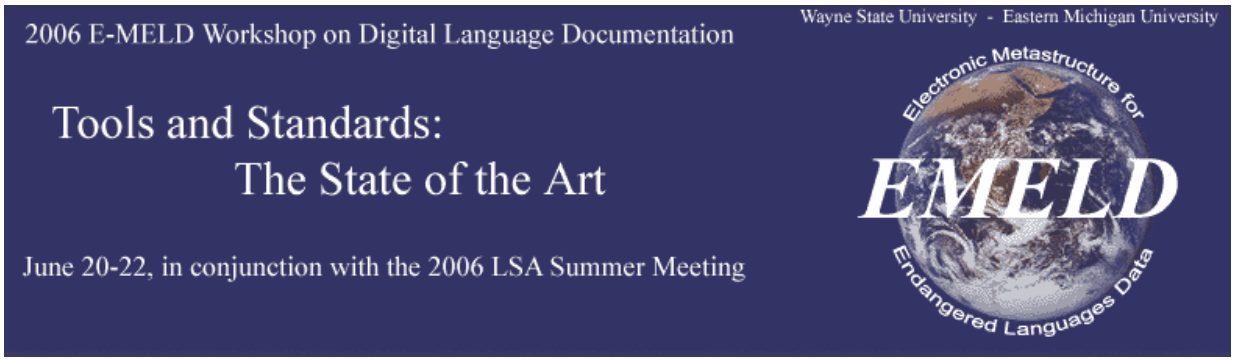

THE MISSING LINKS IN DOCUMENTARY LINGUISTICS: AN APPROACH TO BRIDGING THE GAP BETWEEN ANNOTATION TOOLS

By

Thorsten Trippel

Paper presented at

2006 E-MELD Workshop on Digital Language Documentation
Lansing, MI.
June 20-22, 2006

Please cite this paper as:

Trippel, T. (2006), The missing links in documentary linguistics: An approach to bridging the gap between annotation tools, *in* 'Proceedings of the EMELD'06 Workshop on Digital Language Documentation: Tools and Standards: The State of the Art'. Lansing, MI. June 20-22, 2006.

E-MELD 2006 Workshop on Digital Language Documentation: Tools and Standards: The State of the Art

The missing links in documentary linguistics: An approach to bridging the gap between annotation tools

Thorsten Trippel

2006-05-06

Table of Contents

Abstract

This paper focuses on ways of making linguistic annotation tools for documentary linguistics and archiving interoperable without restricting the features of the individual tools. It is assumed that no tool is exclusively used in the process of documenting and archiving a language but a suite of them according to strengths and the user's needs. The tools have frequently been discussed by developers and archivists in terms of individual tool portability, but the interoperability of different tools has been neglected generally though XML is now accepted for data structure specifications. This paper designs and discusses the architecture of the middleware for linguistic tool interaction.

1. How to avoid expensive and time consuming data-re-collection

Projects with a multitude of people working in the same project on language documentation are often faced with a lot of challenges regarding the use of software tools:

- different persons have different personal preferences, such as for writing and publishing texts: one person prefers Microsoft Word, another OpenOffice.org Writer, and a third person LaTεX; the funding agency may require an open format for archiving, such as the Portable Document Format (PDF) for layout preservation, HTML for web access or some XML coding for long time archiving and harvesting;

- different language characteristics can cause problems with a number of tools, such as the encoding of non-segmental tone, e.g. morphological tone, which cannot be easily encoded inline, or the transcription of non written languages using IPA characters (Unicode) rather than some form of ASCII representation
- some parts of a project may have different requirements for software features, such as the recording and transcription part of a project requires a tool to link the audio signal to a possible transcription with playback features, while a part of the project working on generative grammar aspects may require extensive assistance in coding of hierarchies;
- centralized and networking architecture may results in restrictions for software due to infrastructure: if in a documentation environment there is no internet access, a networking infrastructure does not allow the use of a central repository for backup, access and storage.

These are only a few possible but obviously valid reasons why it seems to be impossible to use one tool only for the documentation of languages starting from the data recording to complex linguistic analysis on semantics, grammar, typology, etc.

However large the difference between the software tools may seem, looking both at the underlying data format and the tools as such, the content of the annotation has a lot of overlaps. An example for the overlapping part can be seen in the orthographic transcription found very often on all levels of annotation, be it time aligned annotation on the recording side or hierarchically and highly structured annotation of other linguistic levels. To reuse the annotation of one tool very often converters are written to create an input data format for one tool from data of the other.

Converters for data written in such projects are very often scripts based on string manipulations such as Perl scripts or for transformation as XSLT stylesheets. These programs are not necessarily generic but adjusted to that portion of the data needed to continue with the work. This includes that it is accepted that some parts of the original data is not transformed into the target format, and very often some parts are added based on defaults that are project specific, often hard coded in the program. In other cases the transformation is done manually, i.e. a person re-creates the annotation with or without the knowledge of previous work with another tool.

The problem with converters written in an ad hoc, project specific fashion is that there has to be a programmer working on the team or the researcher needs to be aware of the tools before he or she can actually use it. Linguists not being *technophil* and concentrating on the linguistic aspects of language documentation will rather unlikely be able to run these programs without any technical assistance. The result will be that the language resources are being lost over the time or not fully exploited for the complete information contained in them.

2. What interoperability of tools needs to account for

The goal of interoperability, is to gap the bridge of different annotation tools, platforms and theories, hence being an integrated part of portability as described by Bird and Simons (2003). The theoretical issue of portability is supposed to gain practical relevance. The gap between different software tools can only be bridged on the following premises:

- data reusability: it is to be acknowledged that a lot of the data created in an annotation by one software is to be further processed by other software and in which areas the underlying data overlaps
- tool reusability: a transformation tool from one file format into another does not have to be reimplemented iff it is well documented in terms of which data structures are transformed, defaults to add necessary information for the target format are configurable, the tool is available in a widely available language or environment such as Java or XSLT, and if the tool itself is made available for being reused

- tool variety: without the tool vendors acknowledging the fact that different tools are relevant and necessary, import or export functionality is not to be expected. It means that the software environment is a hybrid suite of programs, created by and within a given project according to the projects focus. The tools cerated by the vendors of software basically are created for their functionality and features, not for interoperability with other tools, which is naturally not in the perspective of the developer.
- usability: the non programming researcher cannot be required of going into the technical details. In fact it is already a form of hindrance for a lot of researchers to install and use an additional program they are not used to. The user interface for them has to have the so called *look and feel* of an already known program for them to be useful. The easiest would be to have an export and import feature in the preferred software, but a specialized transformation tool which is based on a known user interface would be acceptable to most users.

For lexicons the data reusability and underlying structure was discussed by Trippel (2006) and a model of annotations was introduced by Bird and Liberman (2001). This indicates that language resources of theses forms can be discussed based on formal grounds and compared to each other. However the requirements and architecture enabling interoperability of linguistic tools have not been discussed.

The requirements here are rather complex and it is necessary to look into a way of fulfilling these. A general design and architecture should be the result, in this architecture tools can be positioned and describe their functions. This could be an approach to a standard interchange format on different levels of annotation.

3. The world of interoperability of language documentation tools

Language documentation tools have a strong interrelation with each other in the process of a projects workflow. This section describes a possible documentation workflow, tries to look briefly at comparable problems in the field of business applications and then tries to show a design for an integrative suite of individual software tools in the documentation process using a middleware architecture.

3.1. Process of annotation and documentation

The workflow of annotation is a bit tricky, as all steps are optional, given a certain amount of proficieny. For example an expert in one language can start from scratch to create a generative grammar for a language, without recording audio samples and transcribing speech first, collecting texts or working on a wordlist. The reason is, that the language acquisition process of the expert already included, to say it laxly, the recording, lexicalisation, and transcription process.

The workflow described here is intended to be an idealized workflow, starting from scratch in an unknown language. It is not meant to be complete but targeting at the existing tools and processes in the language documentation community. Different models of the workflow are possible, among them are the following based on:

1. Annotation phases
    1. Field phase: data acquisition, recording, initial rough transcription
    2. Laboratory phase: detailed transcription of audio/video material, correction and systematization of rough transcription, wordlists, simple lexicons and sketch grammars
    3. Analytical phase: typological studies, comparison to other results and languages,
    4. Repository phase: archiving for reuse, distribution and long term availability of the data

2. Linguistic levels
    1. phonetic phase: recording and transcription of speech data
    2. phonological phase: generalizations over speech data
    3. morphological phase: segmentation and generalizations over strings of phonemes or a spelling representation
    4. syntactic phase: generalizations over strings of morphemes
    5. ...
3. Lexicographic complexity (see Gibbon, forthcoming)
    1. corpus creation and annotations
    2. lexicalization
4. Tool use
    1. recording phase: physical recording of audio/video material
    2. transcription phase: transcription of the material with or without assistance by speakers of the language and with different degrees of granularity
    3. lexicalization phase: wordlists, concordances, glossing, wordclass tagging, lexical semantic classification,
    4. grammaticalization phase: syntactic pattern extraction/parsing

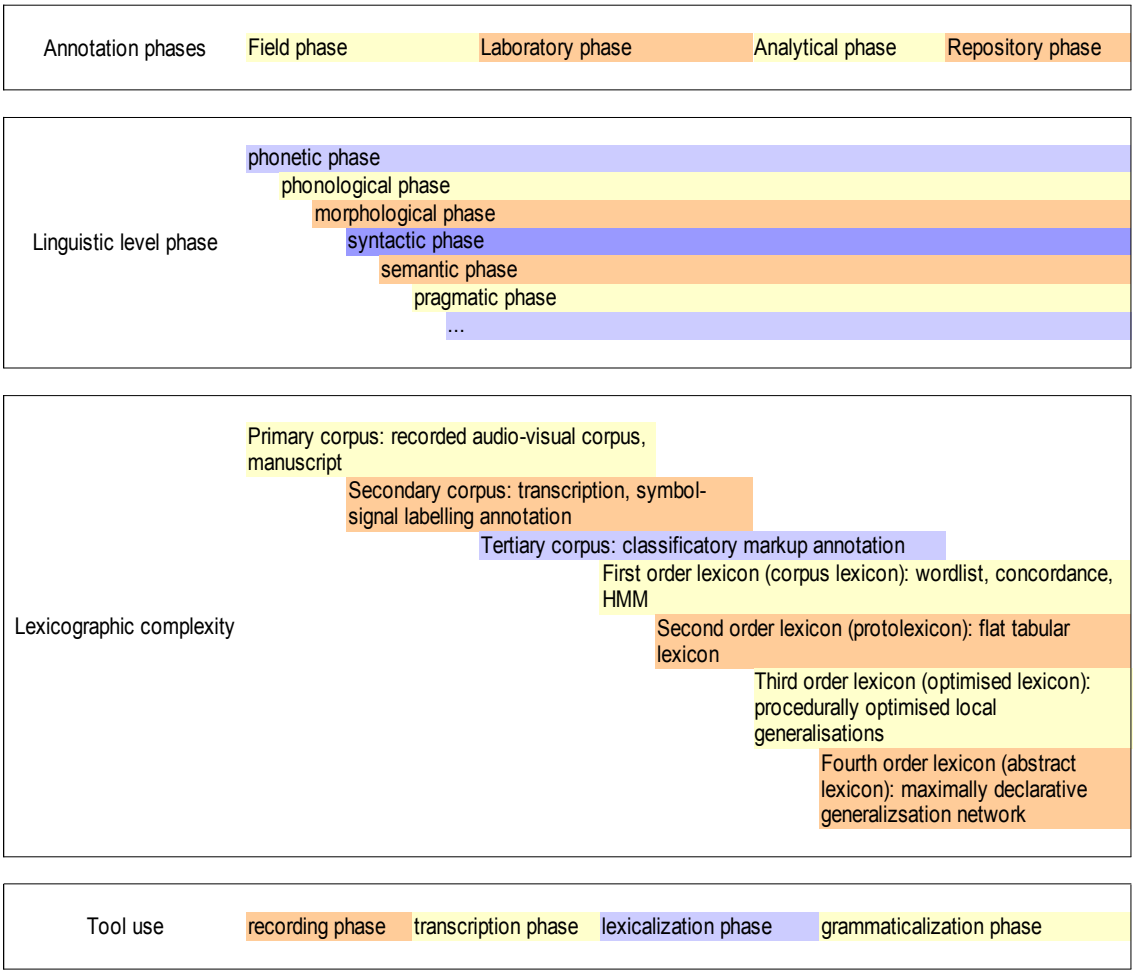

Illustration 1: Relation of the workflow models according to an abstract timeline

Illustration 1 shows an approximate relation of the workflow models according to some abstract time line. The overlapping parts show one of the major problems for an integrative framework for tool use: Though language documentation is likely being based on annotation phases and tools in a linear order with the researcher using one at a time, the work is often crossing linguistic subdisciplines at the same time using the same primary data. Hence software providing for the needs of the researcher in a particular situation will have to relate to a variety of linguistic levels.

Not all language documentation tools show the same strengths on all levels of annotation and analysis, hence it is not only necessary to transform the data from one format into another linearily but also to transform forth and back to allow parallel processing in different subdiciplines.

3.2. Outside of the box: from business integration to language documentation

A similar process of interrelation of different levels going forth and back is well known from business applications. A company with departments such as customer support, sales, book keeping, controlling, warehousing, management, etc. very often has different specialized applications for each of these areas, but they need to address the same data. Sales need to check what is in stock, book keeping needs to bill the customer on what has been shipped, customer support has to explain to the customer what he received or why he did not receive anything. Controlling needs to review all processes, management wants to view the number of customers, the stock, the cashflow, etc. The question is how they treat the problem and how it can be applied to language documentation.

The easiest solution for an integrative software would be to store everything in the same database, having the same data model for everything, such as for stock, customer support, etc. The different usergroups then just need to get different views on the data in the database. However, usually this has not been fully integrated, due to costs and other reasons, hence different computer programs and databases exist. But it is possible to access one database from other applications by mediating software, often also called *middleware*. This mediator gets requests from one application and sends them off to other applications in their required format. Some applications may even specify conditions, such as bookkeeping sending of the bill based on some order recorded by sales but only if the product has been shipped from the warehouse. These conditions mean that one process may be delayed or impossible if another process has not been finished. Another process not requiring the same conditions can on the other hand already be processed. Such integrative software is available from major enterprise software vendors such as SAP or Software AG, and though these business cases are clearly on a different scale there are some things the language documentation community can learn from it.

First of all the language documentation community can learn that a monolithic tool for the all processes in the workflow is rather expensive and probably impossible. If the business people do not see a gain then this is way to large an undertaking for a scientific community. Second it seems to be reasonable to built upon existing tools. This may sound strange for a scientific community trying to find the ideal way to find the golden fleece but improvements of tools can be accomplished on the way, as long as the tools can work together, i.e. compatibility of the tools can be accomplished. Third, it seems necessary to have a mediator between different software programs, which transforms one requirement into another based on a number of conditions and based on the format specifications. These format specifications need to be standardized and in the case of changes versionized.

3.3. System design for an integrative system suite

Based on the idea of tool integration and the idea of having a mediator for different tools it is possible to design an integrated suite of tools for language documentation.

3.3.1. Properties of the Language Data Mediator

The *Language Data Mediator* (LDM) has to have the following properties:

1. check if the source and target formats are known to the LDM
2. what information is provided by the source format
3. what information is required by the target format
4. if the target format requires more or other information than provided by the source format, the LDM has to check for possible other sources of information.
5. transform the information into the target format.

Illustration 2 shows the design of the LDM in a flow diagram. The operations are only possible if an applications data format has been registered with the LDM in a given format which enables the LDM to automatically check for the requirements of the target format and the restrictions imposed by the source format. These requirements and restrictions could be supplied per tool using a restricted vocabulary. In the case of not matching requirements the LDM can check for other sources of information, if the requirements are fulfilled the target format can be created.

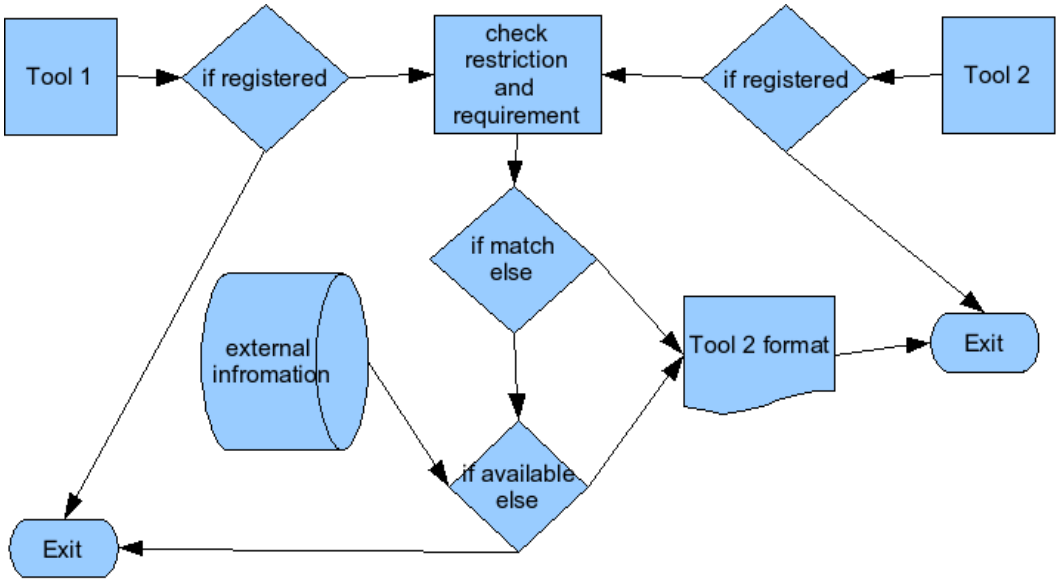

Illustration 2: Flowchart for the LDM

To illustrate this by an example: If a transcription is available in a time aligned format with a start and end time for each word then this transcription provides the orthography and time stamps but no metadata. For using the original data in another tool for phonetic transcription, the LDM could see that this phonetic transcription tool requires start and end timestamps and an annotation of some kind, but may also require metadata such as the recording person, the date and the location. As this is not available in the original the LDM can check on an external file, such as a metadata file which holds the required information and then can merge this information into the output format. In this example it may seem straight forward but what could also happen is another case. Consider a lexicon table which should be used in a time aligned score, an annotation of a signal with multiple tiers. The requirements for a score will be start and end timestamps, a tier specification and an annotation. The lexicon provides data categories that could be used on separate tiers, but no timestamps. However, if from another application there is a wordlevel annotation with timestamps, these timestamps could be used for the creation of the time aligned annotation.

A major advantage of this system is, that given a registration in a predefined format, it becomes obvious, if two formats can be mapped onto each other and which are the conditions. Additionally it is possible to run the applications in the form of web services, i.e. by a technology, where programs

can be distributed via the internet or a *Service Oriented Architecture* (SOA) without needing to install them on a local machine with the ability of distributed computing. However, the system also allows standalone applications on a local machine.

3.3.2. Properties of the user interface

So far the LDM is yet another, possibly rather complex tool with a great amount of questions for the inexperienced user. What is necessary is a user interface allowing the unexperienced user to find their way through the software. As the unexperienced user is not likely to install new software, learn a new graphical or even text interface, another way has to be provided. The optimum would be to have the export function available in the tool, which could as well be integrated as a webservice if those were provided or as a plugin if the tools were providing an architecture for that. However, it is more likely at the moment that this is not going to happen in all tools which are in use, hence a different option has to be found.

The interface most likely to be acceptable for a user is one that is already known. On the other hand the tool has to provide the means of completely new and unknown options. One solution here is to use a client server architecture, the user only seeing the client side of a program running somewhere on a server. One way of doing this without installing a new program is by using a webbrowser as the interface, which is known to the user and where users have seen different of forms and configuration options. A more detailed discussion of the interface design is left out here, as initially an upload and download option is the only requirement with specification of source and target file types. As the number of different file types increases, different options and additional questions may become necessary, but the interface technology could provide for this, for example using AJAX technology, with which the user can get the look and feel of a desktop program within a browser window.

4. Interoperability of linguistic tools at work

This section discusses the interoperability of linguistic tools from the practical side, using concrete examples from a documentation workflow and transforming it into the format required by the next tool in the workflow.

4.1. Tools in the linguistic workflow and a use case

Talking about tools for language documentation requires to know of some of these tools. The tools mentioned here are not meant to be compulsory nor is the list complete, but it is intended to give an idea on possible applications. Starting with the transcription program Transcriber, which is a signal based rapid broad transcription software, a linguist can decide to continue on other linguistic levels. For a morphosyntactic interlinear annotation also including glossing, the Linguist's Toolbox is frequently used for word based annotation and lexicographic work. On the other hand there are tools such as Praat and Wavesurfer for phonetic annotation or the TASX-Annotator or ELAN for signal based annotation based on different modalities. These applications support various combinations of multi-tier text and signal annotations, and metadata administration used for archiving and distributing the corpora.

A possible workflow in a fieldwork situation could then be the following:

1. A linguist could use Transcriber for immediate transcription in fieldwork situations, for rapid data collection on turn or sentence base, roughly aligned to the signal.
2. The sentences from Transcriber serve as the base for morphosyntactic annotation with Toolbox.
3. The roughly signal aligned transcription can be used as the input for a detailed analysis on

word or segment level in score oriented software such as TASX, by taking the Transcriber annotation as a sentence/turn tier and adding more tiers.

4. The Toolbox interlinearization, also multitier, can be used to add morphosyntactic information to the annotation from the phonetic software given a wordlevel annotation there.

Between step 1 and 2 the linguist needs to convert Transcriber data into a sentence based format to import into Toolbox, for step 1 to 3 the Transcriber data needs to be transformed into TASX format. The transition from 3 to 4 requires information from step 2, i.e. the Toolbox file and the word aligned layer/tier of the TASX file.

4.2. Sample implementation

The existing tool supports the source formats Praat, TASX, Transcriber and the output formats Praat, TASX and plain text, using the TASX-format as a generic `lingua franca' format for reasons of simplicity because it permits preservation of all application specific metadata. The implementation from Praat into TASX is implemented in Perl, the other formats are based on XSLT transformations as all those formats are XML based. The user interface is available as a shell script, a HTML based GUI for webbrowsers in client-server architecture is currently in the testing phase. The reason for this is that a server architecture such as this one is a potential security risk, hence detailed mechanisms have to be thought of and implemented for authentication by login and password to prevent server crashes.

5. Problems for interoperability

The architecture shown in this paper that allows interoperability of different linguistic tools relies on the registration of programs with the mediator/middleware LDM and some conversion routine to be available by the LDM. The registration could either be made available by the programmer of a tool also with some conversion routines, but if the programmer does that, there could also be a direct export function integrated. Other options would be that somebody needing and writing a one directional transformation could register the transformation function and at the same time register the source and target tool. Both options require a certain amount of extra work, which may imply a problem.

Another problem is on the side of the data formats. Very few data formats used by the tools are properly documented or distributed with a document grammar such as a DTD or XSchema. And even if the data formats are documented, that does not mean that conversion routines can foresee all possible uses of elements, in fact a lot is left to the interpretation and experience of the transformation creator. For example some people use the option of creating a second speaker in Transcriber to create a second tier for another annotation level, or to use a topic change to indicate a speaker change. The misuse of data categories due to restrictions in the software can create incorrect or unpredictable results when interpreted by software that was created under different premises.

Sharing information and corpora with others is another problem in the development of interoperability tools. This refers to the problem that if there is no data being shared, maintained or archived there is no need for tools working on them and without those tools it can be rather difficult to share language resources.

6. Evaluation of integration and development

The LDM architecture for mediating linguistic data formats based on a client server architecture serves the purpose of transforming different file formats into each other, using a simple webbrowser based interface. The restriction currently are limited to a very small set of import and export file

formats. It allows data and tool reusability, the use of a variety of tools and provides for the usability by unexperienced users without programming background.

7. Future prospects

A major step forward in the process of integration is to be expected as standards in the context of the International Organization of Standardization working group on language resources (ISO TC 37 SC 4) become available. However, as summarized by German members of the committee (see Trippel et al., 2005) standards currently under development are not yet targeted at the interoperability on material used in language documentation but basically in the context of language processing, such as representation formats for feature structures, data categories for lexical information, high level annotation formalisms in the morphosyntactic annotation framework (MAF), etc.

On the other hand the cooperation of linguistic archives using a common framework is going to result in directing linguists in the use of specific tools, which will help to establish interchange formats. It can only be hoped for that this will not result in the reduced support of legacy tools and formats but that legacy data is integrated by sufficient and simple to use transformation tools.

With the ongoing discussion of archiving and standardization, open access policies for publicly funded research in some countries and cooperation requirements there is hope that it will be possible to use the tool best suited to ones need in a given situation and reuse the data in another context with a different but appropriate tool without forcing linguists to program their own tools.

8. Bibliography

Steven Bird and Gary Simons. Seven dimensions of portability for language documentation and description. Language , 79(3):557–582, September 2003.

Steven Bird and Mark Liberman. A formal framework for linguistic annotation. Speech Communication, 33(1,2):23–60, 2001.

Dafydd Gibbon. Spoken language lexicography: an integrative framework. Forum Translationswissenschaft, forthcoming.

Thorsten Trippel, The Lexicon Graph Model: A generic model for multimodal lexicon development, PhD thesis, Universität Bielefeld, 2006

Trippel, Thorsten and Thierry Declerck and Ulrich Heid: Standardisierung von Sprachressourcen: Der aktuelle Stand Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Beiträge zur GLDV-Tagung 2005 in Bonn, Peter Lang Verlag, Frankfurt, 2005