it is being used as a bilingual dictionary, users may want to select a style sheet that shows only the foreign-language gloss. If the dictionary is being used by monolingual speakers to look up the meanings of words in their own language, however, they may want to see only the definitions.

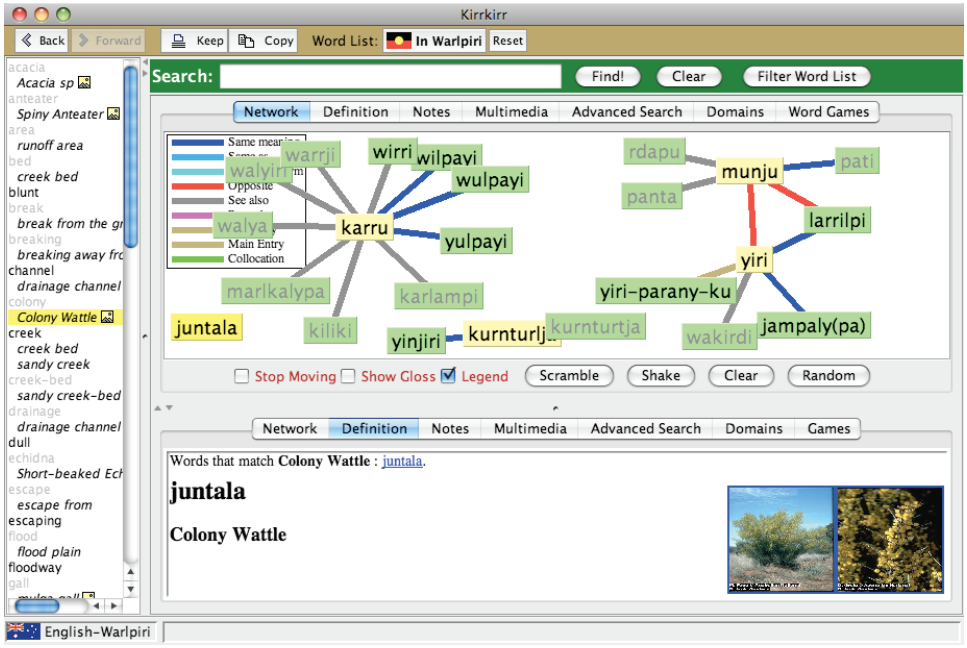

FIGURE 3. Index pane with reverse index shown.

Sounds and images can be attached to entries. Kirrkirr has a special multimedia view that allows users to browse these sounds and images. It is also possible to write style sheets so that sounds and images appear within the definition view. At present, Kirrkirr will only allow one sound per entry to be accessed from the multimedia view, but there is no problem in incorporating more than one sound per entry in the definition view.

Kirrkirr has several games that use content from the loaded dictionary. The crossword game presents a crossword puzzle containing headwords in the dictionary. The picture game shows pictures drawn from entries in the dictionary, and users have to match the picture up to the right headword. In the "word game," users have to match words together according to their semantic relationship. These games could provide a fun way for users to reinforce and expand their knowledge of the vocabulary of a language.

Kirrkirr is written in Java and runs in a Java Virtual Machine. The main advantage of this approach is that Kirrkirr can run on most computer systems. There are Java Virtual Machines available for free for most operating systems; many operating systems come with one built in. The disadvantage of the Java platform is that it can be slow and clunky on an older or lower-end computer system, especially when it is running programs that have a lot of multimedia content, like Kirrkirr.

3. KIRRKIRR FOR THE DICTIONARY MAKER. One of the nicest features of Kirrkirr is that it has been designed so that people outside the project can port their dictionaries into it. The procedure for doing this can be rather involved, but it is within the grasp of the average computer user. Porting a dictionary involves turning the dictionary file into an XML format that can be read by Kirrkirr, creating a specification file for that dictionary file so that Kirrkirr knows where to find entries and headwords and so on in the dictionary file, and creating XSLT style sheets or modifying existing XSLT style sheets so that dictionary entries can be rendered in the definition view. The steps in porting a dictionary into Kirrkirr are outlined fairly clearly and in some detail in instructions accessible on the Kirrkirr homepage (the address is provided in the summary below).

The level of difficulty in turning an existing electronic dictionary into XML depends on its original format. If the dictionary is stored in a machine-readable structured format, such as in the backslash code format produced by programs such as Shoebox/Toolbox, then it is simply a matter of exporting the contents of the file to XML. Many programs, including Shoebox/Toolbox, provide an XML export function. The backslash code format is just a form of mark-up for plain text files and can also be converted to XML with regular expressions.

The only structural requirements to get the basic functionality of Kirrkirr are that each entry must be contained within a single element, and there must be a headword element within each entry. To make use of the other features of Kirrkirr, such as semantic links and multimedia, the elements containing semantic links and references to media files must all be under the same path within entries. The path that is used in a particular dictionary for each type of data is then declared in the dictionary specification file.

From time to time, trivial issues come up with Kirrkirr requiring a piece of data to be expressed in a particular way. I call these *Kirrkirrisms*. For example, earlier versions of Kirrkirr would only accept numbers as headword uniquifiers (current versions of Kirrkirr no longer have this problem). The developers of Kirrkirr are usually fairly responsive in fixing problems like this when they are reported.

Once the dictionary file is in a suitable XML format, the next step is to make a specification file, which is basically an XML-encoded list of XPaths to all the data in the dictionary that Kirrkirr needs to perform its various functions, such as the path to the headword, the path to the semantic domain, and the path to the elements that contain antonyms, syn-

onyms, and so on for the semantic network view. The documentation describes the structure of the dictionary specification file quite clearly. When the instructions are not entirely clear, it is possible to examine the files of the example dictionaries that come bundled with Kirrkirr to see how the files should be structured. A dictionary properties file, which contains just a list of the names of the various files that Kirrkirr needs to read a dictionary, also needs to be written. After these files have been constructed, the dictionary file has to be indexed with the built-in "Make index files" tool. Kirrkirr creates its own binary-coded index files to improve entry look-up speeds.

The most difficult task in porting dictionaries into Kirrkirr is writing the XSLT style sheets for rendering the entries in the definition display. XSLT can be quite complicated, and the Kirrkirr documentation provides only a little guidance in how to write a style sheet. It is possible to examine the style sheets of other dictionaries for an indication of what is required. XSLT is a technology used quite widely in various branches of computing, so dictionary makers who stall at this stage of the process should have no difficulty in finding someone proficient in XSLT who can help.

Primary function:	XML dictionary visualisation tool.
Pros:	The program provides an innovative and engaging interface for electronic dictionaries. It is possible to port existing electronic dictionaries into the program so dictionary makers can take advantage of the interface for their own dictionaries.
Cons:	Since the program is written in Java and makes heavy use of multimedia, it can be a bit clunky on older computer systems. Basic skills in handling XML and XSLT are required to port a dictionary into Kirrkirr. This can be an impediment to less technologically literate dictionary makers hoping to use the program.
Platform:	Java (with various versions tailored to Java running on different operating systems)
Open source:	No
Proprietary:	Freeware
Available from:	http://nlp.stanford.edu/kirrkirr/
Reviewed version:	Kirrkirr 4.0.3 for Mac OS X

Application size:	There are various download bundles that are tailored to different operating systems. The bundles vary between 4.5 MB and 38 MB in size. The larger bundles include the Java Virtual Machine for the target platform while the smaller bundles do not.
Documentation:	There is fairly detailed although somewhat poorly organised documentation available at http://nlp.stanford.edu/kirrkirr/

References

Corris, Miriam, Christopher Manning, Susan Poetsch, and Jane Simpson. 2004. How useful and usable are dictionaries for speakers of Australian Indigenous languages*? International Journal of Lexicography* 17(1):33–68.

Laughren, Mary, and David Nash. 1983. Warlpiri Dictionary Project: Aims, method, organization and problems of definition. In *Papers in Australian linguistics no. 15: Australian aboriginal lexicography*, ed. by Peter Austin, 109–133. Canberra: Pacific Linguistics Series A-66.

James McElvenny
james.mcelvenny@arts.usyd.edu.au