Functional Requirements for an Interlinear Text Editor

Baden Hughes[1], **Catherine Bow**[1] **and Steven Bird**[1,2]

[1]Department of Computer Science and Software Engineering, University of Melbourne, Victoria 3010, Australia
{badenh, cbow, sb}@cs.mu.oz.au

[2]Linguistic Data Consortium, University of Pennsylvania, Philadelphia PA 19104-2653, USA
sb@ldc.upenn.edu

Abstract

Interlinear text has long been considered a valuable format in the presentation of multilingual data, and a variety of software tools have facilitated the creation and processing of such texts by researchers. Despite the diversity of tools, a common core of editorial functionality is provided. Identifying these core functions has important implications for software engineers who seek to efficiently build tools that support interlinear text editing. While few applications are specifically designed for the creation or manipulation of interlinear text, a number of tools offer varying degrees of incidental support for this modality. In this paper we provide a comprehensive set of critieria upon which the derivation of functional criteria can be based. We describe the basis on which a group of tools was selected for investigation, along with the evaluation criteria. Finally we consolidate our findings into a functional specification for the development of software applications for the editing of interlinear text.

1. Introduction

Interlinear text typically consists of a row of source text annotated with one or more rows of morphosyntactic analysis and translation. Vertical alignment represents the correspondence between the elements of each row. An example of a line of interlinear text is given below.

Comment	trouvez-vous	mes	lunettes	de
how	*find.2PL-you.PL*	*my.PL*	*glass.PL*	*of*
soleil?				
sun				

What do you think of my sunglasses?

The rows of an interlinear text can represent many different information types: orthographic, phonemic, phonetic and prosodic forms; morphemes and morphosyntactic categories; word-level and phrase-level translations into one or more languages; and any kind of analytical coding or commentary. Commentary is especially diverse, ranging from cultural notes and cross references to other sources of information that aid the interpretation of the text, to queries about a detail of the transcription and notes on what should be checked in the next meeting with a language consultant. The alignment between the various rows may be morph-by-morph, word-by-word, or even phrase-by-phrase.

Interlinear text is well suited to representation using XML. In previous work (Bow, Hughes and Bird (2003); Hughes, Bird and Bow (2003)) we proposed a model for this expression and explored the flexibility of XML-based interlinear text. While a number of researchers have adopted the models advocated, at the time of writing there is not an interlinear text editing tool which natively supports this format.

In this paper we survey existing tools which enable the creation, editing or presentation of interlinear text, with the view to aggregating a set of functional requirements for an interlinear text editor and corresponding API. In particular we describe the basis on which a group of tools was selected for investigation, and describe the evaluation criteria. In considering current implementations of tools which enable the creation, editing or presentation of interlinear text, a formal set of criteria is needed in order to determine the class of applications to be evaluated in depth. These initial criteria are technically, rather than linguistically oriented. Having selected a group of applications for evaluation, we turn to the establishment of linguistically-grounded evaluation criteria. Building on previous work, we define the evaluation criteria from a functional perspective. Having described our methodology, selected and evaluated the applications, we turn to the primary objective of the paper, reporting a set of functional requirements for an interlinear text tool and API. Drawing on the evaluation phase, we identify a list of commonly implemented functions for working with interlinear text, and the possible degrees of granularity at which these can be implemented. In conclusion, we review the results of our evaluation in the light of other similarly motivated but more narrowly focused efforts.

2. Interlinear Text Applications

A set of criteria is needed in order to determine the group of applications to be evaluated in depth. In applying these criteria we were able to identify a closed set of applications on which to conduct comparable evaluation. Compiling an initial list of tools which offer an interlinear mode for annotation of language data results in over 40 application instances. In considering these tools, there are obvious clusters which provide specialist functionality in one area or another. In order to efficiently review a number of these tools in depth, a selection process is required to categorise these tools. These initial criteria are technically, rather than linguistically oriented (in contrast to the evaluation criteria specified later). Six selection or exclusion criteria have been used: (a) end-user applications rather than application development frameworks; (b) applications which are easily obtainable at low or zero cost; (c) applications which do not require a high degree of technical literacy to install and operate; (d) applications which can be used in multiple contexts (eg. field research, analysis, data management); (e) applications which allow unimodal (text) and multimodal (text, audio, video) input and (f) exclusion of applications primarily oriented towards presentational formats.

After conducting an assessment of the more than forty applications which claimed to support an interlinear modality, a short list of candidates for evaluation was reached: Shoebox[1], Praat[2], TASX[3], InterTrans[4], LinguaLinks[5] and ELAN[6]. (For efficiency, we will not describe these in detail but instead refer the reader to the relevant websites for each application, as well as to the URL listed in the conclusion for further details.)

3. Evaluation Process

The methodology followed in the evaluation process is to use real linguistic data, rather than the samples or demo data which is typically provided with the software. This process is motivated by two factors: first to ensure that the applications were capable of handling a single, consistent set of data as a baseline for comparison, and secondly to replicate typical usage patterns, that is, the creation and manipulation of interlinear text drawn from fieldwork-based linguistic data collection. Where a particular application was available for more than one platform, as many platforms as available were installed, and evaluation of functionality occurred across the range of installed versions rather than on a single operating system installation.

Having selected a group of applications for evaluation, we turn to the establishment of linguistically-grounded evaluation criteria. Building on previous work in which we considered in depth the nature of interlinear text (Bow, Hughes and Bird, (2003)) and identified typical linguistic analytical requirements (Hughes, Bird and Bow, (2003)), we can define the evaluation criteria from a functional perspective. In particular, the six main functional groupings used for evalution used are: (a) general editing functions (e.g. cut, copy, paste, search, replace); (b) structural segmentation and alignment (ability to interlinearise at different levels, and across a range of media); (c) flexible content model (support for ontology-based annotations, and the ability to include incomplete or ambiguous annotations, plus structured and non-structured supplementary notes and observations, and to add these across segmentation boundaries; (d) the ability to import and export structured data according to a formally specified structure and widely supported media format; (e) the ability to handle non-Roman scripts and/or Unicode; and (f) the ability to support a customisable presentation output, e.g. for publishing. For each of these linguistically-oriented criteria, we considered how each application instance meets the functional requirements for an interlinear text editor and API.

[1]http://www.sil.org/computing/shoebox
[2]http://www.praat.org
[3]http://tasxforce.lili.uni-bielefeld.de
[4]http://agtk.sourceforge.net
[5]http://www.ethnologue.com/LL_docs/contents.asp
[6]http://www.mpi.nl/tools/elan

4. Functional Requirements

Drawing on the evaluation phase, we identify a list of commonly implemented functions for working with interlinear text, and the possible degrees of granularity at which these can be implemented. We note that there are a number of such functions which were obvious prior to evaluation; whilst others were discovered during evaluation; still more were discovered when considering the synthesis of results; and others we can derive from previous work (eg. Bickford (1997); Kew and McConnel (1997); Maeda and Bird (2000); Bird et al. (2002); Maeda et al. (2002)) that are not immediately obvious in any single application instance evaluated.

4.1. Requirements Derived from Selection Criteria

A number of functional criteria can be derived from the process of selection of candidates for evaluation. We identify the need for both an end user application as well as a library for software developers. The end user application should facilitate work in different phases of a project lifetime, including data collection, data analysis, and data management. With the emergence of digitized multimodal input, it is also important for applications to allow unimodal (text) and multimodal (text, audio, video) input. While much existing interlinear material is in text format only, the proliferation of multimodal materials requires that a tool be able to handle alignment of text to other modes, such as audio and in some cases video. The library for software developers should be provided in a platform independent manner, accessible via wrappers for commonly used languages of implementation. Furthermore, the distribution conditions of such an application and library would ideally be under open source conditions - in fact for the library this is almost essential. Not only does such a distribution model facilitate the maintenance of software, but also reduces the complexity of licensing administration.

4.2. Requirements Derived from Functional Evaluation

In addition to the requirements described above, there are a large number of functional requirements derived from the process of evaluation.

4.2.1. General Editing Functions

Foundational to any editing function is the notion of how a region of interlinear text can be selected - here we propose that this selection can be made of a sequence of one or more constituents at the level of character, morph, word, or phrase. Further differentiation is required between a selection of (part of) the content and of the structure in which the content is embedded. For example it should be possible select across cell structures to obtain only cell content, or cell structure, or both.

We find that there are a range of functional requirements for an interlinear text editor which fall into three main categories: those related to copy, cut and paste; those related to search; and those related to replace.

A basic copy, cut and paste function is required - anything that is selectable can be copied, cut or pasted. In addition, support for a multi-item clipboard function is common. More complex functions such as splitting or merging can be handled by combinations of copy, cut and paste. In the instance of copy, cut or paste between different levels of a text, full orthographic support should be provided.

A search function should support for regular expressions, matching restricted to a particular row of interlinear text, searching multiple texts, searching within a specified extent of text/audio/video, searching based on previous searches, and navigation between results either via an index or within the data itself.

In the third instance, replace, there are similar requirements: replace, replace with support for regular expressions, replace in multiple data files, replace within a selected quantity of text, replace based on previous replacements and optional replacement at a particular location in the data.

In addition to these requirements, there is also the need for multiple levels of undo or redo.

4.2.2. Structural Segmentation and Alignment

The most significant issue in this area is the granularity of segmentation of an interlinear text resource, which impacts directly on the number of locations to which annotation can be attached. Minimally we find that an interlinear text should be segmented at the morph, word, phrase or text levels. An additional feature which is supported in a number of tools is for attachment of an annotation to multiple continuous constituents (eg. morphs or words.) A further extension is to allow the attachment of commentary to non-constituents.

In reference to annotations themselves, it is a requirement that the editing environment and underlying API support the use of multiple ontologies for linguistic annotation. (Directly supporting XML namespaces appears the obvious way to enable this.)

The degree to which text can be combined with audio and or video is important. Possible variations which should be supported include: text only, text and audio, text and audio and video, and text and video. In all of these cases, support should be provided for separate management of annotation tiers, as well as support for annotations across a number of independent instances of text, audio and video through the use of a URI resolution schema.

4.2.3. Flexible Content Model

A flexible content model is highly desirable, and here we seek to define some of its dimensions. With regard to annotations themselves, it should be possible to include partial annotations or free text commentary. A further requirement is to support third-party commentary in a stand-off mode. Support is required for annotations based on third-party ontologies.

4.2.4. Import and Export

The ability to import and export structured data according to a formally specified structure and acceptable media format is an important functional requirement. For maximum flexibility, the editor application and the API would handle interlinear text natively in an XML-based format. However, any new application also needs to retain backwards compatibility, and as such, it is a requirement that an interlinear text editor and API must also handle conversion to and from formats such as those used by Shoebox, TASX and ELAN; and to and from formats such as SGML, some variant of HTML and structured text files. Where possible, this conversion should be lossless. Yet another desirable variant would be to support a change management and version control system natively in the application and API.

4.2.5. Non-Roman Scripts

We consider the ability of an interlinear text editing application to handle non-Roman scripts and/or Unicode as essential. In particular, we find that many extant tools do not support Unicode character encoding, which is perhaps surprising given the prevalence of special characters used by linguists. However, any future tools should minimally support Unicode (both UTF-8 and UTF-16 encodings). While Unicode support is valuable, an additional but significantly more complex requirement is that rendering support for non-Roman scripts is also available, rather than simply data entry support. In addition, there is a requirement for support of interlinear text in horizontal directionality (right-to-left and left-to-right), and in vertical modality (top-to-bottom and bottom-to-top).

4.2.6. Presentation Output

The ability to support a customisable presentation output is a requirement which is not achieved easily by existing implementations. We find that there are a range of functional requirements for an interlinear text editor in this area, specifically: the ability to save a selected amount of text as an image in a range of formats including GIF, JPEG, EPS and SVG; the ability to generate an interlinear representation in a number of presentation formats including PDF, RTF and HTML; the ability to transform native XML based interlinear text using a range of inbuilt XSL stylesheets derived from common publishers' requirements; and the ability to transform native XML based interlinear text based on a user defined XSL stylesheets or XSLT transformations.

5. Conclusion and Future Work

Interlinear text is a popular format for presenting multilingual data. In this paper we have reported on our study of several interlinear text editors, and elaborated a comprehensive set of functional requirements. We are in the process of implementing this functionality in an open source library and application based on the Annotation Graph Toolkit[7] Additional resources, including the survey data is available.[8]

[7]http://agtk.sourceforge.net/

[8]http://www.cs.mu.oz.au/research/lt/projects/interlinear/

6. References

J. Albert Bickford, 1997. A Rich Model for Presenting Interlinear Text. SILEWP 1997-003. SIL Electronic Working Papers. Summer Institute of Linguistics: Dallas TX. [http://www.sil.org/silewp/1997/003/]

Steven Bird, Kazuaki Maeda, Xiaoyi Ma, Haejoong Lee, Beth Randall, and Salim Zayat, 2002. TableTrans, MultiTrans, InterTrans and TreeTrans: Diverse Tools Built on the Annotation Graph Toolkit. Proceedings of the Third International Conference on Language Resources and Evaluation, Paris: European Language Resources Association, pp 364-370.

Catherine Bow, Baden Hughes and Steven Bird, 2003. Towards a General Model of Interlinear Text. Proceedings of EMELD Workshop 2003: Digitizing and Annotating Texts and Field Recordings. LSA Institute: Lansing MI, USA. July 11-13, 2003.

Baden Hughes, Steven Bird and Catherine Bow, (2003). Encoding and Presenting Interlinear Text using XML Technologies. Proceedings of the Australasian Language Technology Workshop 2003. Melbourne, Australia. December 10, 2003.

Jonathan Kew and Stephen McConnel, 1990. Formatting interlinear text. Summer Institute of Linguistics Occasional Publications in Academic Computing 17. Summer Institute of Linguistics, Dallas TX.

Kazuaki Maeda and Steven Bird, 2000. A Formal Framework for Interlinear Text. Proceedings of the Workshop on Web-Based Language Documentation and Description, Philadelphia, December 2000.

Kazuaki Maeda, Steven Bird, Xiaoyi Ma, and Haejoong Lee, 2002. Creating Annotation Tools with the Annotation Graph Toolkit. Proceedings of the Third International Conference on Language Resources and Evaluation, Paris: European Language Resources Association, pp 1914-1921.

7. Acknowledgements

The research reported here is supported by the National Science Foundation Grant Number 0094934 (Electronic Metastructure for Endangered Languages Data), Grant Number BCS-998009 (Talkbank), and Grant Number BCS-0317826 (Querying Linguistic Databases).