NEWQUAY & TERMITE : Keyword & Keystring Finding Software

(User Notes by Richard Forsyth, August 2014; March 2015)

These programs identify single keywords (newquay3.py) and distinctive token-sequences (termite3.py) in particular categories of texts. The programs have been written in Python3 and are released under the GNU Public License for general usage. (The trailing digits in the program names are version numbers that may change if the programs are revised.)

Why I Wrote this Software

At several corpus linguistics conferences in the recent past I have noticed with a certain amount of frustration that some speakers seem to imply that there is only one way of measuring "keyness", namely using the "log likelihood" formula, which according to my *Oxford Dictionary of Statistics* (Upton & Cook, 2006) is more correctly called the "likelihood-ratio goodness-of-fit statistic" or G-squared for short. (I will refer to it as G2 in the rest of this document.)

My conception of a keyword is a token whose pattern of occurrence makes it distinctive of a particular class of text. This notion includes the possibility of negative keys, which occur unusually rarely in certain text types. (For fuller discussion of this concept, see Scott & Tribble (2006).) There are many different ways of computing the distinctiveness of tokens in textual data, with different advantages and disadvantages relative to G2.

I wrote these programs

- firstly to enable exploration of the pros and cons of a variety of keyness indices, including G2,
- secondly to allow more than 2 categories to be compared (not just a single class of texts against a reference corpus),
- and thirdly (with termite3.py) to seek sequences of tokens that cluster around each keyword more than would be expected, potentially constituting key phrases or multi-word terms.

The programs also allow numbers and, optionally, other symbols such as punctuation to be treated as tokens, but for brevity I use "keywords" in what follows to mean "key tokens", whether entirely alphabetic or not.

There is no Master Key

An indication that different ways of quantifying keyness have different effects can be gained from the two listings below. Both are based on the same data, a corpus of fiction by three different authors, Agatha Christie, Edith Wharton and Iris Murdoch. The first listing shows the highest-scoring 20 keywords of Agatha Christie (contrasted with the other two authors) using G2. The second listing gives Agatha Christie's highest-scoring 20 keywords ranked according to NeoZeta (a measure that will be explained below). The former contains five proper names, the latter only 2.

It is certainly true that many of Agatha Christie's stories feature Poirot or Marple (never both) so the first listing does tell us something about her writings, but if the context were authorship attribution one could expect that it would be easy for a Christie imitator to make sure that either Marple or Poirot was mentioned, though such an imitator would be less likely to ensure that the word "have" was somewhat over-represented. As for "oliver" and "craddock", they will be found only when a story includes particular character, thus cannot be taken to characterize Christie's authorial style.

The point isn't that G2 is in some sense wrong; merely that different ways of assessing distinctiveness may be more or less suitable for different purposes.

Top 20 keys by G2:

```
AC
## 0   53          150147     459981
``                   rank    quayval   #self #other    %self   %other %midrate
poirot                  1  2032.9782     725      0   0.4829   0.0000   0.0000
said                    2  1032.7278    1691   1662   1.1262   0.3613   0.5697
miss                    3   597.9288     583    350   0.3883   0.0761   0.0231
marple                  4   546.8010     195      0   0.1299   0.0000   0.0000
is                      5   432.0544    1118   1459   0.7446   0.3172   0.4372
you                     6   427.0729    2590   4714   1.7250   1.0248   1.3893
i                       7   426.1323    3638   7244   2.4230   1.5748   1.8715
inspector               8   350.5675     129      1   0.0859   0.0002   0.0000
very                    9   348.2776     588    591   0.3916   0.1285   0.1974
think                  10   345.8152     537    507   0.3576   0.1102   0.1638
sir                    11   330.4711     236     89   0.1572   0.0193   0.0000
quite                  12   315.7633     305    181   0.2031   0.0393   0.0756
m                      13   240.2301     139     33   0.0926   0.0072   0.0000
hercule                14   235.5451      84      0   0.0559   0.0000   0.0000
superintendent         15   193.4834      69      0   0.0460   0.0000   0.0000
oliver                 16   190.6793      68      0   0.0453   0.0000   0.0000
know                   17   188.6141     592    848   0.3943   0.1844   0.2315
do                     18   180.0755     586    853   0.3903   0.1854   0.2256
yes                    19   169.5637     442    579   0.2944   0.1259   0.1396
craddock               20   164.1493      62      1   0.0413   0.0002   0.0000
```

Top 20 Keys by NeoZeta:

```
AC
## 0   53          150147     459981
``                   rank    quayval   #self #other    %self   %other %midrate
said                    1     0.3564    1691   1662   1.1262   0.3613   0.5697
poirot                  2     0.3099     725      0   0.4829   0.0000   0.0000
you                     3     0.2843    2590   4714   1.7250   1.0248   1.3893
i                       4     0.2545    3638   7244   2.4230   1.5748   1.8715
is                      5     0.2244    1118   1459   0.7446   0.3172   0.4372
very                    6     0.1969     588    591   0.3916   0.1285   0.1974
miss                    7     0.1937     583    350   0.3883   0.0761   0.0231
think                   8     0.1834     537    507   0.3576   0.1102   0.1638
do                      9     0.1553     586    853   0.3903   0.1854   0.2256
quite                  10     0.1553     305    181   0.2031   0.0393   0.0756
know                   11     0.1513     592    848   0.3943   0.1844   0.2315
yes                    12     0.1399     442    579   0.2944   0.1259   0.1396
what                   13     0.1317     759   1578   0.5055   0.3431   0.3897
have                   14     0.1202     896   1827   0.5967   0.3972   0.4526
are                    15     0.1152     428    709   0.2851   0.1541   0.2000
say                    16     0.1113     350    492   0.2331   0.1070   0.1388
there                  17     0.1035     764   1500   0.5088   0.3261   0.3900
don't                  18     0.1009     418    794   0.2784   0.1726   0.1919
got                    19     0.1002     277    356   0.1845   0.0774   0.0992
marple                 20     0.0920     195      0   0.1299   0.0000   0.0000
```

Setting Up

First you need Python3. If you don't have it already, the latest version can be downloaded and installed from the Python website: www.python.org. This is usually quite straightforward. The only snag is if you have Python2 and want to keep using it. Then you'll probably have to set up a specific command to run whichever version you use less frequently.

Next step is to unpack the keysoft.zip file. After unpacking it (into a folder called "keysoft", unless you want to do quite a lot of editing), you should find the following subfolders.

op
p3
parapath
samples

The programs are in p3. Sample test corpora will be found in samples. Subfolder op is the default location for output files and parapath is a convenient place for storing parameter files, which will be explained later.

Corpus Format

This software is document-oriented. It presumes that a corpus consists of a number of separate text files (in UTF8 encoding). Each file is treated as an individual document, belonging to a particular category.

In the samples folder you will find 2 subfolders (ajps, cics). These contain datasets that enable you to start using the system, prior to collecting &/or reformatting your own corpora.

The first contains poems by 2 eminent 19th-century Hungarian poets, Arany József & Petőfi Sándor. Arany was godfather to Petőfi's child, so we might expect their writing styles to be relatively similar.

The second contains writings by several Latin authors, the three main ones being: Marcus Tullius Cicero, the famous Roman orator, Mark-Antoine Muret, known as Muretus, and Carlo Sigonio. This dataset arises from an interesting authorship problem. Background information can be found in Forsyth et al. (1999), but in a nutshell the problem revolves around a work called the *Consolatio* which Cicero wrote in 45 BC. This was thought to have been lost until in 1583 AD Carlo Sigonio claimed to have rediscovered it. He died the following year never having made public the manuscript, but published a printed version in Venice with himself named as editor. Scholars have argued since then over whether the book is genuinely a rediscovery of Cicero's lost work or a renaissance fake.

Anything You Can Do, I Can Do Meta (;-)

Sorry, couldn't resist that.

Below is a complete listing of a metafile relating to (part of) the cics dataset. It has three columns. A metafile could have more columns than three, but not less. The top line is a header, giving the column names. The first column must be called prepath. It indicates the directory/folder where a particular file resides. The second must be called filename and is the file name of a particular text. The other column contains class labels. It can be called anything, though doctype is the default. (See details of parameter files for alternative ways of indicating the class of a text.) Columns are separated by the horizontal tab character. (Code point 9 in ASCII and Unicode/utf8.) Each line refers to a separate document.

```
prepath      filename     doctype
c:\keysoft\samples\cics\Tullies\ Cicero_Amicitia.txt cics
c:\keysoft\samples\cics\Tullies\ Cicero_ArchiaPoeta.txt    cics
c:\keysoft\samples\cics\Tullies\ Cicero_Atticus1.txt cics
c:\keysoft\samples\cics\Tullies\ Cicero_Brutus1.txt  cics
c:\keysoft\samples\cics\Tullies\ Cicero_Brutus2.txt  cics
c:\keysoft\samples\cics\Tullies\ Cicero_Cat2.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_CatoSenectute.txt   cics
c:\keysoft\samples\cics\Tullies\ Cicero_DeFinibus.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_DeImperio.txt       cics
```

```
c:\keysoft\samples\cics\Tullies\ Cicero_DeInventione2_latlib.txt  cics
c:\keysoft\samples\cics\Tullies\ Cicero_DeLegibus.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_DePartitione_latlib.txt   cics
c:\keysoft\samples\cics\Tullies\ Cicero_InPisonem_latlib.txt      cics
c:\keysoft\samples\cics\Tullies\ Cicero_InVerremII2_latlib.txt    cics
c:\keysoft\samples\cics\Tullies\ Cicero_NaturaDeorum2.txt  cics
c:\keysoft\samples\cics\Tullies\ Cicero_Officiis1.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_Orator.txt   cics
c:\keysoft\samples\cics\Tullies\ Cicero_Philippics2.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_Philippics7.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProCaecina_latlib.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProCluentio.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProFlacco_latlib.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProMarcello.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProMilone_latlib.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProQuinctio_latlib.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProSestio_latlib.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProSexto.txt cics
c:\keysoft\samples\cics\Tullies\ Cicero_ProSulla.txt cics
c:\keysoft\samples\cics\Tullies\ Cicero_Rep2.txt     cics
c:\keysoft\samples\cics\Tullies\ Cicero_Somnium.txt  cics
c:\keysoft\samples\cics\Tullies\ Cicero_Tusculan1.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_Tusculan2.txt       cics
c:\keysoft\samples\cics\Tullies\ Cicero_Tusculan4.txt       cics
c:\keysoft\samples\cics\neolats\ Muretus_Ingress.txt muretus
c:\keysoft\samples\cics\neolats\ Muretus_Laud.txt    muretus
c:\keysoft\samples\cics\neolats\ Muretus_PaulFox.txt muretus
c:\keysoft\samples\cics\neolats\ Muretus_Phil.txt    muretus
c:\keysoft\samples\cics\neolats\ Muretus_Pius.txt    muretus
c:\keysoft\samples\cics\neolats\ Muretus_Rege.txt    muretus
c:\keysoft\samples\cics\neolats\ Muretus_Util.txt    muretus
c:\keysoft\samples\cics\neolats\ Sigonio_Dialogo.txt sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_Elo1.txt    sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_Elo2.txt    sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_HistIt4a.txt       sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_HistIt4b.txt       sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_LatLing.txt sigonio
c:\keysoft\samples\cics\neolats\ Sigonio_LaudHist.txt       sigonio
```

This metafile describes a corpus with 3 categories: 33 texts by Cicero, 7 texts by Muretus and 7 texts by Sigonio. Many of these 47 texts are extracts rather than full works. Note that no disputed texts are included in this selection.

The format of metafiles is intended to be suitable for manipulation in a spreadsheet package such as Excel or OpenOffice/Calc as a tab-delimited worksheet. The idea behind this is to make it possible to select a variety of subsets of a larger corpus as training or test texts in different runs of the system.

To make your own initial metafile, it is convenient (if many of the files are in a single folder, as is often the case) to use the minimet4.py program. The output of this program can then be edited in a text-editor, such as Notepad++, or a spreadsheet until it specifies exactly the desired set of files. Notepad++, a versatile text-editor that I personally recommend, can be obtained from the website
http://notepad-plus-plus.org/
free of charge.

For example, the following parameter file could be supplied to minimet4.py in order to create a metafile for the non-Ciceronian classical Latin texts held in subfolder claslats, of which there are 17. Briefly, corpath tells the program where the text files are located; metazero specifies the initial metafile to be created and where to place it; and targval gives the value to be put in the doctype column. (More on parameter files below, especially Appendix 2.)

```
comment  parameter file for minimet4.py
```

```
jobname  claslats
corpath  c:\keysoft\samples\cics\claslats\
metazero  c:\keysoft\samples\cics\metadat\claslats.txt
targval  claslat
```

Running minimet4.py with the above parameter file produces the following metafile.

```
prepath      filename     doctype
c:\keysoft\samples\cics\claslats\Caesar_BC2.txt      claslat
c:\keysoft\samples\cics\claslats\Caesar_Gal1.txt     claslat
c:\keysoft\samples\cics\claslats\Nepos_Atticus.txt   claslat
c:\keysoft\samples\cics\claslats\Nepos_Cato.txt      claslat
c:\keysoft\samples\cics\claslats\Nepos_Dion.txt      claslat
c:\keysoft\samples\cics\claslats\Nepos_Hannibal.txt  claslat
c:\keysoft\samples\cics\claslats\Nepos_Milt.txt      claslat
c:\keysoft\samples\cics\claslats\Quintilain_Institutio4_latlib.txt claslat
c:\keysoft\samples\cics\claslats\Quintilian_DecMai19_latlib.txt    claslat
c:\keysoft\samples\cics\claslats\Sallust_BellCat.txt claslat
c:\keysoft\samples\cics\claslats\Sallust_BellJug.txt claslat
c:\keysoft\samples\cics\claslats\Seneca_Brevit.txt   claslat
c:\keysoft\samples\cics\claslats\Seneca_Cons.txt     claslat
c:\keysoft\samples\cics\claslats\Seneca_Ira1.txt     claslat
c:\keysoft\samples\cics\claslats\Seneca_Otio.txt     claslat
c:\keysoft\samples\cics\claslats\Seneca_Prov.txt     claslat
c:\keysoft\samples\cics\claslats\Tacitus_Agricola.txt      claslat
```

This metafile could be appended to the cics.txt metafile to select a four-category subcorpus from the overall corpus.

I have also written an alternative program, called metaget.py, that users accustomed to pointing and clicking might find easier to use, which is included with the distribution. This program can be run just by double-clicking on its name. It will then display a window with four elements:

> Enter next category name:
> Select file(s):
> Enter output metafile name:
> Exit & save metafile:

The idea is that you type a category label in the upper box (then press the Enter button) then choose files by picking the second option which will allow the customary ways of navigating the file system and selecting files or groups of files. This pair of actions can be repeated several times to include files from a number of different categories. Then you provide a destination file name for the resulting metafile (again not forgetting to press the Enter button) and quit using the final option. If you do forget to name the output metafile, it will be called metazero.txt and placed on the directory from which the program was launched.

Note that entering the category or metafile name does require clicking the Enter button alongside the text-entry box to confirm your input; just hitting Carriage-Return won't do, as I have yet to master the intricacies of binding a keypress-response procedure to the Return key. (Still writing programs as if the 20th century hadn't gone out of fashion, I'm afraid. Nevertheless, I suspect most people will find metaget.py somewhat simpler to use than minimet4.py, though I doubt if it will eliminate cases where using a text-editor, such as Notepad++, will still be needed to put a nearly-correct metafile into its final form.)

Preparing a Parameter File

Below is a listing of parameter file cics.txt which comes with the keysoft distribution.

```
comment  Cics testing :
jobname  cics
metafile  c:\keysoft\samples\cics\metadat\cics.txt
##metafile  c:\keysoft\samples\cics\metadat\cic3.txt
wordonly  1
targvar  doctype
keymode  g2
topkeys  26
```

A parameter file is just a plain text file with one item per line. Each line should begin with the parameter name, then 1 or more blank spaces, then the parameter value. The following table interprets the above parameter file, line by line.

Parameter	Default value	Function
comment	[None]	This (or in fact any unrecognized parameter name, e.g. "##") can be used to insert reminders about what the file is meant to do.
jobname	newquay	This gives the job a name. Any text string can be the value. It isn't necessary but it is useful as the jobname will be used as a prefix to the program's output files, so it can be seen that they form a group.
metafile	[None]	This should be the full file specification of a metafile that indicates the text files that belong to the corpus, each associated with its target variable, i.e. class label, as described above.
wordonly	0	This should be integer 0 or 1. If it is 1, the tokenizer will ignore tokens unless they begin with an alphanumeric character. If it is zero, all tokens will be considered, even sequences of punctuation symbols and so on. Since punctuation of works authored by Cicero is definitely not Cicero's, it is set to 1 here.
targvar	doctype	This gives the name of the column in the metafile that contains the target variable, i.e. the class identification of each file.
keymode	g2	This indicates which function should be used to compute the keyness index. Details of keyness indices implemented follow below, after specimen output.
topkeys	32	This specifies how many of the highest scoring keys should be listed in the output file. The listing will contain double this number for each category, since the same number of positive and negative keys will be listed.

Running newquay3.py

When you run newquay3.py it will ask for a parameter file. If this file is in the same directory as the program, or in its parapath subfolder, you won't have to give the full path specification, just its name (and .txt extension will be presumed).

Using the parameter file shown in the previous section on the cics dataset produces four output files. The most important will have a name composed of the jobname, then "_keys", with .txt extension. A listing of the output file cics_keys.txt is shown below. (The other three output files are mainly relics for debugging.)

```
Wed Nov  6 17:17:29 2013
parafile: C:\keysoft\parapath\cics.txt
metafile: c:\keysoft\samples\cics\metadat\cics.txt
keymode : g2
47 3

cics
```

```
## 0   33          202593      44637
``                    rank    quayval   #self #other    %self   %other %midrate
te                       1   166.5289     695     19   0.3431   0.0426   0.1029
c                        2   134.5858     364      1   0.1797   0.0022   0.0287
non                      3   113.6102    3118    406   1.5390   0.9096   1.3135
p                        4    98.7635     248      0   0.1224   0.0000   0.0000
est                      5    84.7624    2869    397   1.4161   0.8894   1.1377
l                        6    81.6392     205      0   0.1012   0.0000   0.0103
ego                      7    76.0952     457     23   0.2256   0.0515   0.1263
nec                      8    74.2937     630     45   0.3110   0.1008   0.1369
tu                       9    72.8951     373     15   0.1841   0.0336   0.0696
enim                    10    69.0338    1117    119   0.5514   0.2666   0.3659
sed                     11    68.1048    1516    184   0.7483   0.4122   0.6504
m                       12    58.0897     278     10   0.1372   0.0224   0.0459
q                       13    57.7448     145      0   0.0716   0.0000   0.0000
quid                    14    56.1585     961    105   0.4744   0.2352   0.3854
iudices                 15    53.3642     134      0   0.0661   0.0000   0.0000
sex                     16    41.0187     103      0   0.0508   0.0000   0.0000
iste                    17    40.0107     121      1   0.0597   0.0022   0.0000
consul                  18    39.0275      98      0   0.0484   0.0000   0.0000
esset                   19    38.9229     477     44   0.2354   0.0986   0.1568
cn                      20    38.2310      96      0   0.0474   0.0000   0.0000
autem                   21    37.8253     895    111   0.4418   0.2487   0.2575
nihil                   22    37.3118     640     70   0.3159   0.1568   0.2388
quem                    23    36.6072     446     41   0.2201   0.0919   0.1811
tibi                    24    36.1309     284     19   0.1402   0.0426   0.0386
modi                    25    31.4610      79      0   0.0390   0.0000   0.0000
contra                  26    31.3467     210     12   0.1037   0.0269   0.0635
prope                  -26   -38.9913      41     39   0.0202   0.0874   0.0260
sibi                   -25   -40.9060     201    101   0.0992   0.2263   0.1283
ecclesia               -24   -41.0821       0     12   0.0000   0.0269   0.0000
aristotelis            -23   -41.0821       0     12   0.0000   0.0269   0.0000
imperii                -22   -42.2381      10     23   0.0049   0.0515   0.0000
cognitionem            -21   -42.4744       5     19   0.0025   0.0426   0.0000
deo                    -20   -45.7840      20     31   0.0099   0.0694   0.0000
litterarum             -19   -47.9525      27     36   0.0133   0.0807   0.0000
pontificis             -18   -50.8738       1     17   0.0005   0.0381   0.0000
linguae                -17   -52.6345       2     19   0.0010   0.0426   0.0000
regis                  -16   -52.6922       9     26   0.0044   0.0582   0.0000
quotidie               -15   -54.7762       0     16   0.0000   0.0358   0.0000
italiae                -14   -59.0680      18     35   0.0089   0.0784   0.0000
caroli                 -13   -65.0467       0     19   0.0000   0.0426   0.0000
s                      -12   -65.0467       0     19   0.0000   0.0426   0.0000
anno                   -11   -69.2647      39     52   0.0193   0.1165   0.0000
ad                     -10   -71.3564    1601    545   0.7903   1.2210   0.8314
auditores               -9   -71.8938       0     21   0.0000   0.0470   0.0000
adversus                -8   -72.6544       6     30   0.0030   0.0672   0.0000
carolum                 -7   -75.3173       0     22   0.0000   0.0493   0.0000
dei                     -6   -79.6680       4     30   0.0020   0.0672   0.0000
artium                  -5   -84.0590      15     42   0.0074   0.0941   0.0000
demum                   -4   -90.9114       1     29   0.0005   0.0650   0.0000
ecclesiae               -3  -106.1289       0     31   0.0000   0.0694   0.0000
inde                    -2  -123.3643      25     64   0.0123   0.1434   0.0000
ac                      -1  -515.4698     501    496   0.2473   1.1112   0.3630

muretus
## 1    7           18542     228688
``                    rank    quayval   #self #other    %self   %other %midrate
ac                       1   311.6948     258    739   1.3914   0.3231   0.3630
litterarum               2    50.8044      25     38   0.1348   0.0166   0.0000
quotidie                 3    44.7957      12      4   0.0647   0.0017   0.0000
quidquam                 4    42.9057      11      3   0.0593   0.0013   0.0000
sibi                     5    40.9191      57    245   0.3074   0.1071   0.1283
eas                      6    40.8769      27     61   0.1456   0.0267   0.0275
regina                   7    40.2793       9      1   0.0485   0.0004   0.0000
quasique                 8    36.2639       7      0   0.0378   0.0000   0.0000
philosophiae             9    33.5672      14     16   0.0755   0.0070   0.0000
quamlibet               10    30.3915       7      1   0.0378   0.0004   0.0000
christi                 11    30.3915       7      1   0.0378   0.0004   0.0000
libros                  12    29.5764      12     13   0.0647   0.0057   0.0000
demum                   13    28.9440      13     17   0.0701   0.0074   0.0000
eruditionis             14    27.0411       7      2   0.0378   0.0009   0.0000
praecipue               15    26.9220      10      9   0.0539   0.0039   0.0000
ita                     16    26.8203      91    598   0.4908   0.2615   0.2747
praestantium            17    25.9028       5      0   0.0270   0.0000   0.0000
artium                  18    24.8381      17     40   0.0917   0.0175   0.0000
amplissimi              19    24.3643       9      8   0.0485   0.0035   0.0000
```

olim	20	23.8992	11	15	0.0593	0.0066	0.0000
publice	21	23.8992	11	15	0.0593	0.0066	0.0000
adversus	22	23.8415	13	23	0.0701	0.0101	0.0000
earum	23	22.5775	20	61	0.1079	0.0267	0.0098
adolescentes	24	22.3978	6	2	0.0324	0.0009	0.0000
illos	25	22.2502	17	45	0.0917	0.0197	0.0154
prius	26	22.1793	12	21	0.0647	0.0092	0.0000
autem	-26	-15.3721	45	961	0.2427	0.4202	0.2575
is	-25	-15.6938	8	316	0.0431	0.1382	0.0890
hoc	-24	-16.0096	47	1003	0.2535	0.4386	0.3695
etiam	-23	-16.0574	55	1131	0.2966	0.4946	0.4466
sex	-22	-16.0599	0	103	0.0000	0.0450	0.0000
ante	-21	-17.1810	6	286	0.0324	0.1251	0.0975
quia	-20	-17.7496	2	190	0.0108	0.0831	0.0551
tu	-19	-17.8353	10	378	0.0539	0.1653	0.0696
contra	-18	-17.9047	3	219	0.0162	0.0958	0.0635
ego	-17	-18.6270	14	466	0.0755	0.2038	0.1263
iste	-16	-19.0223	0	122	0.0000	0.0533	0.0000
senatus	-15	-19.1783	0	123	0.0000	0.0538	0.0000
iudices	-14	-20.8934	0	134	0.0000	0.0586	0.0000
maxime	-13	-21.7059	2	219	0.0108	0.0958	0.0821
q	-12	-22.6085	0	145	0.0000	0.0634	0.0000
m	-11	-22.8463	4	284	0.0216	0.1242	0.0459
non	-10	-27.7659	186	3338	1.0031	1.4596	1.3135
inquit	-9	-28.3776	0	182	0.0000	0.0796	0.0000
est	-8	-30.7732	166	3100	0.8953	1.3556	1.1377
solum	-7	-30.8181	1	248	0.0054	0.1084	0.1104
l	-6	-31.9638	0	205	0.0000	0.0896	0.0103
te	-5	-33.7303	18	696	0.0971	0.3043	0.1029
p	-4	-38.6684	0	248	0.0000	0.1084	0.0000
enim	-3	-44.1992	38	1198	0.2049	0.5239	0.3659
nec	-2	-54.8741	9	666	0.0485	0.2912	0.1369
c	-1	-56.9111	0	365	0.0000	0.1596	0.0287

sigonio
2 7 26095 221135

``	rank	quayval	#self	#other	%self	%other	%midrate
inde	1	152.6919	58	31	0.2223	0.0140	0.0000
ac	2	143.7547	238	759	0.9121	0.3432	0.3630
carolum	3	86.5273	21	1	0.0805	0.0005	0.0000
italiae	4	82.6156	33	20	0.1265	0.0090	0.0000
caroli	5	73.3365	18	1	0.0690	0.0005	0.0000
s	6	73.3365	18	1	0.0690	0.0005	0.0000
atque	7	67.2069	236	1063	0.9044	0.4807	0.4636
anno	8	66.4438	40	51	0.1533	0.0231	0.0000
linguae	9	64.3931	18	3	0.0690	0.0014	0.0000
ecclesiae	10	63.5938	22	9	0.0843	0.0041	0.0000
dei	11	63.0824	23	11	0.0881	0.0050	0.0000
italiam	12	54.2017	23	16	0.0881	0.0072	0.0000
imperii	13	53.8555	21	12	0.0805	0.0054	0.0000
ad	14	49.6865	333	1813	1.2761	0.8199	0.8314
quamobrem	15	47.1380	12	1	0.0460	0.0005	0.0000
auditores	16	43.6685	15	6	0.0575	0.0027	0.0000
aristotelis	17	42.8077	11	1	0.0422	0.0005	0.0000
misit	18	41.6838	18	13	0.0690	0.0059	0.0000
artium	19	41.4107	25	32	0.0958	0.0145	0.0000
italia	20	38.5848	24	32	0.0920	0.0145	0.0000
pontificis	21	38.3081	13	5	0.0498	0.0023	0.0000
pontifex	22	36.8634	12	4	0.0460	0.0018	0.0000
quondam	23	36.5962	21	25	0.0805	0.0113	0.0077
ioannes	24	35.9772	8	0	0.0307	0.0000	0.0000
imitatione	25	35.5897	11	3	0.0422	0.0014	0.0000
leonis	26	34.1958	9	1	0.0345	0.0005	0.0000
esset	-26	-22.4577	25	496	0.0958	0.2243	0.1568
tua	-25	-22.7553	0	102	0.0000	0.0461	0.0130
sex	-24	-22.9784	0	103	0.0000	0.0466	0.0000
quidem	-23	-23.1685	56	870	0.2146	0.3934	0.3103
nec	-22	-23.3620	36	639	0.1380	0.2890	0.1369
enim	-21	-23.8877	81	1155	0.3104	0.5223	0.3659
fuit	-20	-25.2877	17	412	0.0651	0.1863	0.1207
iudices	-19	-29.8942	0	134	0.0000	0.0606	0.0000
nihil	-18	-30.7222	34	676	0.1303	0.3057	0.2388
modo	-17	-30.9044	26	574	0.0996	0.2596	0.2202
dicere	-16	-30.9568	3	221	0.0115	0.0999	0.0650
m	-15	-31.5661	6	282	0.0230	0.1275	0.0459
q	-14	-32.3483	0	145	0.0000	0.0656	0.0000
tamen	-13	-34.7936	31	669	0.1188	0.3025	0.2845

```
l                  -12   -45.7337       0    205   0.0000   0.0927   0.0103
mihi               -11   -46.0350      18    563   0.0690   0.2546   0.1926
tibi               -10   -46.7580       3    300   0.0115   0.1357   0.0386
est                 -9   -46.8685     231   3035   0.8852   1.3725   1.1377
quid                -8   -53.3766      47   1019   0.1801   0.4608   0.3854
tu                  -7   -54.4788       5    383   0.0192   0.1732   0.0696
p                   -6   -55.3267       0    248   0.0000   0.1121   0.0000
ego                 -5   -56.1421       9    471   0.0345   0.2130   0.1263
sed                 -4   -56.8865      92   1608   0.3526   0.7272   0.6504
c                   -3   -71.9054       1    364   0.0038   0.1646   0.0287
non                 -2   -80.0579     220   3304   0.8431   1.4941   1.3135
te                  -1  -148.4210       1    713   0.0038   0.3224   0.1029
```

This shows, for example, that the words "te" (thou) and "c" (100 in roman numerals) are most distinctive of Cicero compared to the 2 selected Neolatin authors, while "inde" (thence) and "ac" (and) score highest for Sigonio.

In each category there are eight columns. The first gives the keyword itself. Other columns are as follows.

rank	Rank according to keyness, with negative numbers indicating inverse keyness ranks.
quayval	Score on whichever keyness index is being used.
#self	Total number of occurrences of the word in the selected category.
#other	Total number of occurrences of the word in the other categories taken together.
%self	Percentage occurrence rate in the selected category.
%other	Percentage rate in the other categories taken together.
%midrate	Median of the percentage rates computed separately for each document in the corpus.

Just above rank and category at the head of each block is the number of tokens and the number of characters for that category.

Keyness Indicators

As of version 3, twelve keyness functions have been implemented. These are detailed below. In all descriptions below "rate" refers to the number of occurrences of a token in a text divided by the total number of tokens in that text.

Value of keymode parameter	**Description**
coda	Coefficient Of Differential Abundance. This is analogous to the area under the ROC curve (http://gim.unmc.edu/dxtests/roc3.htm). It is computed by comparing every document in the focus category with every document in all other categories. coda = (h + 0.5*e) / N where N is the total number of comparisons performed, between all rates of the given term in the focus-category and every rate in documents of the other categories, (N = n1*n2); h is the number of comparisons in which the rate in the focus-class document was higher than the rate in the other class document; e is the number of comparisons in which the relative frequencies were equal. This coefficient ranges between 0 & 1, with 0.5 indicating lack of association.
czec	An index proposed by researchers working on the Czech National corpus which they call a variation on Dice's coefficient (mistakenly, as far as I can tell),

	slightly modified to attenuate the effect of zero rates. czec = (r1 - r2) / (r1 + r2) where r1 is (f1+1)/(n1+2); r2 = (f2+1)/(n2+2); f1 is the total frequency of the token in the focus category; f2 is the total frequency in the other categories; n1 is the total number of tokens in the focus category; n2 is the total number of tokens in the other categories. (The 1 added to the numerators and the 2 added to the denominators are attenuation factors that slightly bias this index away from over-sensitivity to zero frequencies -- although not enough, in my view!)
docz	Document-based version of Zeta. This is a variant of Zeta as described in Craig & Kinney (2009) but using individual documents as text blocks whereas Craig & Kinney used fixed-size chunks. See Neoz, below.
fazs	Frequency-Adjusted Z-Score. fazs = 100 * (t1+t2) / (n1+n2) where n1 is the number of focus-category documents; n2 is the number of documents in other categories; t1 is the sum of (z1 * r) for all documents in the focus category; t1 is the sum of (z2 * r) for all documents in the other categories; r is the token's rate in a given document; z1 is (1-p) / s; z2 is -p / s; p is the proportion of documents in the focus category; s is the standard deviation of the proportion p, computed using the binomial formula (p*(1-p))^0.5. Does it look complicated? Basically, it is obtained by multiplying each rate by one or other z-score according to category membership. I invented this one; some day I will attempt to explain why.
g2	G2, the so-called "log-likelihood". (Default method.) Computed in the textbook manner. See for instance the website below. http://ucrel.lancs.ac.uk/llwizard.html
hedg	Hedges's g. This is a statistic designed to measure effect size. hedg = d / sqrt(v) where d is m1-m2; m1 is the mean of all the rates of the token in the focus-category documents; m2 is the mean of all the rates of the token in the other categories; v is the weighted pooled variance of all rates, ((n1-1)*v1 + (n2-1)*v2) / (n1+n2-2); v1 is the variance of the rates in the focus category documents; v2 is the variance in the other categories; n1 is the number of documents in the focus category; n2 is the number of documents in other categories; sqrt is the square root function. In effect, this tries to express the difference in mean rates between the 2 groups in terms of standard deviations; in other words, it is a z-score.
neoz	Neo-Zeta. This is a variant of Zeta as described in Craig & Kinney (2009) using fixed-size chunks of text, here referred to as snippets.

	neoz = r1 - r2
	where
	r1 = x / k1;
	r2 = y / k2;
	x is the number of snippets containing the token in the focus category;
	y is the number of snippets containing the token in the other categories;
	k1 is the number of snippets in the focus category;
	k2 is the number of snippets in the other categories.
	In fact, neoz = z-1 where z is the original zeta. NeoZeta ranges from -1 to +1, which I think is nicer for a quasi-correlation, unlike zeta which ranges from 0 to 2.
	(Default snippet size is 115 tokens, the length of Shakespeare's sonnet XVIII. To change this, give your desired number of tokens as value of the parameter snipsize.)

pbcx	Point-Biserial Correlation Coefficient (modified). Originally this was pbcc, the point-biserial correlation coefficient, but that gives exactly the same ranking as Hedges's g (above), so I modified it to be the product-moment correlation between sqrt(size) or -sqrt(size) for the lengths of the texts in tokens in the focus category and the other categories on the one hand, and sqrt(rate) for the rates (relative frequencies) of the token concerned on the other. If anyone uses it and likes it I might attempt to explain why.
pq	This is computed as pq = 100 * (p * (1-q)) where p is the proportion of snippets in the focus category that contain the token; q is the proportion of snippets in the other categories that contain the token. Despite its simplicity, I haven't come across this index elsewhere. It is slow to compute with large corpora, but gives interesting results and, unlike g2 for instance, yields mostly high-frequency words as positive keys, thus being potentially more suitable as authorial (stylistic?) than content-based markers. The negative keys, on the other hand, tend to be more content-based and give a good indication of topics avoided in the focus category.
ppqq	This is the same as pq, above, except that it uses the proportion of documents for p & q, rather than the proportion of snippets, which contain the token concerned. Unless the documents are of roughly similar sizes, and not very long, I would recommend pq ahead of ppqq.
reld	Relative difference. This is simply the difference in mean rates expressed as a percentage. reld = (m1-m2) * 100 where m1 is the mean rate of the token in the focus-category documents; m2 is the mean rate of the token in the other categories.

roar	Ratio Of Absence Rates. I devised this one as an attempt to avoid the need to add an arbitrary attenuation constant in the "simple maths" index of Kilgarriff (2009), by looking at rates of absence instead of rates of presence. I suspect Engels would have called this move a "negation of a negation" -- though I'd hate to be accused of Hegelianism. ;-)
	roar = (1-p2) / (1-p1)
	where
	p1 = f1 / n1;
	p2 = f2 / n2;

	f1 is the total frequency of the token in the focus category; f2 is the total frequency in the other categories; n1 is the total number of tokens in the focus category; n2 is the total number of tokens in the other categories. For those used to medical statistics, roar can be viewed as specificity / (1-sensitivity) where sensitivity is true_positives / (true_positives+false_negatives); specificity is true_negatives / (false_positives+true_negatives). With word-frequency data there is no realistic danger of dividing by zero, and thus no need to include any fudge factors.

I have pondered the idea of releasing a version some day in which a user can define a Python3 function computing a user-written index and call that (e.g. with keymode of "user") but that would require that the program precompute all reasonable ingredients for such a function in advance, which is likely to prove tricky; so I will wait to see if there is any demand for such a facility.

Running termite3.py

Termite (Text-Exploring Ranked Multi-Item Term Extractor) is a program designed to seek distinctive token sequences. For want of an accepted term, I call these sequences "desmoglyphs", from the Greek *desmos*, a chain, and *glyphe*, carving. This is meant to convey the idea of a fixed series of symbols. In the literature, they would be called word n-grams, but their length (n) isn't given by a user; instead it is a side-effect of the search process. The wide variety of sizes (n's) among these n-grams seemed to justify a specific name.

To give an example of the sorts of subsequences that the system finds, the listing below shows the results of a 2-category comparison, contrasting a standard multi-register corpus, the Lancaster-Oslo-Bergen corpus of 500 texts (Hofland and Johansson, 1982) with 461 documents from a collection of patient information leaflets which were extracted from the Patient Information Leaflet Corpus 2.0, originally compiled at the Natural Language Technology Group at the University of Brighton and discussed in greater detail by Buoayad-Agha & Kilgarriff (1999) and Buoayad-Agha (2006). This corpus is available at http://www.mcs.open.ac.uk/nlg/old_projects/pills/corpus . It contains 465 texts but four were excluded from this study as near-duplicates.

```
Thu Mar 26 15:00:36 2015
parafile: C:\keywork\parapath\lobpils.txt
metafile: c:\keywork\mets\lobpils.txt
keymode : coda
sidespan: 8
961 2

Desmoglyphs by category :

# 0 lobcorp
 Sortval Natscore docrate+ docrate-              seed    subsequence
    0.86     1.49   57.40%    0.00%              been => had been
    0.82     2.11   41.20%    5.64%               did => did not
    0.65     1.12   59.60%    2.60%             would => would be
    0.57     1.83   31.60%    0.87%              fact => in fact
    0.49     1.42   35.00%    0.65%               say => to say
    0.47     1.13   60.40%   30.37%             there => there is
    0.47     1.74   27.40%    0.43%              fact => fact that
    0.44     1.33   36.00%    8.24%            little => a little
    0.43     1.25   35.40%    2.17%              good => a good
    0.43     1.61   26.80%    0.00%               own => his own
    0.42     1.12   38.00%    0.00%              said => he said
    0.40     1.23   33.00%    1.52%             great => a great
    0.31     1.50   21.00%    0.22%               far => so far
    0.30     1.23   25.80%    4.12%             those => those who
```

```
    0.10     1.05   99.40%   90.24%                on => on the

# 1 patleaf
 Sortval Natscore docrate+ docrate-              seed    subsequence
   50.67   170.51   29.72%    0.00%         questions => if you have any questions or are not
sure about anything ask your
   45.97   166.88   27.55%    0.00%              sure => if you have any questions or are not
sure about anything ask your doctor or pharmacist
   10.70    29.90   35.79%    0.00%           licence => product licence holder
    7.17     8.43   85.03%    0.00%              tell => tell your doctor
    4.94     7.69   64.21%    0.00%           feeding => breast feeding
    4.47     5.63   79.39%    0.00%              read => please read
    3.16     8.33   37.96%    0.00%           product => the product licence
    3.04    23.79   12.80%    0.00%           contain => this leaflet does not contain the
complete information about
    2.93     6.39   45.99%    0.20%             reach => reach of children
    2.78     8.22   33.84%    0.00%         carefully => please read this leaflet carefully
    2.74     6.02   45.55%    0.00%              date => the expiry date
    2.53    18.49   13.67%    0.00%          symptoms => it may harm them even if their
symptoms are the same as yours
    2.32     7.63   30.37%    0.00%         carefully => carefully before you start to take
    2.28     7.96   28.63%    0.00%            please => please read this leaflet carefully
before
    2.09     3.35   62.47%    0.00%           effects => side effects
    2.09     3.37   92.19%   32.80%                do => do not
    1.70     8.35   20.39%    0.00%             start => please read this leaflet carefully
before you start
    1.64     5.18   31.67%    0.00%          remember => as soon as you remember
    1.46     5.06   28.85%    0.20%          children => out of the reach of children
    1.30     6.30   20.61%    0.00%         medicines => is one of a group of medicines
called
    1.27     7.67   16.49%    0.00%          allergic => had an allergic reaction to
    1.24     3.88   32.10%    0.00%            should => you should know about
    1.21     5.63   21.48%    0.00%          remember => remember this medicine is for you
only a doctor
    1.20     2.68   44.90%    0.00%              used => used to treat
    1.15     1.88   75.70%   19.20%             other => any other
    1.09     3.50   31.02%    0.00%             about => what you should know about
    1.08     3.46   31.24%    0.00%             blood => high blood pressure
    1.07     2.24   48.16%    0.40%               use => do not use
    1.06     3.46   30.59%    0.00%       ingredients => inactive ingredients
    1.05     3.81   27.55%    0.00%          pregnant => if you are pregnant
    0.98     7.25   13.45%    0.00%             reach => in a safe place where children
cannot reach
    0.95     3.36   28.20%    0.00%            before => before you start to take your
    0.92     2.53   36.23%    0.00%            taking => taking your medicine
    0.80     2.04   39.48%    0.20%             cause => may cause
    0.78     1.98   39.70%    0.00%              take => take your medicine
    0.74     2.13   34.71%    0.00%              stop => stop taking
    0.74     3.69   19.96%    0.00%             start => start to take your medicine
    0.64     2.54   25.38%    0.00%              your => carefully before you start to take
your
    0.62     3.32   18.66%    0.00%              stop => if your doctor decides to stop
    0.61     3.33   18.22%    0.00%           leaflet => leaflet carefully before you start
to take
    0.45     2.65   17.14%    0.00%          medicine => carefully before you start to take
your medicine
    0.44     2.34   18.87%    0.00%           effects => effects in some people
    0.34     1.60   21.26%    0.00%             after => after taking your medicine
    0.31     1.52   20.61%    0.00%              dose => if you forget to take a dose
    0.28     2.95    9.33%    0.00%             cause => cause unwanted effects in some
people
    0.19     1.39   13.88%    0.00%               may => may want to read it again
    0.19     1.95    9.76%    0.00%              date => date shown on the
    0.17     2.36    7.38%    0.00%              keep => keep this leaflet you may want to
read it
    0.17     1.98    8.68%    0.00%              tell => answer to any of these questions is
yes tell
    0.15     2.94    4.99%    0.00%           feeding => feeding if the answer to any of
these questions
    0.15     2.49    5.86%    0.00%          pregnant => pregnant trying to become pregnant
or breast feeding
    0.14     1.78    7.81%    0.00%          contains => contains important information about
your
    0.13     2.45    5.42%    0.00%            breast => are you pregnant trying to become
pregnant or breast
    0.13     1.67    7.59%    0.00%       information => information this leaflet does not
```

```
contain
    0.10     1.52    6.51%    0.00%          treatment => wear off after a few days treatment
if they are severe or
```

This output illustrates quite well the typical differences between desmoglyphs found in a generic corpus and those found in a more restricted text collection.

All the sequences derived from the LOB corpus are just 2 tokens in length. On the other hand, the specific corpus gives rise to a number of long sequences, including "if you have any questions or are not sure about anything ask your doctor of pharmacist" which contains 16 tokens. Many of these fragments could be described as fixed formulaic phrases, or parts of such phrases. They are not terms in the terminological sense, mainly because this corpus consists of texts aimed at the general public; though with a more technical corpus, the program is capable of uncovering technical terminology. Most of the sequences are syntactically incomplete: they are best regarded as prefabricated chunks used to compose a rather rigid kind of discourse within a narrow field. In any case, they do give an insight into the text-type under consideration.

Interpreting this output

The first few lines of this output file record the parameter file and metafile that were being used. The keymode used to generate the initial list of keys is also given, as is the sidespan, which determines how many tokens either side of the seed token will be considered in searching for desmoglyphs. Then there follows a block of output for each text category. The headings of each block give the meaning of each column.

Sortval is the numeric value used to rank the subsequences. Natscore is a measure of collocational coherence. Both of these are explained in the next section.

The headings docrate+ and docrate- give the percentages of the focus category documents that contain the subsequence and of the other-category documents that contain that subsequence, respectively.

Seed is the keyword around which the subsequence is built up, and subsequence is the resulting desmoglyph itself.

Thus, for example, we can see in the listing above that the phrase "it may harm them even if their symptoms are the same as yours" was generated from the seed keyword "symptoms" and occurs in 13.67% of the patient leaflets but in none of the LOB texts. By contrast, the phrase "a little" which was generated from "little" occurs in 36.00% of the LOB texts but only 8.24% of the patient information leaflets.

How the program works

The basic idea behind this program is to use keywords as starting points from which to develop distinctive subsequences. Thus the program first generates a list of keys for each text type in the same way as newquay3.py and then scans the near context of these keywords for sequences that occur relatively frequently. This implies that desmoglyphs that don't include a single keyword will not be found. (Such things are conceivable, but presumably rare.)

More precisely, for each keyword, all n-grams that precede it anywhere in the focus-category texts (for values of n up to the input parameter sidespan, which defaults to 5) will be collected; as will all successor n-grams. Any of these n-grams that occur in the focus category with frequency less than the square root of the frequency of the seed word will then be discarded. The remainder are ranked

according to the value shown as Natscore in the listing above. This is computed as the sum of $-\ln(p_j)$ for each token j in the subsequence, where p_j is $(f_j+1)/N$, with f_j being the overall frequency of that token and N the size of the whole corpus in tokens. This total is then multiplied by w, where w=n/F, n being the frequency of the subsequence and F the frequency of the seed keyword (within the focus category).

Obviously this is a heuristic measure. Its value can only be tested empirically. Factors that make it higher are: longer subsequences, composed of rare words, which occur relatively often preceding (following) the seed keyword.

Another heuristic is used to decide whether to join a predecessor with a successor subsequence (e.g. "had an allergic" with "allergic reaction to" giving "had an allergic reaction to"). At present this is simply whether they occur together in more than 38.1966% of the cases where they could occur.

Sortval, which is used to rank the desmoglyphs on output, is computed as Natscore × pp × qq, where Natscore is as described above, pp is the proportion of focus-category documents that contain the subsequence, and qq is the proportion of documents in the other categories that don't contain the subsequence. At present, subsequences that are wholly contained within subsequences higher up the output list are silently suppressed. This does cut out a certain amount of quasi-repetition.

This program is particularly good at picking up formulaic fragments ("boilerplate" language). It also throws up, at least in English, a good number of technical terms when used to contrast a general with a technical corpus. Whether desmoglyphs, as discovered by termite3.py, have potential as diagnostic indicators has yet to be tested. Since there is no established criterion for evaluating such things, feedback from users (if any!) would be welcome.

Acknowledgements

Thank you for reading this far. :-)

References

Bouayad-Agha, N. 2006. The Patient Information Leaflet (PIL) 2.0 corpus. Available at: http://mcs.open.ac.uk/nlg/old_projects/pills/corpus/PIL/ (accessed May 2012).

Craig, H. & Kinney, A.F. (2009). *Shakespeare, Computers and the Mystery of Authorship*. Cambridge: Cambridge University Press.

Forsyth, R.S., Holmes, D.I. & Tse, E.K. (1999). Cicero, Sigonio, and Burrows: investigating the authenticity of the "Consolatio". *Literary & Linguistic Computing,* 14(3), 1-26.

Hofland, K. and Johansson, S. (1982). *Word frequencies in British and American English*. Bergen: Norwegian Computing Centre for the Humanities/London: Longman.

Kilgarriff, A. (2009). Simple Maths for Keywords Proc. Corpus Linguistics, Liverpool, July 2009. http://www.kilgarriff.co.uk/publications.htm

Scott, M. & Tribble, C. (2006). Textual Patterns: *Key Words and Corpus Analysis in Language Education*. Philadelphia: John Benjamins.

Upton, G. & Cook, I. (2006). *Oxford Dictionary of Statistics*, second ed. Oxford: Oxford Univ. Press.

Appendix 1 : Metafiles

A metafile is a kind of data dictionary. It specifies which text files to work on, and may link associated data with each file. The main point is that metafiles can be read into a spreadsheet program such as Excel, modified, then written back out again to guide further processing (without necessarily rearranging a large collection of documents on disc). Another point to note is that all the software described herein assumes that the first 2 columns of a metafile are called "prepath" and "filename" and contain the file path then the file name. Columns within a metafile are delimited by the horizontal tab character. The programs newqyay3.py and termite3.py also need a third column, called "doctype" by default.

The first line of a metafile is treated as a header, giving column names.

As an example, the Hungarian poetry metafile (keysoft\samples\ajps\mets\ajps.txt) is listed below. Since all these texts have doctype "poem", the column "producer" would be used as the value of targvar for authorial contrast in this case.

prepath	filename	doctype	year	monthnum	producer
c:\keysoft\samples\ajps\aj\	AJ_KedvesBaratom_1847.txt	poem	1847	8	AJ
c:\keysoft\samples\ajps\aj\	AJ_Telben_1848.txt	poem	1848	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_ValaszPetofinek_1847.txt	poem	1847	2	AJ
c:\keysoft\samples\ajps\aj\	AJ_VarroLeanyok_1847.txt	poem	1847	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_CzakoSirjan_1848.txt	poem	1848	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_MehRomanca_1847.txt	poem	1847	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Aranyaimhoz_1847.txt	poem	1847	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_TudosMacskaja_1847.txt	poem	1847	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_FalusiMulatsag_1847.txt	poem	1847	9	AJ
c:\keysoft\samples\ajps\aj\	AJ_Emleklapra_1850.txt	poem	1850	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_FurkoTamas_1850.txt	poem	1850	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_EvekMegJovendo_1850.txt	poem	1850	2	AJ
c:\keysoft\samples\ajps\aj\	AJ_Koldusenek_1850.txt	poem	1850	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_EvUtoljan_1852.txt	poem	1852	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_PoetaiRecept_1852.txt	poem	1852	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_KisPokol_1852.txt	poem	1852	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Rozgonyine_1852.txt	poem	1852	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_AlkalmiVers_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_ArkadiaFele_1853.txt	poem	1853	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_UjeviKoszontes_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_TorokBalint_1853.txt	poem	1853	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_Vigasztalo_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Vagy_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Sarkany_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Hegedu_1853.txt	poem	1853	4	AJ
c:\keysoft\samples\ajps\aj\	AJ_BaroKemeny_1865.txt	poem	1865	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_ArtatlanDac_1865.txt	poem	1865	12	AJ
c:\keysoft\samples\ajps\aj\	AJ_Leanyomhoz_1865.txt	poem	1865	12	AJ
c:\keysoft\samples\ajps\aj\	AJ_CsillagHullaskor_1867.txt	poem	1867	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Lepke_1877.txt	poem	1877	7	AJ
c:\keysoft\samples\ajps\aj\	AJ_Epilogus_1877.txt	poem	1877	7	AJ
c:\keysoft\samples\ajps\aj\	AJ_Vasarban_1877.txt	poem	1877	7	AJ
c:\keysoft\samples\ajps\aj\	AJ_TamburasOreg_1877.txt	poem	1877	7	AJ
c:\keysoft\samples\ajps\aj\	AJ_Reggel_1881.txt	poem	1881	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_JosagosOzvegynek_1880.txt	poem	1880	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_ToldiIIReszehez_1879.txt	poem	1879	8	AJ
c:\keysoft\samples\ajps\aj\	AJ_HarmincEv_1879.txt	poem	1879	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_TolgyekAlatt_1878.txt	poem	1878	5	AJ
c:\keysoft\samples\ajps\aj\	AJ_Elegia_1839.txt	poem	1839	3	AJ
c:\keysoft\samples\ajps\aj\	AJ_FeledFeled_1840.txt	poem	1840	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_VersesBiralatok_1878.txt	poem	1878	1	AJ
c:\keysoft\samples\ajps\aj\	AJ_AlkalmatossagraIrott_1856.txt	poem	1856	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_WalesiBardok_1857.txt	poem	1857	6	AJ
c:\keysoft\samples\ajps\aj\	AJ_ToldiElso_1846.txt	poem	1846	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_DaliasIdok2_1854.txt	poem	1854	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_ElveszettAlkotmany1_1845.txt	poem	1845	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_RozsaIbolyaII_1853.txt	poem	1853	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_RozsaIbolya_1847.txt	poem	1847	6.5	AJ
c:\keysoft\samples\ajps\aj\	AJ_UtolsoMagyar_1858.txt	poem	1858	6.5	AJ

```
c:\keysoft\samples\ajps\aj\   AJ_CsabaTrilogiaUtolso_1881.txt      poem   1881   6.5    AJ
c:\keysoft\samples\ajps\ps\   PS_Borozo_1842.txt     poem   1842   4      PS
c:\keysoft\samples\ajps\ps\   PS_KetVandor_1842.txt  poem   1842   4      PS
c:\keysoft\samples\ajps\ps\   PS_FelkoSzontes_1842.txt      poem   1842   11     PS
c:\keysoft\samples\ajps\ps\   PS_FurcsaTortenet_1842.txt    poem   1842   10     PS
c:\keysoft\samples\ajps\ps\   PS_Bucsuzas_1838.txt   poem   1838   6      PS
c:\keysoft\samples\ajps\ps\   PS_Elegia_1841.txt     poem   1841   6      PS
c:\keysoft\samples\ajps\ps\   PS_Palnapkor_1843.txt  poem   1843   1      PS
c:\keysoft\samples\ajps\ps\   PS_Dinomdanom_1843.txt poem   1843   1      PS
c:\keysoft\samples\ajps\ps\   PS_SzegetSzeggel_1843.txt     poem   1843   4      PS
c:\keysoft\samples\ajps\ps\   PS_UtosoAlamizsna_1843.txt    poem   1843   4      PS
c:\keysoft\samples\ajps\ps\   PS_Honfidal_1844.txt   poem   1844   2      PS
c:\keysoft\samples\ajps\ps\   PS_Vegszohoz_1844.txt  poem   1844   2      PS
c:\keysoft\samples\ajps\ps\   PS_Magany_1844.txt     poem   1844   4      PS
c:\keysoft\samples\ajps\ps\   PS_CsaplarneBetyart_1844.txt  poem   1844   4      PS
c:\keysoft\samples\ajps\ps\   PS_JanosVitez02_1844.txt      poem   1844   11     PS
c:\keysoft\samples\ajps\ps\   PS_JanosVitez25_1844.txt      poem   1844   12     PS
c:\keysoft\samples\ajps\ps\   PS_MagyarNemzet_1845.txt      poem   1845   1      PS
c:\keysoft\samples\ajps\ps\   PS_KetTestver_1845.txt poem   1845   1      PS
c:\keysoft\samples\ajps\ps\   PS_MidonNagyon_1845.txt       poem   1845   2      PS
c:\keysoft\samples\ajps\ps\   PS_KeketMutatnak_1845.txt     poem   1845   2      PS
c:\keysoft\samples\ajps\ps\   PS_IstenCsodaja_1846.txt      poem   1846   1      PS
c:\keysoft\samples\ajps\ps\   PS_Orult_1846.txt      poem   1846   1      PS
c:\keysoft\samples\ajps\ps\   PS_VajdaPeter_1846.txt poem   1846   2      PS
c:\keysoft\samples\ajps\ps\   PS_Tunderalom_1846.txt poem   1846   2      PS
c:\keysoft\samples\ajps\ps\   PS_FerfiVagy_1847.txt  poem   1847   1      PS
c:\keysoft\samples\ajps\ps\   PS_Kutyakaparo_1847.txt       poem   1847   1      PS
c:\keysoft\samples\ajps\ps\   PS_SzilveszterEje_1847.txt    poem   1847   12     PS
c:\keysoft\samples\ajps\ps\   PS_KinnMenes_1847.txt  poem   1847   12     PS
c:\keysoft\samples\ajps\ps\   PS_TeliEstek_1848.txt  poem   1848   1      PS
c:\keysoft\samples\ajps\ps\   PS_FelesegekFelesege_1848.txt poem   1848   1      PS
c:\keysoft\samples\ajps\ps\   PS_EvVegen_1848.txt    poem   1848   12     PS
c:\keysoft\samples\ajps\ps\   PS_VesztettCsatak_1848.txt    poem   1848   12     PS
c:\keysoft\samples\ajps\ps\   PS_NemzetiDal_1848.txt poem   1848   3      PS
c:\keysoft\samples\ajps\ps\   PS_15dikMarcius_1848.txt      poem   1848   3      PS
c:\keysoft\samples\ajps\ps\   PS_UjevNapjan_1849.txt poem   1849   1      PS
c:\keysoft\samples\ajps\ps\   PS_BudaVaran_1849.txt  poem   1849   1      PS
c:\keysoft\samples\ajps\ps\   PS_SzornyuIdo_1849.txt poem   1849   7      PS
c:\keysoft\samples\ajps\ps\   PS_SzentHaborura_1849.txt     poem   1849   6      PS
c:\keysoft\samples\ajps\ps\   PS_Tunderkaland_1847.txt      poem   1847   6      PS
c:\keysoft\samples\ajps\ps\   PS_Hazamban_1842.txt   poem   1842   10     PS
```

Of course, the point of metafiles is that they can be edited, so there is no need to stick to this particular selection.

Appendix 2 : Parameter Files

Parameters used by **minimet4**.py. Normally only the first 6 items in this table need to be specified by a user.

Parameter	Default value	Function
comment	None	This (or in fact any unrecognized parameter name, e.g. "##") can be used to insert reminders about what the file is meant to do.
corpath	[None]	Specification of directory where files to be included in metafile reside.
jobname	minimeta	This gives the job a name. Any text string can be used.
metazero	[None]	Full path/file specification of output metafile.
targname	doctype	Name to be given to target column.
targval	00	Initial value to be given to the target column (normally a class label).
outfile	minimeta.txt	File where logging information will be written. (Really only needed for debugging.)
outpath	[subfolder "op"	Directory where logging file will be written.

	of parent directory]	

Parameters used by **newquay3**.py.

Parameter	Default value	Function
atomize	1	This can be zero or 1. If it is 1, the input texts are tokenized by the program's built-in tokenizer. Only set this to zero if your files have already been tokenized, in which case whitespace will be considered to delimit tokens.
casefold	1	This can be 0 or 1. Zero means that upper and lower case is left as found on input; 1 means that input texts will have all letters forced into lower case. (No effect on character sets without upper/lower case distinction.)
comment	[None]	This (or in fact any unrecognized parameter name, e.g. "##") can be used to insert reminders about what the file is meant to do.
jobname	newquay	This gives the job a name. Any text string can be the value. It isn't necessary but it is useful as the jobname will be used as a prefix to the program's output files, so it can be seen that they form a group.
keymode	g2	Short name (no more than four characters) of the keyness measure selected. (See section on keyness indicators, above.)
metafile	[None]	This should be the full path specification of a metafile that indicates the text files that belong to the input corpus, along with their category memberships.
mindocf	3	No term that does not occur in at least this many documents will be considered as a keyword.
snipsize	115	This gives the size of a text block, in tokens, to be used when calculating keyness using keymode neoz or pq.
targvar	doctype	This should be the name of the column in the metafile containing the category labels.
topkeys	32	This specifies how many of the highest-scoring keys should be written to the keyfile. Both positive and negative keys will be included, so twice this number will be listed, for each category.
wordonly	0	This should be integer 0 or 1. If it is 1, the tokenizer will ignore input tokens unless they begin with an alphanumeric character. If it is zero, all tokens will be considered, even sequences of punctuation symbols and so on. Unless you're sure the punctuation is original, it is advisable to set this parameter to 1.
keyfile	jobname with "_keys.txt" appended	The program writes the keyword listing to this file. You can send this to a specific named file if you don't want to use the default name.
listfile	jobname with "_pars.txt" appended	You can give a specific filename on which a summary of parameter settings will be written if you don't want it to have the default name.
outfile	newquay.txt (on outpath)	File where logging information will be written. (Really only needed for debugging.)
outpath	subfolder "op" of parent directory	You can send the output to a specified directory if you like.

Parameters additional to the above used by **termite3**.py

Parameter	Default value	Function
termfile	jobname with "_term.txt" appended	Full filepath name onto which desmoglyphs will be written.
sidespan	5	maximum number of tokens either side of the seed item to be considered in searching for subsequences.

N.B. At present, if you have more than 1 blank lines in a parameter file, the input routine chokes. I do plan to fix this at some stage, but, in the mean time, it is quite easy to delete empty lines using a text editor.