Verb inflection in the SAND

Erik Tjong Kim Sang
Meertens Institute Amsterdam
erik.tjong.kim.sang@meertens.knaw.nl

October 21, 2016

1 Introduction

In their article *Variation in verbal inflection in Dutch dialects*, Hans Bennis and Alies MacLean [4] study the variation of verb inflection in Dutch dialects based on data from the SAND, the Syntactic Atlas for Dutch dialects [2]. The data used in the study, can only be extracted from the resource by carefully selecting certain parts while following the selection steps described in the paper. We trace the footsteps of the study and describe all the steps necessary to reproduce the data of the study by Bennis and MacLean. However, ten years after their study, we find that we cannot reproduce their data from the SAND.

2 Data source: Syntactic Atlas for Dutch dialects

The Syntactic Atlas for Dutch dialects (SAND) is a data set containing syntactic features of 267 locations in the Dutch language area (The Netherlands, Flanders and French Flanders). The data are available on printed maps [2, 3], on an interactive website [1] and in a relational database [6]. We use the relational database because it offers the largest flexibility for selecting and combining various parts of the data.

Bennis and MacLean [4] state that they have used a part of the SAND based on the following criteria:

1. the data concerns the Dutch verb *leven* (*to live*)
2. only the finite verb form is used
3. only verb forms preceded by a subject are used
4. ignore locations with an incomplete paradigm for the target verb[1].
5. phonological differences have been ignored, in particular the difference between the affixes *-e*, *-en*, *-m* and *-n*

	-(e)d -(e)n -(e)s -(e)st -(e)t -de -den -e -je	suffixes
	1 2 3	person
	?	unknown
	conj	conjunctive
	eind	end position
	eind1	most important verb in sentence
	eind2	second most important verb in sentence
	eind3	third most important verb in sentence
-	imp	imperative
-	inf	infinitive
-	inv	inverse order
	onacc	unaccusative verb
	onerg	iunergative verb
	ov	other
	pl	plural
	pro-drop	pro-drop
	recht	straight order
	s	singular
	s-b	polite version of singular
	trans	transitive verb
+	tt	present tense
	v-cl	verbal cluster
	v3	pronoun in third position
-	volt+prefix	past participle with prefix
-	volt-prefix	past participle without prefix
-	vt	past tense

Table 1: The 36 part-of-speech attributes associated with the verb *leven*. These can be used for selecting the verb forms of interest to this study. A plus or minus sign before the attributs indicates whether we want to select (+) or reject (-) words of which the part-of-speech tag contains this attribute.

All the dialect data in the SAND has been annotated with lemma information and syntactic information. The selection method of Bennis and MacLean uses this information. First, there is the selection of all forms of the verb *leven* which is possible by selecting all words with the lemma *leven* (value 2 in the column `lemma_id` of the database table `sandtag_lemma`).

For the criteria 2, 3 and 4, we need to inspect the part-of-speech tags attached to each word. These contain several attributes, for example a part-of-speech tag associated with the word *leeft* could be V(-(e)t,eind,tt,3,s,onerg). This tag contains six attributes. The

[1] A complete paradigm should include six verb forms first person singular and plural, second person singular and plural, and third person singular and plural.

main form	phological variants
leef	leew leeuw lef lèf levv
leefs	lefs
leeft	lebt left lèft lift leevt levt levvt leewt leeuwt
leven	lebben lebe leben lebm lebn leebm leebn leefn leeven leevm
	leevn leewe leewn lefe leve lève levm levn levve lewe

Table 2: Presumed phonological variants of the verb forms *leef*, *leefs*, *leeft* and *leven*.

part-of-speech tags associated with the verb (V) *leven* contain 36 different attributes which are listed in Table 1.

For selecting the finite forms of the verb, we require the verbs to be in present tense (attribute *tt*) and remove all participles (*volt+prefix* and *volt-prefix*), past tense (*vt*), imperative (*imp*) and infinitive (*inf*). For the subject-verb order, we remove all verbs with attribute inverted (*inv*). Bennis and MacLean ignored 13 incomplete paradigms. We found 71(!) locations which were missing at least one verb form. Finally, we converted 35 presumed phonological variants to equivalent main forms: 5 for *leef*, 1 for *leefs*, 9 for *leeft* and 20 for *leven*, see Table 2.

3 Results

We extracted the paradigms for the verb *leven* from the relational database of the SAND with the methods described in the previous section. Next, we compared the data with the paper by Bennis and MacLean [4]. We found three differences.

First, we found that 71 of the 267 locations did not contain a complete paradigm, where a paradigm was considered complete when it contained six verb forms: first, second and third person in both singular and plural form. Bennis and MacLean had found only 13 locations with an incomplete paradigm. We will discuss one of these unsuspected incomplete locations (Lemmer) with the next difference.

Next, we compared the paradigms associated with the seven locations mentioned in examples (2) and (6) of the paper by Bennis and MacLean with our data. We found that for three locations (Bree, Hasselt and Leuven) the data were the same but that for the four other our data were different than from those mentioned in the paper, see Table 3.

Two of the differences (Lokeren third person singular and Beekbergen first person plural) are caused by the presence of multiple values in our data. We have looked up the part-of-speech tags of the two alternative forms, V(recht,tt,3,s,onerg) and V(-e,eind,tt,1,pl,onerg) and see no reason for excluding them from the data (part-of-speech tag attributes are explained in Table 1).

For two locations, there are different values in our data: Beekbergen third person plural and Diksmuide second person plural. For Beekbergen, the form mentioned in the paper is only present in the inverted subject-verb usage. For Diksmuide we could not find the form

Beekbergen		Diksmuide		Lemmer		Lokeren	
leef	leef	leven	leven	leef	-	leef	leef
leef	leef	leeft	leeft	leefst	-	leeft	leeft
leef	leef	leeft	leeft	leeft	leeft	leeft	leef/leeft
leef	leef/leven	leven	leven	leven	leven	leven	leven
leef	leef	leven	leeft	leven	-	leeft	leeft
leef	leeft	leven	leven	leven	leven	leven	leven

Table 3: Paradigms for the verb *leven* for the four locations mentioned in the paper by Bennis and MacLean (left in the columns) which are different from our data (right in the columns). Different values in our data have been colored red.

mentioned in the paper associated with the second person plural.

Finally there is the location Lemmer for which the paper mentions six verb forms while we found only three in the data. For the first person singular, the form mentioned in the paper is only available in the inverted subject-verb usage. The second person singular form is available in the polite form but that one (*leven*) is different from the one listed in the paper (*leef*). Then there is the second person plural form which is available in the inverted subject-verb usage but that one (*leef*) is different from the one shown in the paper (*leven*).

To summarize: while they are mostly similar to our data, the data of Bennis and MacLean contain only single forms while we found multiple alternatives in some cases, they sometimes contain different values than in our data and they sometimes contain values where we found none.

Because we find fewer and different verb forms than mentioned in the paper by Bennis and MacLean, it is no surprise that the verb form totals in our data are also different from those in the paper. Like Bennis and MacLean, we find that paradigms with three different verb forms are the most frequent (144 of 196 locations vs 147 of 253). However we find fewer locations with two different verb forms, 26 vs 86 in the data of Bennis and MacLean. The other totals are similar: 0 vs 1 for one verb form, 21 vs 19 for four verb forms and 5 vs 0 for five verb forms. Since we found fewer locations with two different verb forms, the areas on the map by Bennis and MacLean with a majority of such locations (blank squares in the south west and east) are neither visible on the map based on our data, see Figure 1.

4 Concluding remarks

We have attempted to reproduce the data used in the paper *Variation in verbal inflection in Dutch dialects* by Bennis and MacLean [4]: paradigms of the Dutch verb *leven* (*to live*) from 267 different locations in the Dutch-speaking language area. The data originated from the Syntactic Atlas of the Dutch Dialects [2]. The selection steps for the data were described reasonably well in the paper. Still we have been unable to extract the same data set. We found differences with respect to the number of locations with incomplete paradigms (71 vs

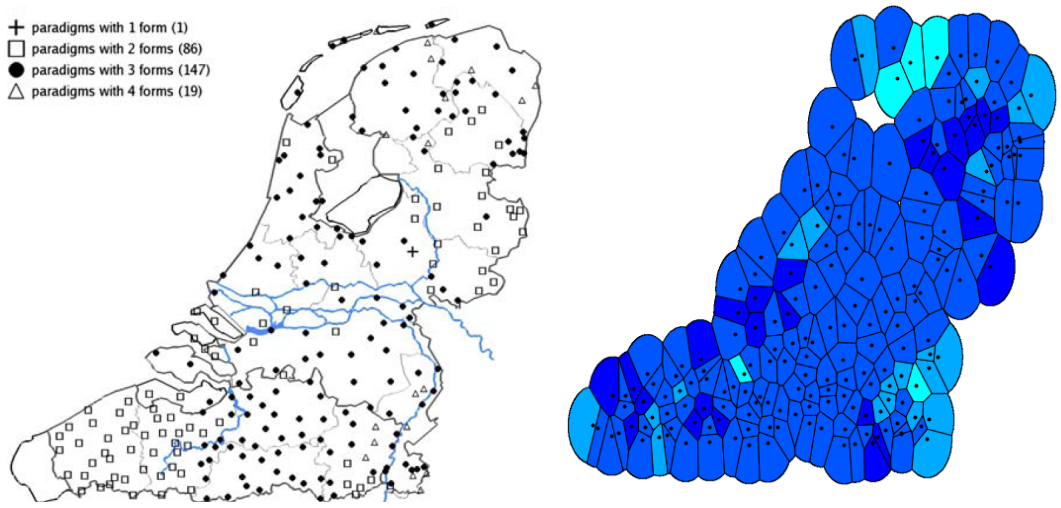

Figure 1: Maps displaying the number of forms in the paradigms of the verb *leven*. Left: the map from the paper by Bennis and MacLean. Right: a map corresponding to our data. Like Bennis and MacLean, we find that paradigms with three different verb forms are in the majority but we do not find clusters like the south west and east, both with a majority of two different verb forms (blank squares left and dark areas right).

13), the example paradigms mentioned in the paper (Table 3) and the number of different verb forms per location as shown in the maps in Figure 1.

In order to harmonize the two data sets we would have to remove some competing verb forms, change some verb forms and add verb forms where no data is present. It is unclear how this should be done.

References

[1] Sjef Barbiers. Dynamische Syntactische Atlas van de Nederlandse Dialecten (DynaSAND), 2006. http://www.meertens.knaw.nl/sand/ Accessed 26 February 2015.

[2] Sjef Barbiers, Hans Bennis, Gunther Vogelaer, Magda Devos, and Margreet van der Ham. *Syntactic Atlas of the Dutch Dialects*. Amsterdam University Press, 2005.

[3] Sjef Barbiers, Johan van de Auwera, Hans Bennis, Eefje Boef, Gunther Vogelaer, and Margreet van der Ham. *Syntactic Atlas of the Dutch Dialects*. Amsterdam University Press, 2008.

[4] Hans Bennis and Alies MacLean. Variation in verbal inflection in dutch dialects. *Morphology*, 16:291–312, 2006.

[5] Ton Goeman, Marc van Oostendorp, Piet van Reenen, Oele Koornwinder, and Boudewijn van den Berg. *Morfologische Atlas van de Nederlandse Dialacten, Deel II.* Amsterdam University Press, 2009.

[6] Jan Pieter Kunst. MYSQL SAND Database, 2013. personal communication.

[7] Georges De Schutter, Boudewijn van den Berg, Ton Goeman, and Thera de Jong. *Morfologische Atlas van de Nederlandse Dialacten, Deel I.* Amsterdam University Press, 2005.

A Extracting data from the SAND

This section describes how to extract the data referred to in this report on a machine running Mac OS X.

A.1 Installing and running XAMPP and mysql

We use the database version of the SAND and therefore we need a database program to access the data. We use `mysql` which is a part of the XAMPP package which can be downloaded from `https://www.apachefriends.org/`. Download the package and install it by following the installation instructions. Next, start XAMPP from the installation window or from a Finder window, where the program can be found in /Applications/XAMPP/manager-osx . In the program, click on *Manage Servers*, select the line *MySQL Database* and click on *Start* to start the MySQL daemon.

A.2 Loading the data in the database program MySQL

We start the database program `mysql` in a terminal window. Terminal windows can be started from a Finder window from the location /Applications/Utilities/Terminal. In the Terminal window type:

```
/Applications/XAMPP/xamppfiles/bin/mysql -u root
```

to start the database program. Note that the MySQL database daemon should be running in order for this to work (see the previous section on how to start the MySQL daemon).

In the database program you can enter commands, for example:

```
MariaDB [(none)]> select "Hello world!";
```

Do not forget the semicolon (;) at the end of the command.

We start with creating a database for the SAND data. You will only need to perform this command once:

```
MariaDB [(none)]> create database sand;
```

Next, we load the data into the database. In order to do this, you need to exit the database program:

```
MariaDB [(none)]> quit;
```

You are now back at the basic terminal level. Enter:

```
/Applications/XAMPP/xamppfiles/bin/mysql -u root sand < sand.sql
```

You should be in the same directory as the SAND data file `sand.sql` (size 236,795,232 bytes). Loading the data in the program may take more than ten minutes, please be patient and do not interrupt the process.

A.3 Working with the SAND

Now that the data from the SAND is loaded into the database program `mysql`, we can inspect the data and select parts of it. Start the program `mysql`:

```
/Applications/XAMPP/xamppfiles/bin/mysql -u root
```

Next, select the SAND database:

```
MariaDB [(none)]> use sand;
```

The SAND data is distributed over several tables. With the following command you can check which tables are present in the database:

```
MariaDB [sand]> show tables;
```

Do not forget the semicolon (;) behind the command. You can use the command `select` to examine the contents of the tables. Since the tables may contain thousands of rows, it is advisable to limit the view to a few rows, for example 3:

```
MariaDB [sand]> select * from sandtag_lemma limit 3;
```

With this command you ask for all the columns (*) of the first three rows of the table `sandtag_lemma`. This table contains all the lemmas in the SAND and links them to a number which can be used for finding information about the lemmas in other tables. For example, if you want to look up the number associated with the lemma *leven* (*to live*) you can use the command:

```
MariaDB [sand]> select * from sandtag_lemma where lemma="leven";
```

In the database program you can create your own tables by putting `create table mytable` before a selection command, for example:

```
MariaDB [sand]> create table mytable1 select * from sandtag_lemma where
lemma="leven";
```

You can inspect the contents of the new table with the select command:

```
MariaDB [sand]> select * from mytable1;
```

And when you do not need the table anymore, you can remove it:

```
MariaDB [sand]> drop table mytable1
```

A.4 Selecting the inflection forms of *leven*

In order to select the inflection forms of the verb *leven* (*to live*) as used in this report, we need six commands which combine information from seven tables of the SAND database. We start with selecting all part-of-speech tags associated with `lemma_id` value 2 (*leven*):

```
MariaDB [sand]> create table mytable1 select sandtag_toegekend.woord_id,
lemma_id,sandtag_leesbaar from sandtag_toegekend_lemma left join
sandtag_toegekend on (sandtag_toegekend_lemma.woord_id =
sandtag_toegekend.woord_id) where lemma_id=2 and not
sandtag_toegekend.woord_id is NULL;
```

If you copy and paste these command to the database program, please check if the underscores (_) have been copied as well

If you want to apply this command for another verb, find the `lemma_id` value for this verb with the command `select * from sandtag_lemma where lemma="YOURVERB";` and replace the value 2 above with the alternative value.

Next, look up the words associated with the part-of-speech tags:

```
MariaDB [sand]> create table mytable2 select mytable1.woord_id,lemma_id,
sandtag_leesbaar,interval_id,token from mytable1 left join woord on
(mytable1.woord_id = woord.woord_id);
```

Then we find the interview section associated with the word. We need this to find out the location associated with the word.

```
MariaDB [sand]> create table mytable3 select mytable2.woord_id,lemma_id,
sandtag_leesbaar,tier_id,token from mytable2 left join praat_interval
on (mytable2.interval_id = praat_interval.interval_id);
```

Next we look for the person that said the word, again to get access to the location:

```
MariaDB [sand]> create table mytable4 select woord_id,lemma_id,
sandtag_leesbaar,file_id,token from mytable3 left join praat_tier on
(mytable3.tier_id = praat_tier.tier_id);
```

With the speaker, we can get access to the interview details, among which a location id:

```
MariaDB [sand]> create table mytable5 select woord_id,lemma_id,
sandtag_leesbaar,meetpunt_id,token from mytable4 left join praat_file
on (mytable4.file_id = praat_file.file_id);
```

And finally we can find the Kloeke number of the location with the location id:

```
MariaDB [sand]> create table mytable6 select kloeke_nr,woord_id,lemma_id,
sandtag_leesbaar,token,plaatsnaam from mytable5 left join meetpunt on
(mytable5.meetpunt_id = meetpunt.meetpunt_id);
```

The result of this is a table with six columns and 8750 rows.

A.5 Converting MySQL database to Excel tables

In the previous section, we described how we can extract useful information from the SAND database in MySQL. However, we want to use this information in the program Excel. We perform three steps to transfer the data from MySQL to Excel: 1. dump the data from MySQL to disk; 2. convert the data to csv format (comma-separated values); 3. perform additional processing (counting values). The csv file produced by this process can be imported by Excel.

The table `mytable6` can be saved as a file with the program mysqldump by running the following command in a terminal window:

```
/Applications/XAMPP/xamppfiles/bin/mysqldump -u root sand mytable6
> mytable6.sql
```

(the command should be typed on a single line) Note that in order for this command to succeed, the MySQL daemon should be running. See section A.1 on how to start the daemon.

We cannot load the MySQL file directly in Excel so we convert it to the format csv (comma-separated values) which can be imported in Excel. We use the program `sql2csv`, supplied with this report, for the conversion. In a terminal window type:

```
bin/sql2csv < mytable6.sql | sort > mytable6.csv
```

The table contains separate rows for each verb form per location. We want to recreate the data of Bennis and MacLean and therefore we need a table with one row per location. For that purpose we process the data with a second program `countverbforms.sand`, which is supplied with this report:

```
bin/countverbforms.sand lèf=leef lèft=leeft lève=leven lebben=leven
lebe=leven leben=leven lebm=leven lebn=leven lebt=leeft leebm=leven
```

```
leebn=leven leefn=leven leeuw=leef leeuwt=leeft leeven=leven
leevm=leven leevn=leven leevt=leeft leew=leef leewe=leven leewn=leven
leewt=leeft lef=leef lefe=leven lefs=leefs left=leeft leve=leven
levm=leven levn=leven levt=leeft levv=leef levve=leven levvt=leeft
lewe=leven lift=leeft < mytable6.csv > sand-leven-counted.csv
```

The command takes as arguments all the phonological variants that should be considered as equal, in the format `variant=main_form`. The resulting table `sand-leven-counted.csv` contains 268 rows (267 locations and a heading row). It can be imported in Excel.

A.6 Loading data in Excel

The file `sand-leven-counted.csv` can be read by Excel. The best method to load the file in Excel is (for our Excel version 14.6.0) to open a new blank file (New Workbook) and then import the csv file in the blank file:

1. select from the top menu *File* and then from the dropdown menu *Import*
2. choose *CSV file* and *Import*
3. select the file (in this example `sand-leven-counted.csv`)
4. select *Delimited* and choose *UTF-8* at *File origin*, then press *Next*
5. deselect *Tab* and press *Next*
6. press *Finish*
7. press *OK*

While this instruction sequence works for our version of Excel, you might need different instructions for different versions of Excel. The most important step in the instruction list is the selection of the encoding (UTF-8). It is important that the correct character encoding is chosen or else the data might look different in Excel than you would expect.

B Other data selections

In the previous section, we presented all steps that are necessary to extract the data associated with the paper by Bennis and MacLean [4]: the inflections of the verb *leven* (*to live*). In this section we present the commands for generating three other data selections.

B.1 First person singular forms in the SAND

The following commands can be used for extracting all the first person singular forms of finite verbs from the SAND. We start with the database commands to be executed in MySQL. See appendices A.1 and A.2 on how to start the MySQL daemon and how to start MySQL.

```
MariaDB [(none)]> use sand;
MariaDB [sand]> drop table mytable1,mytable2,mytable3,mytable4,mytable5,
mytable6;
MariaDB [sand]> create table mytable1 select sandtag_toegekend.woord_id,
lemma_id,sandtag_leesbaar from sandtag_toegekend_lemma left join
sandtag_toegekend on (sandtag_toegekend_lemma.woord_id =
sandtag_toegekend.woord_id) where sandtag_leesbaar like "V%" and not
sandtag_toegekend.woord_id is NULL;
MariaDB [sand]> create table mytable2 select mytable1.woord_id,lemma_id,
sandtag_leesbaar,interval_id,token from mytable1 left join woord on
(mytable1.woord_id = woord.woord_id);
MariaDB [sand]> create table mytable3 select mytable2.woord_id,lemma_id,
sandtag_leesbaar,tier_id,token from mytable2 left join praat_interval
on (mytable2.interval_id = praat_interval.interval_id);
MariaDB [sand]> create table mytable4 select woord_id,lemma_id,
sandtag_leesbaar,file_id,token from mytable3 left join praat_tier on
(mytable3.tier_id = praat_tier.tier_id);
MariaDB [sand]> create table mytable5 select woord_id,lemma_id,
sandtag_leesbaar,meetpunt_id,token from mytable4 left join praat_file
on (mytable4.file_id = praat_file.file_id);
MariaDB [sand]> create table mytable6 select kloeke_nr,woord_id,lemma_id,
sandtag_leesbaar,token,plaatsnaam from mytable5 left join meetpunt on
(mytable5.meetpunt_id = meetpunt.meetpunt_id);
```

The commands are the same as those used for extracting the data for the verb *leven* in the previous section except for the underlined part selecting the specific verb in the command building table `mytable1` which has been replaced by a phrase selecting all verbs.

These commands produce a table with 6 columns and 86,909 rows. We save the table to a file by running the same command as for *leven* in a terminal window:

```
/Applications/XAMPP/xamppfiles/bin/mysqldump -u root sand mytable6
> mytable6.sql
```

(the command should be typed on a single line) The file `mytable6.sql` contains all verb forms. When we convert it to csv format, we select the part-of-speech tags that we are interested in (*tt*, *1* and *s*) and deselect inverted subject verb orders (*inv*):

```
bin/sql2csv < mytable6.sql | grep '\btt\b' | grep '\b1\b' | grep '\bs\b'
| grep -v '\binv\b' | sort > mytable6.csv
```

The result is a table of six columns and 6572 rows which can be imported in Excel.

B.2 Inflection forms of the verb *leven* from the MAND

The Morphological Atlas of the Dutch Dialects (MAND) [7, 5] is a dialect resource similar to the SAND but with a focus on morphology. It is also known as GTRP (Goeman Taeldeman Van Reenen Project), referring to the project that produced the data. The database file `mand.sql` (size 77,691,952 bytes) can be loaded in MySQL in the same way as the one of the SAND (see section A.2, it takes about half an hour). Much of the data are stored in one table so we only need one MySQL command to extract the data that we are looking for:

```
MariaDB [mand]> create table mytable1 select id,item_basic,kloeke,
transcription,transcription_vereenvoudigd,transcription_ipa
from modneddia left join item_basic on modneddia.item_id=
item_basic.item_id where modneddia.item_id=1658 or
modneddia.item_id=1659 or modneddia.item_id=1660 or
modneddia.item_id=1661 or modneddia.item_id=1662 or
modneddia.item_id=1663;
```

The verb forms are selected by specifying the required values of the column `item_id`. The verb forms associated to these values are listed in the table `item`. For example, the command `select * from item where item="zij leven";` returns the value 1663 in the column `id`.

This command creates a table with six columns and 3702 rows. The table can be saved to disk in the same way as the SAND tables, with the terminal command:

```
/Applications/XAMPP/xamppfiles/bin/mysqldump -u root mand mytable1
> mytable1.sql
```

(the command should be typed on a single line) Next we convert the sql file to csv format:

```
bin/sql2csv < mytable1.sql | sort -t, -k3 > mytable1.csv
```

And finally we combine the different verb forms on one line per location:

```
bin/countverbforms.mand < mytable1.csv > mand-leven-counted.csv
```

This command produces a table with 16 columns and 618 rows. The file `mand-1-s.csv` can be imported in Excel. It contains characters of the International Phonetic Alphabet (IPA) in the six right-most columns. These may or may not be displayed correctly in your version of Excel.

B.3 First person singular forms in the MAND

Unfortunately MAND does not contain a generic marker for the first person sigular forms of finite verbs. We tried to find out for which verbs paradigms were available by searching in the database for the phrase *wij ...* (*we ...*). Thus we found ten different verbs: *breken*, *doen*, *hebben*, *kloppen*, *komen*, *krijgen*, *leven*, *zien*, *zijn* and *zwijgen*. We could find only nine of these in singular form, of which we suspected one (*kom*) was a noun. Therefore we selected only eight verbs: *krijg* (1275), *ben* (1592), *heb* (1608), *doe* (1624), *klop* (1641), *leef* (1658), *breek* (1675) and *zwijg* (1692):

```
MariaDB [mand]> drop table mytable1;
MariaDB [mand]> create table mytable1 select id,item_basic,kloeke,
transcription,transcription_vereenvoudigd,transcription_ipa from
modneddia left join item_basic on modneddia.item_id=item_basic.item_id
where modneddia.item_id=1275 or modneddia.item_id=1592 or
modneddia.item_id=1608 or modneddia.item_id=1624 or
modneddia.item_id=1641 or modneddia.item_id=1658 or
modneddia.item_id=1675 or modneddia.item_id=1692;
```

The result is a table with six columns and 4936 rows which can be saved as a file in a terminal window:

```
/Applications/XAMPP/xamppfiles/bin/mysqldump -u root mand mytable1
> mytable1.sql
```

(the command should be typed on a single line) This file can be converted from sql format to csv format:

```
bin/sql2csv < mytable1.sql | sort -t, -k3 > mand-1-s.csv
```

The resulting file `mand-1-s.csv` can be imported in Excel.