A Model Syllabus of Language Documentation

(1) Introduction to Language Documentation

In this course, language documentation is defined as a set of knowledge on data models and a way of assignation of data conversion.

(2) An Idea of System Engineering; steps of software development

In a field of system engineering, there are four phases in developing software systems: requirement analysis, designing(modeling), implementation, and test. In language documentation, we learn these phases in order to recognize the substantial value and mechanism of requirement analysis, which is needed to understand characteristics of your own language data on computers.

(3) Function First vs. Data First

Usually main targets of requirement analysis are functions, and a type of data is often re-defined by software engineers at the phase of modeling. Thus, it can be called as a function-first approach. However, in the case of language documentation, data models are more important than any functions in that language data is permanent but functions you want at present is temporal. Therefore, a data-first approach is essential in language documentation. This is a kind of new or often-disregarded field for software engineers. That is one of the reason why the value of archives or repositories are often emphasized. The data-first approach is the best for making your language resources life-long data.

(4) Data, Function, and View

Of course, functions are necessarily analyzed as well as data. In this case functions mean data processing. And in addition to functions what kind of view are expected in language data is also analyzed and listed as a member of functions. Data view is one of functions in requirements, but it is often the case that data view is regarded as an independent work flow from implementing data processing work in a computer system. You have to make these three kinds of lists on data, functions and view, and do not confuse data and functions or views. The data is the most important for your data model and your research life.

(5) Data Models

A data model is a bundle of relations, and can be built-up on the ladder of abstraction. In practice we usually use a practical data model that is an abstract and/or compound model.

i. Data format = data units + a bundle of relations

A data format is data with a structure. A structure varies with the interest of users, characteristics of data, requirement from software, and so on. The structure consists of relations of data.

ii. Minimum (primitive) relations

A data structure is a compound of relations. The relations are established by an accessor that ensures a connection. A relation that cannot be divided into any smaller one is called as a minimum(primitive) relation.

A Physical existence (or memories in the brain)

The most basic relation is the existence of a physical object. In this case, the accessor is a memory in a human brain. Physical existence is very primitive but fundamental and useful if you use physical positions systematically. For example, a USB memory itself can play a role of the relation. CD-ROM and other media for recording are also regarded as that.

B Name

Names are probably the most popular way to ensure a relation of data. A name can be a target of a relation and be a relation itself of a set of data. In the case of a computer system, an accessor of the names is called a resolver.

C Ordered pairs

An ordered pair is the most important minimum relation for data models in computer science because any practical relation can be reconstructed by using ordered pairs. In mathematics an ordered pair is a higher-abstract relation than a set. However, in computer science, the set is not so useful in programming. Thus if you make a logical theory of data, you can use a set as a minimum relation, but if you wish to process your language data or make a graphical view, it would be best for you to use an ordered pair as a minimum relation.

iii. Practical (abstract/compound) relations

Actual data models you use are usually compound models consisting of primitive relations, and are typical data structures for ordinary users. There are many kinds of practical relations, but in language documentation the four types of data models are good for you: a record, a regular table, an interlinear table, and a tree structure.

A Record

A record, also called a list, is probably the most popular data structure for linguists on a field. A record is a flat structure, but, if tags are embedded, a record can be used for a complex data structure. The point in processing is that order is a key relation to handle each data unit without embedded tags. Thus, you have to pay attention to the order in making language sources.

B Table 1 (regular)

A regular table is a set of records that have a same inner-structure(thus called regular). A spreadsheet is the typical of a regular table. Data in a regular table is easy to be handle in a database or programming, but it is difficult to represent a complex data model in the same table. A regular table is regarded as a set of records compiled up with the same rule. If you can make your language data in the regular table, you can use a relational database engine as a tool for retrieval.

C Table 2 (interlinear)

An interlinear table is the most popular data structure for linguists. However handling data with this structure is not so easy because if you want to process it in programming languages, the number of sub-units has to be regulated. It would be best for you to keep in mind that the number of items at the same level is the same within the parent level at least.

D Tree

A tree structure is familiar to linguists as a model in XML or other marked-up language data. In this model, a path is key information as an accessor. A tree data model can be rearranged into a set of records, regular or interlinear table, and a set of ordered pairs. This adaptability is welcome in computer science, then is used elsewhere. There are many applications to support XML data. However, on the other hand, you have to write source code or instructions by yourself. A tree relation or structure is flexible in a model, and you enjoy the flexibility by your own risk.

E Graph

A graph structure is a simple but powerful data model, which is the base of a variety of dynamic models such as automata, petri net or other networks. Hyper-text data is on a kind of a graph structure. A graph model is the only choice if you use time-based information. However, a graph structure is not suitable for text-based data models because of its flexibility. Note that a so-called stand-off style XML data format is not a tree model, but a hybrid model based on this graph model with name relations.

(6) Practice

In order to understand the essence of data models, it is the best way to handle it by your own hands.

i. handling of delimiters

A delimiter is a milestone symbol to be a target for distilling data units. Delimiter handling is a basic skill of text-based data handling. As well-known models, there are CSV data, data with tags such as RUNOFF or TeX, markup phases, and so on. In many programming languages a function to handle delimiters are called a string splitter.

ii. handling of XML data

When you use XML data in programming languages, you have to chose a functional interface from among many such as DOM, XSLT, XPath, and so on. Unfortunately this task is not so easy. Therefore, you select one or more functional interfaces and do the practice with it.

iii. analyzing data output from applications

ELAN and FLEx are using XML data models, but the actual structure of output data is in a stand-off style, which is based on an idea proposed by many international standard developers such as ISO. This data model is new and difficult to handle in a programming language. Thus, in order to handle the output data by these applications, it would be best for you to learn the nature of them.

iv. analyzing data needed for yourself

As is often the case in international activities, these proposed standards have been made in a top-down style without paying much attention to needs from the actual users. Or at least, a general format is made for a kind of virtual or abstract model, but the model you want is for your own data. The model proposed by the developers is like a Lang by de Saussure, not like parole as something reflecting personal activities. Therefore, you have to analyze your own data to mach with the format output by ready-made applications in order to ensure a life-long usage of the data. It is best for you to establish a data model you wish to use.

(7) Plan of Assignation

Ideally you have ability to convert your language data into another data formats when you change applications or a phase of your research processes. However, if you use adopt complex practical relations for your data format, you are required to be equipped for high-level programming. In many cases, you get along with computer people such as computer scientist or engineers. In this case, there are at least four obstacles; requirement analysis, a data model you want, assignation of roles in making data conversion, and the two-culture(sciences and humanities). In order to overcome them and to aim for data conversion with computer people, you should learn requirement analysis, data models, and characteristics of your language data. An advice for you to do it is to know your own data and requirements deeply and inform computer people of them clearly. Making a list of your requirements is a good preparation.

(8) Summary

Language documentation is probably not a main task for your activities as linguists. Language documentation is meta-research of linguistics; a study of language data or recording language data. However, language data you have is not only for your own research but also for human beings as common heritage of humanity. Thus, you are obliged to set your language data in a state that other people can handle without consulting you about it. You do not have to publish it or make it open for other researchers to access. The point is that you keep it arranged in a theory-based approach that is introduced in this course. The value of language data you have is permanent. In this sense, language documentation is one of essential knowledge for linguists who glean language data in a field.