



•
•

Celex and Gawk

Dirk Janssen

`dirkj@rz.uni-leipzig.de`

Institut für Linguistik, Universität Leipzig

Celex History

Celex is a lexical database created by the *Instituut voor Nederlandse Lexicografie* (INL) and the Max-Planck-Institute, Nijmegen. It contains data for Dutch, English, and German. All data are based on corpora, large amounts of text that was scanned in, automatically classified, and hand corrected. You never see the texts that Celex is based on due to copyright reasons. In this and some other things, it is clear that Celex was made in the '80s, when electronic texts were rare.

Accessing Celex

Celex is (was) available via the Flex interface, an interface with text windows that is rather complicated.

Aparantly, the MPI Nijmegen is currently creating a web interface to Celex, but it is unclear to me whether that can be accessed from the rest of the world already.

My preferred way to access Celex is via the Celex CDROM: This is faster, more flexible, but more complicated.

Celex Data

Each of the three lexicons come in two variants: lemma and wordform lexicon. The lemma lexicon is like a dictionary, *work* occurs twice, once as a noun, once as a verb. In the wordform lexicon, *work* occurs more times: The inflectional variants are listed too. For English, this is only *works*, *worked* and *working*, for German this would be a longer list. Of course, not all inflectional variants of a each word did actually occur in the corpora, therefore, inflectional variants were automatically added from the base form: For each verb *V*, the three forms *Ved*, *Vs*, and *Ving* are added for English.

Celex Data (2)

The same happened for lemmas: Because some quite common words happened to not be in the corpus, Celex was extended by merging it with a medium size dictionary (for Dutch, I think this was the *Van Dale Hedendaags Nederlands*).

The words that were added to the database but were not in the corpus can be recognised by a frequency entry of '0' (zero). This was a rather unfortunate choice.

Type vs. Token Frequencies

A type count counts how many different words there are in a corpus, “chocolate cake” is one type.

A token count is about how many words there are, each time the word “chocolate cake” occurs in the text, this is one token.

For a lemma lexicon, this is slightly more complicated: The words “chocolate cake” and “chocolate cakes” count as one type (inflectional variation is disregarded), and the token frequency for “chocolate cake” consists of both singular and plural usages of the word.

The total number of word tokens in the corpus can be counted in two ways: (1) By computing the sum of all tokens in the corpus, or (2) by looking at the size of the scanned corpus before some words were thrown out. The second method is better.

Celex Frequencies

Here are the size of the three Celex corpora:

	Dutch	English	German
Number of types (lemmas)	124.136	52.447	51.728
Number of types (wordforms)	381.292	160.595	365.530
Number of tokens(1)	40.2M	18.6M	5.0M
Number of tokens(2)	42M	?	6.0M

The frequency of homographs (*Messer knife* or *Mess+er measurer*) are added and then equally divided over both analyses. This is obviously wrong, as Celex guide points out: On a hand analysis of the corpus, 84 times *Messer* was *knife*, 12 were actually a last name, and 4 were hard to disambiguate.

24502\Messer\Messer\50\.\ ((mess) [v] , (er) [N|v, l]) [M

Notes on frequencies

- The corpora (esp. the German corpus) are not so large, the sample error is substantial
- The text used are written text, some are scientific, some are from newspapers. The actual (spoken) frequencies deviate from these quite a bit. Alternative: familiarity ratings as used by Schreuder and Baayen.
- The frequency zero is actually an ‘unknown’ frequency, a word with frequency zero did not appear in the corpus but it’s actual frequency of usage could easily be around 10 or so. Occasionally, a zero-frequency word might actually be quite common in everyday speech (*toilet-roll*).

Tagging

Tagging is the process of classifying the words in a corpus. Because the Celex corpora are large, most of it was done automatically. This leads to some problems, mostly with word classes and morphology:

- The adjective–adverb distinction is of course hard to make, therefore English *direct*[Adj] has a frequency of 1472, *direct*[Adv] has 0.
- The morphology is sometimes inconsistent, and sometimes absent:
 - seaweed: sea+weed
 - seaward: seaward
 - air-sea rescue: —

Phonology: DISC

Celex uses its own system of transcribing phonology, called Disc. Disc uses precisely one character for each sounds, which makes it hard to read, but easy to process by computer. Capital letters for vowels are usually short, capital letters for consonants are special variants or diphthongs, @ is schwa, special characters (ie. '=') are complex consonants.

Universit\ "atsbiblotheK\ /U-ni-vEr-zi-'t)=-bi-bli-o-tek/

with faehig\ /'f)-Ix/

Steinbau\ /'StWn-bB/ seekrank\ /'ze-kr\&Nk/ seeartig\ /'ze-ar-tIx/

and Salzsee\ /'z\&l=-ze/ M\ "unchen\ /'mYn-x@n/

Disc is multilingual, so where for Dutch /a/ and /A/ exist, German has the latter only in Kalevala /'kA-IE-vA-IA/, otherwise /&/.

Morphology

There are two major fields with morphology, the immediate analysis and the full analysis. The immediate analysis only contains the top-level breakup, the full analysis is hard to read (by human or computer):

freigeben frei+geb ((frei)[A],(geb)[V])[V]

freigebig freigebig (freigebig)[A]

seeartig See+Art+ig ((See)[N],(Art)[N],(ig)[A|NN.])[A]

Boesartigkeit boesartig+keit (((boes)[A],(Art)[N],(ig)[A|AN.])[A],
(keit)[N|A.])[N]

The [N|A.] notation with *-keit* means that a Noun is formed out of an Adjective and this element (the dot). Notice that Celex is not always consistent in its analyses.

Inflectional Classes

German Celex has an extra facility: The inflectional paradigm for nouns and verbs is listed in the 'InflPar' field:

```
Zahn '=an (Zahn)[N] S1/Plu
```

More details in the German Users Guide, see the file `gug_a4.ps` on the cdrom. For irregular verb codes see Brockhaus-Wahrig Wörterbuch, 'gug' file p174. For nouns p180 of this file.

Celex on CDrom – the dumps

The Celex files on CDrom are so-called dumps of the larger database. Each file contains the number and name of each word, and then some selected fields. The name of the file consists of a language letter [deg: dutch, english, german], a dump type [opms: orthography, phonology, morphology, syntax], and a form type [lw: lemmas, wordforms].

The CDrom is first divided into three language subdirectories. These contain the local user guides (gug, dug, eug) in postscript format (you will need `ghostview`). Then they contain a directory for each lexicon (gml, gmw, gpl, etc). Each subdirectory then contains the dump in plain text with backslashes as separators (gml.cd), a readme file, and sometimes some Awk or C programs to make the dumps look nicer.

The Celex Readme's

The Celex readme's are intended to be read with the User Guide in hand. It lists for each field the name in the database, which is also how you can find it in the User Guide:

1. IdNum
2. Head
3. Mann
4. MorphStatus
5. MorphCnt

Creating your own dumps

The dumps on the CDrom often separate things too much: I frequently want phonological and morphological information (from dpl and dml).

You can create your own dumps

- (1) From the Flex interface, by saving a lexicon to disk and then moving it to your computer with ftp (user friendly).
- (2) By using awk or any other language (flexible).
- (3) By using unix commands 'cut' and 'paste' (easy).

```
cut -d\\ -f1-3,7 dpl > part1
```

```
cut -d\\ -f4-6 dml > part2
```

```
paste -d\\\ part1 part2 > mylexicon.cd
```

```
rm part1 part2
```

(Gnu)Awk – the language

Awk is a small programming language for databases, and made for things like Celex. Its advantages are – available on all unices – easy to install on Dos/Win (one 250k exe file) – standardised – doesn't change anymore and will be around for ever – small and easy to learn.

Perl and Python can do many more things than Awk can do, but they are also harder to learn (esp. Perl). I doubt whether anyone using Celex will **ever** use the extra functionality that these give. If you have to, I advise Python for clarity or Perl because everyone uses it.

Principles of Awk

Standardly, Awk opens a file, reads it line by line, splices each line into words, and then looks at the *rules* in your program to see which ones apply. Words are referred to with \$1 etc, the whole line is \$0

```
BEGIN { FS="\\ " }  
$2 ~ /Schu/ { print $1, $2, $4 }
```

```
gawk -f schu1 gmp1.cd
```

This will only find the words that start with 'Schu', but the match syntax lets you specify alternatives:

```
BEGIN { FS="\\ " }  
$2 ~ /[Ss]chu/ { print $1, $2, $4 }  
$2 ~ /Schu/ || $2 ~ /schu/ { print $1, $2, $4 }  
$2 ~ /.chu/ { print $1, $2, $4 }
```

[ab] matches 'a' or 'b', | | means or (&& means and), . matches any one character. These are regular expressions.

Useful tests

Some useful functions to use in tests are:

- `length(string)`: returns the length in letters of a string. To test whether the word is short enough `length($2) < 10`
- `gsub(search,replace,string)`: changes the string by replacing things. Returns the number of replacements made. To remove the stress and syllable marks from Disc:

`gsub(/ [' " -] / , " " , $6)` To count how many syllables a word has: `nsyl=gsub(/ - / , " - " , $6) + 1`

- Comparison: `<` `>` `<=` `>=` `==` `!=` `!`

Combinations of tests

To look for all stress-initial verbs that start with a plosive, have two syllables, and are at least of a frequency of 10:

```
$6 ~ /^[pbt dk]/ && $4>=10 && $15==4 { # plosive,freq,verb
  nsyl=gsub(/-/,"-", $6)+1;          # compute syllables
  if (nsyl==2 || nsyl==0) {
    print $1,$2,$6 }}
```

This uses \$15 of my dump, that lists a word class code (4 for verbs).

Alternatively, we could have required that the segmentation ends in

.. [V] with something like \$8 ~ /\ [V] \$ /, which looks funny

because [is a special character and needs a \ and \$ means “end of the string”.

Use \$6 ~ /^[?][pbt dk]/ to make the stress mark optional. Leaving it out altogether asks for stress-final words.

More More More...

The official and old-fashioned reference to Awk is Aho, Kernigan, and Weinberg 1979. Arnold Robins has a book out with O'Reilly, also available on line (search for "Effective AWK Programming", this is recommended! An earlier O'Reilly book ("Sed and Awk") is not so good. Almost everything Celex is included on the CDrom, view the postscript (*.ps) files. There is a Celex website, but it isn't very informative, AFAIK.