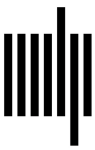
The Open Handbook of Linguistic Data Management

Edited By: Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister

Citation:

The Open Handbook of Linguistic Data Management
Edited By: Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller, Lauren B. Collister
DOI: 10.7551/mitpress/12200.001.0001
ISBN (electronic): 9780262366076
Publisher: The MIT Press
Published: 2022



The MIT Press

54 Managing Data for Descriptive Morphosemantics of Six Language Varieties

Malin Petzell and Caspar Jordan

1 Introduction

This case study evaluates methods of data processing used in a linguistic research project on the semantics of verbal morphology in Bantu languages.[1] The project involves language data collected during fieldwork on six language varieties, all of which are under-described. Early on it became clear that combining a relatively complex research subject with a multitude of languages, together with the fact that there is little previous information on them, would lead to some specific requirements on managing the language data, and on the way the researchers could interact with them. After having dismissed traditional approaches such as detailed manual interlinearization and computational corpora for reasons to be explained herein, we experimented with two different methods. The first involved a qualitative data analysis software and the second a fairly uncomplicated database setup. The software used for data processing included Toolbox (https://software.sil.org/toolbox/), NVivo (https://www.qsrinternational.com/nvivo/home), and a combination of Microsoft Excel 2016 (http://products.office.com/excel/) and OpenRefine (http://openrefine.org); the latter being the one we opted for in the end.

This chapter will start by briefly describing the research project, the requirements we had for the data management and the data collection. Then we will mention the different methods we dismissed and examine the two we tried including the work of coding, preparing for filtering, and tidying up the language data. After a note on metadata, we will evaluate and explain why we opted for the more straightforward database setup in the end.

2 The project and its background

The purpose of the research project, "The semantics of verbal morphology in central Tanzanian Bantu languages: a comparative study," is to describe and analyze the semantic construal of (universal) tense, aspect, and mood (TAM) notions and their grammatical encoding on the verb in the East Ruvu Bantu language varieties (Kagulu, Kami, Kwere, Kutu, Luguru, and Zaramo). These varieties show a significant degree of diversity in terms of their TAM systems, despite an otherwise close relationship (Petzell & Hammarström 2013). In the project, we examine the TAM systems of these languages by studying the verb forms, their basic meanings, extended functions and distribution, and how the TAM markers interact with the lexical semantics (aspectual classification) of the verb stem. To do this we needed to gather language data involving different verbs in a number of different frames as well as single sentences. Data collection was carried out by the principle investigator, Malin Petzell, in the Morogoro region in central Tanzania. Because of the complexity of the language data, some project funds were used to hire a dedicated data manager, Caspar Jordan, to work on structuring the data and making them accessible.

The Bantu language varieties in the East Ruvu group are under-described in general, and there is little information about their grammatical structure and TAM systems. There are three comparative analyses (Petzell 2012; Petzell & Hammarström 2013; Bar-el & Petzell, forthcoming), and there are published grammatical descriptions in three of the language varieties in which TAM and the verb are described, namely in Luguru (the dated Seidel 1898; the somewhat newer Mkude 1974), Kagulu (Petzell 2008), and Kami (Petzell & Aunio 2019). For the other language varieties, only outdated or short linguistic documentations can be found, such as an account of Kami written more than a hundred years ago (Velten 1900). Additionally, a comparison of TAM in Kami, Kagulu, and Luguru is being produced (Petzell & Edelsten, in prep.).

As mentioned, the aim of the research project was to answer questions on the semantics of TAM and their interaction with the verb stem. How this was done will

be explored herein, starting with the requirements we had on the data from the six language varieties.

3 Requirements on the data

First of all, TAM is a rather broad field covering all sorts of semantic and morphological intricacies, implying that an extensive amount of language data is needed to map TAM in general. Covering six language varieties further expands the need for data. Furthermore, the comparative nature of the project entails that each language can be studied from a typological angle, rather than forming the basis for an in-depth analysis of TAM in a single language. The very fact that the language varieties are closely related is the motivation for studying them together. Variations within and between languages may give clues that lead to further insights into the whole system, but this requires comparisons of morphemes and semantics between language varieties; in other words, the data need to be comparable. The complexity of the subject (TAM) and the need for comparability together give rise to a central requirement of the data management for this project: the researcher needs to be able to perform advanced filtering and search operations on the data, using categories such as the semantics of the sentence, interesting morphemes, specific language(s) as well as language user(s) (to be able to identify idiosyncrasies). For instance, the project needed a tool that enabled searches for a specific morpheme in a specific semantic frame, for example, in the vicinity of a temporal adverbial, or where a phrase in the affirmative carries the habitual marker but is not tagged for past.

3.1 (Un)Structured data and data collection

At the beginning of the project it was not clear to what level the data we included would be structured (i.e., comparable). In the collected data there were both stories and elicited sentences; the former were assumed to be less predictably structured than the latter. By *unstructured data,* we mean data that do not follow a predefined data model, such as a relational database. To a linguist, describing text as unstructured may seem counterintuitive, but in a context where computers are used to sift through data, the quantity of which makes a close reading less feasible, it makes sense, because the methods of searching for and retrieving information differ between structured and unstructured data. In this specific case, both the stories and the elicited sentences may be viewed as unstructured, even though the sentences were elicited to be comparable, and thus to contain more or less the same semantic structure, across languages. In the end, we decided not to include the stories in the data set, as they were not semantically comparable enough to benefit from being coded along the same principles as the elicited sentences. The stories will still be used in the final grammatical analysis, although not coded and processed in the same way as the single sentences in the database.

It should also be mentioned that the coding functions primarily as a facilitator for searching and filtering and does not constitute a comprehensive analysis on its own, even though we do consider coding to be an analytical process, which naturally involves a certain amount of interpretation.

The language data were collected during field trips to central Tanzania in 2016 and 2017 (for more on data collection, see Good, chapter 3, this volume). The elicited sentences were collected in the following way: language users were given questionnaires to take home consisting of a set of sentences in Swahili, which is their second language, to be translated into their respective first languages. The language users and the principal investigator would then meet for an elicitation session where they would discuss the sentences and their translation to make sure that the Swahili was understood as intended. This is a necessary step, given that the sentences were not situated in a broader context. There was also an English translation of every sentence to further clarify the intended meaning. Most language users wrote their translations by hand and these were later typed on a computer and put into Microsoft Word tables. A few language users wrote their translations directly into Word tables. In addition to the source sentences and translated sentences, the tables also contain an index number (sometimes a letter as well, e.g., 126A) and, in some cases, comments or alternative translations by the language users. These index numbers combined with the language user abbreviation and the questionnaire number subsequently made up the unique *identifier*, as described in the metadata discussion (section 6). For storage, we decided to use our university's cloud storage solution, GUbox (https://www.box.com), given that we wanted to keep our emerging data files safe and accessible for both Petzell and Jordan. This gave us the safety of automated backup and version handling, while allowing both of us easy access to the files at all times.

The language data and metadata will be archived with the Swedish National Data Service (www.snd.gu.se) once the project is finished. The Swedish National Data Service will keep the data safe and accessible for a foreseeable future (for a discussion about archiving, see Andreassen, chapter 7, this volume).

4 Dismissed methods

Had the project focused on one of the major languages of the world, with documented texts and other written data, it would have been relatively easy to build a corpus suitable for computational analysis that could then be queried using a corpus querying tool such as Korp (Borin, Forsberg, & Roxendal 2012). However, we decided not to do that for several reasons: (1) Corpus building is greatly supported by the existence of language materials such as grammars and lexica, but for our languages, these do not generally exist. (2) A discussion with the Korp development team revealed that Korp (and several other corpus tools) does not support multilinguality very well. Because we are working with six languages, we would thus have had to go through six separate corpus-building processes, still ending up with a setup that would not have been ideal for comparative research. (3) Corpora tend to be large, containing millions or hundreds of millions of words. Though this does not seem to be necessary for a corpus to be useful (cf. Ross 2018), it would not be worth the endeavor in this project to build a relatively minute corpus.

The traditional alternative to building a computational corpus would be to build a corpus in Toolbox or some other tool for linguistic annotation that requires meticulous manual annotation (interlinearization) by the researcher—a laborious process. Having built Toolbox corpora for two of the language varieties (Kami and Kagulu) for previous research projects, it was decided that the level of detailed annotations that Toolbox and similar programs offer was not necessary for our purposes in the current project. In addition, Toolbox does not accommodate comparative work on several languages very well, as every language would have to be put in its own Toolbox project, making it impossible to search simultaneously over multiple languages. The same goes for other relevant software such as Fieldworks (FLEx; http://software.sil.org/fieldworks).

Seeing that neither computational corpora nor traditional annotation-based work seemed to be the solution, we searched for a system and/or workflow and/or data structure that would allow us to smoothly code a fairly large amount of textual data and that would allow complex filtering and search operations. Section 5 discusses the two methods we tested, namely NVivo and OpenRefine.

5 Two tested methods

As there was no obvious best way to approach the data-related processes in this project and as we did not have any methodological role model, we did a fair bit of experimenting before trying NVivo. NVivo is a Computer-Assisted Qualitative Data Analysis Software (CAQDAS) application designed for qualitative analysis of text, audio, video, and other kinds of data. The NVivo program gives the user the possibility of selecting parts of the textual data and assigning to them one or more user-defined codes called *nodes*. These nodes can then be used to filter through the data to find elements relevant to a specific theme or category, but also to build a hierarchy of nodes or to relate nodes to each other in other ways. The right half of figure 54.1 shows a list of parts of text assigned to the node "PFV" (perfective aspect). The left half shows the hierarchy of nodes, where PFV is one of several nodes grouped together under the top node "TAM English." NVivo also provides visualization tools that support the user in finding structure in the data.

After discarding NVivo for reasons discussed in section 5.1, we moved on to trying a second method, which involved using a combination of Microsoft Excel and OpenRefine to build a fairly straightforward database. This approach ended up being our solution of choice. OpenRefine is a tool that allows its users to perform a broad selection of data-cleaning operations on tabular data, both numerical and textual. It supports regular expressions (see Han, chapter 6, this volume) and has its own syntax, called GREL (General Refine Expression Language), which can express complex processing tasks. Together, these features make it possible to perform complicated text processing operations, both searching and transforming (disorganized) data into useful, well-organized tables. OpenRefine works well with Excel spreadsheets, which turned out to be advantageous because Excel has some functions that are missing from OpenRefine. A combination of the two provided a powerful tool for cleaning and indexing untidy textual data.

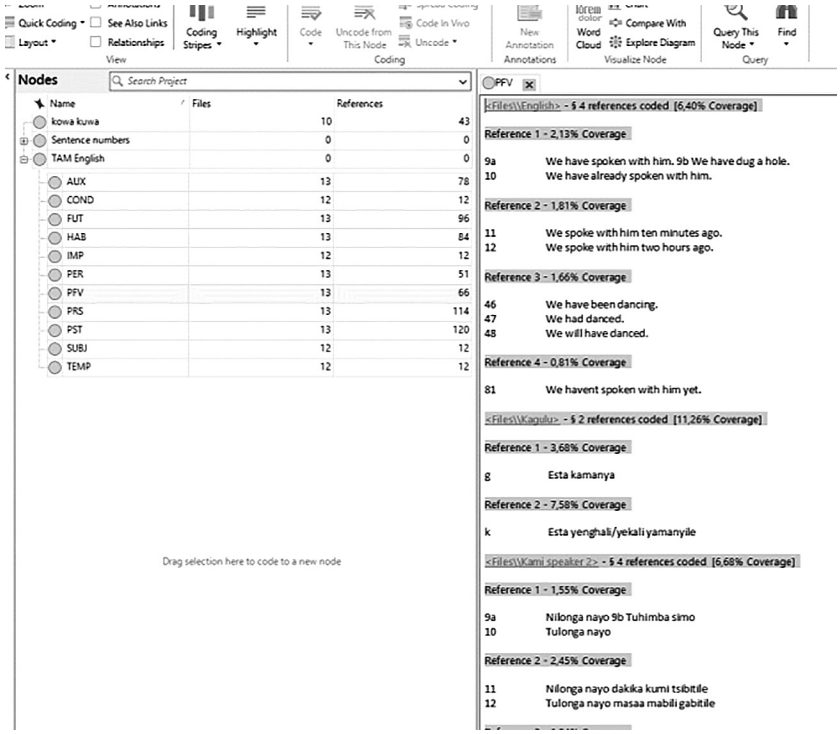


Figure 54.1
Screenshot from NVivo 12.

5.1 The first approach—coding in NVivo

Seeing that we wanted to describe, analyze, and compare the TAM markers in these six language varieties, we needed a way to compile concordances both for specific morphemes and for specific semantics across six language varieties. Additionally, it would also be helpful to be able to search for temporal adverbials (as a string of characters) or for specific realizations of morphemes. This multitude of categorizations, together with the fact that our data are of a "qualitative nature,"[2] led us to believe that it would be to our advantage to use a piece of CAQDAS software, and because there was some experience in our department with using NVivo, we opted for that. The first nodes (i.e., codes) we decided to use were typical TAM categories such as past, present, and future tense and perfective, habitual, and persistive aspects. The coding for these categories was mostly based on the English-language elicitation sentences. Cues used were the tense and aspect of the English sentences and adverbs denoting temporal and/or aspectual semantics. For two categories, Swahili was used to code for the Swahili morphemes *-me-* (perfect and/or perfective) and *-li-* (past), because the distinction between the two is not always overtly present in English. A document explaining the coding procedure for each code was created and stored together with the data.

It very quickly became clear that NVivo was not ideal for the coding necessary in this project. One reason for this is that we wanted the same coding for the same sentence in the target languages, and we wanted to code the sentence in the six language varieties at the same time—something that NVivo only allows using a workaround. By coding every sentence to a node for its index number, it becomes possible to write a "coding query" to get a list of all sentences with a specific number. This list could then be coded to a TAM node. For example, if sentence 126A was in the present tense, a coding query for sentence number "126A" would give a list of all the translations of sentence number 126A in every one of the six language varieties. This list could then easily be coded to the node "PRS" (present tense). This method worked, but required a lot of additional data cleaning, such as creating lines for every case of missing data, that is, where a language user had not given a particular sentence, merging two lines when the language user offered two alternative translations, and other similar tasks. This extra work was necessary because coding sentences to their index number automatically required each sentence to be exactly one line, as the coding was based on the number of the line.

Another reason for abandoning NVivo in our project was the difficulty of coding on the morpheme level instead of on the word or even sentence level. A sentence would

usually be coded to more than one node (e.g., past tense, persistive aspect, and the Swahili morpheme *-li-*), but in the list of results only one coding could be seen at a time.

What is more, the search function in NVivo was insufficient for our needs. In a search, the result list would not show the sentence surrounding the word that matched the search criteria, nor the translation, which was in another file ("source" in NVivo lingo). This rendered a smooth analysis impossible because it required switching between the different tabs. Another problem was that searching in NVivo does not allow truncation at the beginning of search expressions, making it impossible to search for morphemes in the middle of words, which is a serious shortcoming for Bantu linguistics given that Bantu languages are agglutinative and that Bantu words usually are concatenations of multiple morphemes. Finally, the program inexplicably crashed on a few occasions.

When we began coding in NVivo, we realized that we would get a better overview of the coding and also be able to work faster if we first collected our coding in a spreadsheet. Therefore, we created a spreadsheet with one row per elicitation sentence and one column per code, which was then used to code every single elicitation sentence by marking the appropriate cell. This left us with a spreadsheet of all the coding, which would soon turn out to be highly useful.

5.2 The second approach—coding in OpenRefine combined with Excel

After realizing that NVivo was not ideal for our purposes, it was practical to try using the spreadsheet as the primary tool for the entire data management, including the first steps of analysis. To be able to do these first steps of analysis, however, we needed to be able to filter out lists of sentences based on various combinations of coding. Given that we did not consider that Excel alone provided us with the search and filtering options we required, we then turned to OpenRefine.

The codes were transferred from the sheet with the elicitation sentences (a few hundred) to another sheet containing all the translated sentences (just over 8,700) using the Excel function =VLOOKUP, and matching by sentence ID, so that every translation of, for instance, sentence 126A received the same coding. Excel was also useful in filling in blank rows in the coding columns so that every cell in a column contained a 1 or 0 value rather than a 1 value or no value. In this way, it became possible to filter not only for positive coding but also for negative coding, which is a very useful functionality that allows for double-checking.

As mentioned, our coding thus far had been based on the English language, and in one case Swahili. As the work progressed, we needed to extend our methods of coding to include searching for a morpheme (string of characters) that only existed in the target languages and that did not necessarily translate consistently into English or Swahili. One such morpheme may appear as *-tsa-* or as the allomorph *-tso-* and is a future marker. We used OpenRefine to create a code for this by using the regular expression *.ts[ao]\B* to filter through all the sentences in the six language varieties examined. This generated a list with all the sentences where the strings *tsa* or *tso* appear within a word, that is, not word-initially and not word-finally (*tsa* in the beginning of a word is a different morpheme altogether [see section 5.3] and *tsa* at the end is the verb "come"). Based on this list, we created a new coding column by means of the GREL expression *if(or(contains(value, "tsa"), contains(value, "tso")), "tsa tso", "")*. This generated a column containing the value "tsa tso" for all sentences in the list created with the regular expression. All other sentences had no value in that column. In another step, Excel was used to replace the coding with 1 for "tsa tso" and 0 for no value, because, by then, we realized that it was favorable to have a code for no value as well. The gain here was clear: creating the coding by means of a regular expression may give some faulty coding but it saves the researcher the trouble of going through all sentences by hand. The coding was verified by spot checks but also by cross-referencing with the coding for non-future tense, as *tsa/tso* was not expected to turn up in sentences that are, for instance, in the past tense. What is more, searching for sentences in the future but excluding the *tsa/tso* future marking generated a small number of sentences where there were interesting morphological realizations of the future tense in the target languages.

Because we worked in Excel spreadsheets, we had to take precautions always to work in copies of the data set when changing anything, to avoid the danger of involuntarily changing the original data. For a while, we considered using a SQLite database that offers stricter content control, in other words, it makes it harder to involuntarily change the original data files. However, SQLite does not come with support for regular expressions, and finding

information on how to add such support was harder than expected; consequently, the idea was discarded.

Having decided on the software, we then proceeded to perform advanced filtering operations.

5.3 Filtering in OpenRefine

In OpenRefine, the filtering function (i.e., search) would always display the whole sentence, because the results list is row based and every sentence has its own row. In addition to displaying the context of a word, this also means that all the other coding for that sentence is visible as well, and even the translation in a separate column. One drawback here is that the coding can only be done on sentence level, not word level, but in this specific case, sentence level was sufficient as it was easy to identify the relevant morphemes within the sentences.

There are two main filtering options that we used in OpenRefine: text facet and text filter. The *text facet function* provides a list of all the unique values in a column, and by clicking on one of the values, all the rows that have another value will be hidden. It is also possible to reverse the effect so that all rows except the ones with the selected value are shown. As mentioned, generating a list where certain codes or strings of characters (such as a morpheme) are *not* present turned out to be decidedly constructive. The *text filter function* allowed us to search for a string of characters within a column, displaying all the rows that contained the string. This function understands regular expressions, which means that we could do very complex text string searches. An example is the regular expression *\b(tsa|dza|za|nz'a)*, which returns all rows where any of the strings *dza, tsa, za* or *nz'a*[3] is found word-initially or as an independent word. This way of searching and filtering formed the basis for an article about the unique particle *tsa*[4] that has no equivalent in English or Swahili (Petzell 2020) and that would have been hard to analyze otherwise. A further filtering operation performed on the particle *tsa* was to combine it with the code for the Swahili perfect *-me-*, which should not find any results because the particle uncommonly co-occurs with a perfect reading—and indeed, this filter did not generate any hits.

A final example of filtering is when we wanted to find out how the future perfect is expressed in these languages. We filtered for the code "FUT" (for future tense) combined with the code for the Swahili perfect *-me-*, which generated a list of thirty-seven sentences; see figure 54.2.

5.4 Tidying up the data

The Excel–OpenRefine method required a tidier data set than NVivo would as both Excel and OpenRefine give advantageous filtering functionality provided that the information is well organized. Sentence numbers, translated sentences, and original Swahili and English sentences all needed to be placed in separate columns, while language users' comments and the researcher's comments were moved out of the spreadsheet and stored elsewhere. Fortunately, OpenRefine provides a range of tools to solve these kinds of problems, for example, to standardize sentence numbers that were sometimes found in a column of their own, sometimes in front of the sentence itself, sometimes demarcated by a full stop and

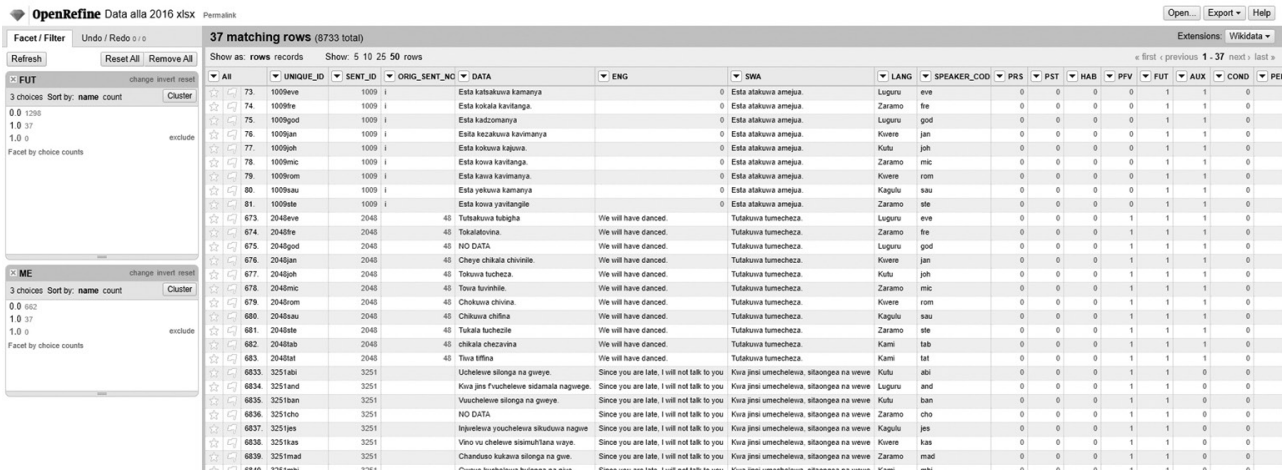


Figure 54.2
Screenshot from OpenRefine.

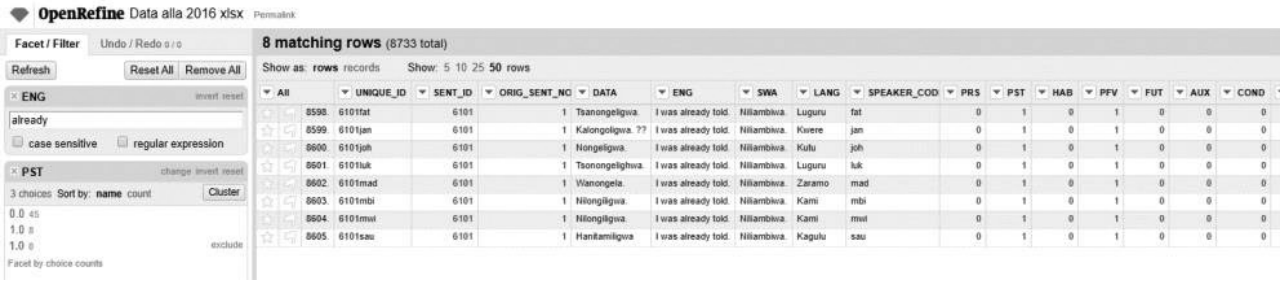


Figure 54.3
OpenRefine filters and the results of the filtering.

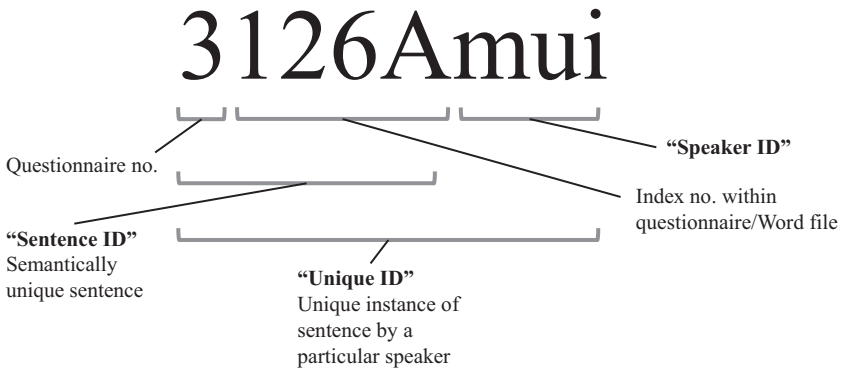


Figure 54.4
This figure shows an ID that refers to a specific translation of an elicitation sentence (sentence 3126A) by a specific language user (mui). By removing the first number, the remaining numbers (126A) point to the exact location of the data in the original questionnaire. A few sentence numbers are followed by a capital letter, which is part of the sentence number. The questionnaire number refers not to the filled-in questionnaire but to the template questionnaire. This is because data from several different questionnaires were used. The questionnaire number is arbitrary.

sometimes not, and so on. Once this standardization was completed, it became possible to filter the 8,700+ sentences to get all instances of one specific sentence, something that otherwise would only have been possible by manually going through all the Word files separately. Figure 54.3 shows two filters, a text filter for the word *already* and a text facet where the user has chosen the value 1 for PST, applied to the data set of 8,733 sentences. Together the filters result in a list of 8 sentences which are in the past tense (value 1 for the PST column) and where the English elicitation sentence contains the word *already*.

Keeping Swahili, English, and the target-language data in separate columns allowed us to search efficiently for morphemes and words in a specific language. Other important columns included sentence ID, unique ID (see section 6), language user ID, language ID, and columns for all the codes we used.

6 Metadata

Each language user was coded with a three-letter abbreviation, and the *metadata* (i.e., data on the data; see Good, chapter 3, this volume; Kung, chapter 8, this volume) on the language user, such as their place of birth, age, level of education, occupation, family situation, and so forth, were kept in a separate file. The metadata also noted when, where, how, and by whom the data were collected.

Excel was used to create unique identifiers for every instance of a sentence by combining the sentence ID (derived from the index numbers found in the Microsoft Word tables preceded by the number of the questionnaire) and the three-letter code for the language user. In this way, it became possible to refer to every single sentence in the collection (see figure 54.4), which can also be done when citing the data in various publications. It also made it possible to cross-reference every sentence with the language

user's metadata to check for variables such as education, parents' background, spouse from another language group, and such, as well as with the original Microsoft Word file where the language users' and the principal investigator's comments and alternative translations are kept. This can also be useful if a language user were to wish to be withdrawn from the study, as the data supplied by that person would be easy to locate and delete.

7 Conclusions

We have shared our experience building a small database containing data from six under-described language varieties using OpenRefine. The advantages of the software were the possibility to filter using regular expressions, the easy tidying of disorganized data, and the combined searches (filters) using codes and strings of characters. The software facilitated the analysis and eventually helped to answer research questions about the semantics of TAM and the verb in these language varieties. It would have been problematic to get an overview of the diverse and rather substantial material without the coding, filtering, and search functionality that OpenRefine or similar software offers. Had we had more time and resources, we would have liked to fully annotate all the language data for future purposes in software such as FLEx or Toolbox, but for this project it was neither necessary nor feasible.

NVivo did not serve our purposes, but for even more unstructured data it is most likely a useful tool. The two main reasons we ended up not using NVivo were that the language translations ended up in different tabs, making it difficult to compare between languages, and that cross-referencing and searching generated lists that did not contain the amount of information we needed.

Given the quantity of our textual material, our research objectives, and the resources at hand, we are very satisfied with the results these methods of data processing generated. Consequently, we recommend using a combination of a spreadsheet and a data cleaning and filtering software such as OpenRefine.

Notes

1. We are grateful to two anonymous reviewers for valuable comments and to Riksbankens Jubileumsfond for providing the funding for the research project.

2. It may be argued that qualitative and quantitative are features of the analysis method rather than of the data. However, text is usually thought of as qualitative data in the data management discourse.

3. The target languages are not standardized; thus there is considerable variation in pronunciation and spelling of the same morpheme.

4. The particle encodes some type of shared knowledge or shared reference and conveys different meanings similar to "at a specific time," a locative reference "at that place," or even "for a reason."

References

Bar-el, Leora, and Malin Petzell. Forthcoming. (Im)perfectivity and actionality in East Ruvu Bantu. *Language Typology and Universals* 74 (3).

Borin, Lars, Markus Forsberg, and Johan Roxendal. 2012. Korp—the corpus infrastructure of Språkbanken. In *Proceedings of LREC 2012*, 474–478. Istanbul: ELRA.

Mkude, Daniel J. 1974. A study of Kiluguru syntax with special reference to the transformational history of sentences with permuted subject and object. PhD thesis, University of London.

Petzell, Malin. 2008. *The Kagulu Language of Tanzania: Grammar, Texts and Vocabulary*. Cologne, Germany: Rüdiger Köppe Verlag.

Petzell, Malin. 2012. The under-described languages of Morogoro: A sociolinguistic survey. *South African Journal of African Languages* 32 (1): 17–26.

Petzell, Malin. 2020. An analysis of the verbal marker *tsa* in Luguru. In *The Semantics of Verbal Morphology in Under-Described Languages*, ed. M. Petzell, L. Bar-el, and L. Aunio. Special issue, *Studia Orientalia Electronica* 8 (3): 119–133.

Petzell, Malin, and Lotta Aunio. 2019. Kami G36. In *The Bantu Languages*, 2nd ed., ed. M. van de Velde, K. Bostoen, D. Nurse, and G. Philippson, 563–590. London: Routledge.

Petzell, Malin, and Peter Edelsten. In preparation. TAM marking in Bantu languages of Morogoro region, Tanzania. Unpublished manuscript.

Petzell, Malin, and Harald Hammarström. 2013. Grammatical and lexical comparison of the Greater Ruvu Bantu Languages. *Nordic Journal of African Studies* 22 (3): 129–157.

Ross, Daniel. 2018. Small corpora and low-frequency phenomena: *try and* beyond contemporary, standard English. *Corpus* 18. https://journals.openedition.org/corpus/3574.

Seidel, August. 1898. Grundriss der Wa-Ruguru-Sprache. In *Die mittleren Hochländer des nördlichen Deutsch-Ostafrika*, ed. C. Waldeman Werther, 436–455. Berlin: Verlag von Hermann Paetel.

Velten, Carl. 1900. Kikami, die Sprache der Wakami in Deutsch-Ostafrika. *Mitteilungen des Seminars für orientalische Sprachen* 3:1–56.