FORMULIB : Formulaic Language Software Library

(User Notes by Richard Forsyth, June 2016)

This software helps to explore the somewhat nebulous concept of "formulaic language" as well as identifying relatively typical or anomalous texts within given text types. There are four programs in the suite: outgrams.py, which finds commonly occurring n-grams in a selection of texts (where n can range between 1 and 9 but is typically 2 to 6); formulex.py, which calculates, among other things, an index of formulaic coverage both in individual documents and in text categories; flicshow.py which produces html files that highlight the token sequences actually covered by the n-grams found by outgrams.py; and taverns.py which ranks individual texts according to how completely the n-grams found from each category of text cover texts from the same and from other categories. The last program not only indicates which texts are typical or atypical of their type but can be used in text-classification mode to allocate documents to their most compatible categories, with an indication of the strength of each such allocation.

The programs are written in Python3 and made available under the GNU public licence for general usage.

Why I Wrote this Software

Over the past three years I have been drawn into investigating concept of "formulaic language" (Wray, 2002). This notion means subtly different things to different linguists, so I won't attempt to provide a definition. Nevertheless, there does seem to be a core notion common to most interpretations of the term, namely that formulaic discourse relies more heavily on relatively fixed, apparently prefabricated, sequences of language elements than the more creative uses of language emphasized by linguists such as Chomsky (1972). After all, it is obvious to anyone whose eyes glaze over trying to read the legalistic fine print in a consumer contract that some texts are much more repetitive than others.

The question then arises whether it is possible to measure the amount of formulaic language in a text or group of texts. Strictly speaking, without a definition of formulaic language, this isn't possible. However, repetitions of various kinds can be counted, and since repetitiveness is at the heart of this matter, it seems plausible that a measure based on frequently recurring token sequences could have practical value as an indicator. It could be highly effective in picking up the tell-tale signs produced in real texts by reliance on formulaic phrasings. Thus it could provide a useful tool for those wanting to separate, so to speak, the idiom-principle sheep from the open-choice goats in various collections of documents.

The programs described here implement this idea to perform four related functions:

- **outgrams.py** compiles the components of a 'formulexicon' by finding the most frequent n1-grams up to n2-grams in 2 or more groups of text files where n1 is 2 and n2 is 5 by default;
- **formulex.py** computes what proportion of each document (and hence each text category) is covered by the elements of this formulexicon, as well as producing a list of 'collocades' (cascades of collocations, to be explained below);
- **flicshow.py** (Formulaic Language In Context) allows a user to select a number of text files and see precisely which sequences are covered by the elements of the formulexicon;
- **taverns.py** computes coverage of 2 or more categories of document not only by the collocades generated from their own category but by those of the other categories, thus identifying highly typical and highly atypical texts in each class, and performs a classification as well.

(The programs of the keysoft suite are also provided in the distribution. See keysoft.pdf for details. Some day I may have time to integrate these two related software suites better.)

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 formulib.zip file. After unpacking it (into a folder called "formulib", preferably at the C:\ level in Windows unless you want to do quite a lot of editing), you should find the following subfolders.

mets
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. Subfolder mets is the default location for metafiles which will be explained below.

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 seven subfolders containing files from seven different text sources. These contain datasets that enable you to start using the system, prior to collecting &/or reformatting your own corpora. (More details in Appendix 3.)

Overall Outline

As mentioned above, the software comprises four main programs with complementary functions which would normally be executed in sequence.

Program	Input metafile(s)	Main outputs
outgrams.py	metafile / trainmet ['training set']	1. _gram.txt n-grams ordered by size then frequency for each category of text; 2. _list.txt n-grams ordered by frequency only (with various sizes intermingled) for each category of text.
formulex.py	trainmet / metafile (testmeta) ['test set']	1. _food.txt (Formulaic Ordering Of Documents) 2. _flab.txt (Frequent Lexically Assembled Bundles)
flicshow.py	metaflic	colour-coded html files in a separate directory for viewing with a browser.
taverns.py	gramdata testmeta	1. _ales.txt (Affinity Listing Exploiting Sequences) 2. _blox.txt (Basic List Of Covering Sequences)

The distribution also includes a couple of utility programs, metaget.py and randmet.py, which are described below.

Making a Metafile

Below is a complete listing of a metafile relating to wine portion the bottlabs corpus. 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:\formulib\samples\bottlabs\wine\	2011_old_man_creek.txt	wine
C:\formulib\samples\bottlabs\wine\	alta_vista_premium_2011.txt	wine
C:\formulib\samples\bottlabs\wine\	arniston_bay_coast.txt	wine
C:\formulib\samples\bottlabs\wine\	ashgrove_malbec.txt	wine
C:\formulib\samples\bottlabs\wine\	black_label_red.txt	wine
C:\formulib\samples\bottlabs\wine\	campo_viejo_rioja.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_chianti_2013.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_chilean_merlot.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_cotes_du_rhone_2014.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_explorers_vineyard.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_fairtrade_chenin_blanc.txt	wine
C:\formulib\samples\bottlabs\wine\	coop_shiraz_rose.txt	wine
C:\formulib\samples\bottlabs\wine\	crooked_creek_vineyards_lenoir_2011.txt
	wine
C:\formulib\samples\bottlabs\wine\	fabcab.txt	wine
C:\formulib\samples\bottlabs\wine\	fairtrade_cabernet_sauvignon.txt	wine
C:\formulib\samples\bottlabs\wine\	fairtrade_cinsault_shiraz_sa.txt	wine
C:\formulib\samples\bottlabs\wine\	ferreira_port_ruby.txt	wine
C:\formulib\samples\bottlabs\wine\	ferreira_port_tawny.txt	wine
C:\formulib\samples\bottlabs\wine\	finca_carelio.txt	wine
C:\formulib\samples\bottlabs\wine\	finca_las_moras_shiraz.txt	wine
C:\formulib\samples\bottlabs\wine\	hardys_privatebin_shiraz.txt	wine
C:\formulib\samples\bottlabs\wine\	hardys_stamp_of_australia.txt	wine
C:\formulib\samples\bottlabs\wine\	hardys_voyage_2015.txt	wine
C:\formulib\samples\bottlabs\wine\	inycon_growers_choice.txt	wine
C:\formulib\samples\bottlabs\wine\	kissing_tree_zinfandel.txt	wine
C:\formulib\samples\bottlabs\wine\	kumala_2013.txt	wine
C:\formulib\samples\bottlabs\wine\	la_chiave_2013.txt	wine
C:\formulib\samples\bottlabs\wine\	la_consulta.txt	wine
C:\formulib\samples\bottlabs\wine\	la_paz_merlot.txt	wine
C:\formulib\samples\bottlabs\wine\	le_provenance_cotes_de_provence.txt
	wine
C:\formulib\samples\bottlabs\wine\	lime_tree_cabernet_sauvignon.txt	wine
C:\formulib\samples\bottlabs\wine\	long_slim_chile.txt	wine
C:\formulib\samples\bottlabs\wine\	merlot_kekfrankos_2014.txt	wine
C:\formulib\samples\bottlabs\wine\	mondelli_montepulciano.txt	wine
C:\formulib\samples\bottlabs\wine\	ms_corte_ibla.txt	wine
C:\formulib\samples\bottlabs\wine\	ms_crozes_hermitage.txt	wine
C:\formulib\samples\bottlabs\wine\	nero_davola_syrah.txt	wine
C:\formulib\samples\bottlabs\wine\	one_tree_lake_shiraz.txt	wine
C:\formulib\samples\bottlabs\wine\	orvieto_classico_2014.txt	wine
C:\formulib\samples\bottlabs\wine\	oxford_landing_2011_merlot.txt	wine
C:\formulib\samples\bottlabs\wine\	oxford_landing_2011_sauvignon_blanc.txt
	wine
C:\formulib\samples\bottlabs\wine\	oxford_landing_estates_2012_merlot.txt
	wine
C:\formulib\samples\bottlabs\wine\	paris_street.txt	wine
C:\formulib\samples\bottlabs\wine\	redtree_california_2010.txt	wine
C:\formulib\samples\bottlabs\wine\	salice_salentino_2012.txt	wine
C:\formulib\samples\bottlabs\wine\	terras_de_alleu.txt	wine
C:\formulib\samples\bottlabs\wine\	three_mills_mini_rose.txt	wine
C:\formulib\samples\bottlabs\wine\	two_ravens_2013.txt	wine
C:\formulib\samples\bottlabs\wine\	valpolicella_ripasso_superiore.txt
	wine
```

```
C:\formulib\samples\bottlabs\wine\      vanderburg_shirazpinotage_2013.txt
      wine
C:\formulib\samples\bottlabs\wine\      via_vecchio.txt     wine
```

This metafile describes a small corpus of 51 "back label" texts from wine bottles that I have been collecting over the past year. It cannot claim to be a statistically representative sample of what I drink, still less of what is written on back labels worldwide, and all information about colouring, pictorial images, placement, type fonts and such like has been lost. Moreover, I suspect I have been somewhat inconsistent about which bits of non-English text that I have been prepared to include, and it would be possible to argue about some of the choices made in reducing each text from 2 dimensions to a single linear sequence. Still, it is a nice accessible sample which definitely contains a high proportion of "boilerplate" language required by legal regulations as well as some classic oenological clichés. (More info in Appendix 3.)

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 an initial metafile, it is convenient to use the metaget.py program, which is included with the distribution. The output of this program can then be edited in a text-editor, 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.

The metaget.py program can be run just by double-clicking on its name. It will then display a window with four labelled entry boxes:

> 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 &/or different folders. Then you provide a destination file name and extension 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 its precursor 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.)

The mets subfolder contains several metafiles that you can inspect as examples before making your own. Many of these come in groups of three, such as the trio below.

ares_1.txt
ares_2.txt
aresmeta.txt

In this group aresmeta.txt is a metafile for all the UN General Assembly resolutions held in subfolder ares (700 of them). Metafiles ares_1.txt and ares_2.txt contain the names of files forming randomly chosen disjoint subsets of the whole ares collection. Triplets of metafiles for the other 6 text types that follow the same pattern are also provided.

Preparing a Parameter File
Below is a listing of parameter file formtest.txt which comes with the formulib distribution.

```
comment  formulib testing :
jobname  formtest
metafile  c:\formulib\mets\mainmeta.txt
outmeta1  c:\formulib\mets\training.txt
outmeta2  c:\formulib\mets\testing.txt
randfrac  0.61803398875
trainmet  c:\formulib\mets\training.txt
testmeta  c:\formulib\mets\testing.txt
metaflic  c:\formulib\mets\testing.txt
wordonly  1
maxtops  20
topgrams  80
outforms  80
miniglen  3
maxiglen  6
##outpath  c:\fout\
topkeys  64
```

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. Again, the simplest way to make a parameter file is by using Notepad++ or a similar text editor. 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	[Name of program being executed]	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.
outmeta1	[None]	Specification of a file into which a random subset of items in the input metafile will be writen. (Training example.)
outmeta2	[None]	Specification of a file into which a second random subset of items in the input metafile will be writen. (Testing examples.)
randfrac	0.5	Proportion of items in metafile to be written into outmeta1.
trainmet	[None]	This should be the full file specification of a metafile that indicates the text files that belong to the 'training' corpus, each associated with its target variable, i.e. class label, as described above.
testmeta	[None]	This is optional; if present it should be the full file specification of a metafile that specifies a holdout sample of files. The idea behind this is to check how well programs formulex.py and taverns.py perform

		on data not used by outgrams.py to generate the original list of n-grams.
metaflic	[None]	This is optional; if present it should be the full file specification of the metafile that specifies which files are to be processed by flicshow.py to produce html files highlighting formulaic collocades.
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.
maxtops	20	This is used to specify the number of 'stop words' considered by outgrams.py. It can be 0, but if this number is greater than zero, outgrams.py will compile a list of high-frequency stop-words (method to be explained later). These will be listed in the _list.txt file so you know what they are. Their only effect on the processing is that an n-gram that consists solely of stop-words will not be included in the _gram.txt file, thus won't be used by subsequent programs. In practice, this only makes more than a trivial difference if the smallest n-gram size is 1 (default is 2 but it can be changed).
topgrams	100	This number specifies how many of the most frequent n-grams in each text category will be retained for each chosen value of n. Thus with topgrams equal to 80 and the default n-gram range of 2 to 5, 320 n-grams will be generated, 80 2-grams, 80 3-grams, 80 4-grams and 80 5-grams. (With small corpora this could be fewer, since n-grams occurring only once or twice will be suppressed.)
outforms	100	This applies to the formulex.py and taverns.py programs and governs how many collocades will be listed.
miniglen	2	This specifies the smallest n-gram size to be used.
maxiglen	5	This specifies the largest n-gram size to be used.

The line beginning with "##outpath" is a commented-out specification of a nonstandard folder to receive output. By deleting the 2 hash-signs at the front it could be re-activated. I tend to keep several such lines in a parameter file in order to try different options by commenting them in or out.

The line
topkeys 64
is applicable to the keysoft programs (see keynotes.pdf). I have tried to make it so that a single parameter file can apply to a whole suite of programs, with each program only reading in the parameter options that apply to it.

Running randmet.py

Lines 3 to 6 of the parameter file, above, actually apply to the randmet.py program. This is a simple program that simply splits an input metafile randomly into 2 disjoint output metafiles, which are intended to be used as 'training' and 'testing' subsamples. The train/test division is an idea from the field of statistical machine-learning. In the present context it means that n-grams derived from one subsample of a corpus would normally be applied to texts in another subsample (as well as the training sample). This helps to give an idea of whether the figures computed are likely to be stable, i.e. similar when applied to previously unseen texts of the same types.

Thus it is normal to start an analysis by creating a metafile describing an entire text collection and use randmet.py to produce 2 output metafiles which describe disjoint subsets of that collection.

Running outgrams.py

This is a fairly standard n-gram finder, which is run to provide data for the other three programs of the suite. It can be executed by double-clicking, but it would be more usual to run it from a command window by typing a command such as the following.

c:\formulib\p3>**python outgrams.py**

It will then ask for a parameter file. (See above.) If you reply with formtest.txt you can check how it works with the samples provided. You should then see on screen something like the listing below.

```
C:\formulib\p3\outgrams.py 1.3 Tue May 31 16:46:24 2016
command-line args. = 1
prepath : C:\formulib\p3
working folder:  C:\formulib\p3
script usage:  python C:\formulib\p3\outgrams.py <parafile>
please give parameter file name : formtest
Paths to search for parameter file :
['C:\\formulib\\parapath',   'C:\\formulib\\p3',   '..',   '.',   'C:\\Users\\Richard.lounge-
pc\\parapath', 'C:\\Users\\Richard.lounge-pc']
 formtest
trying to open : C:\formulib\parapath\formtest.txt
C:\formulib\parapath\formtest.txt opened for reading.
c:\formulib\mets\training.txt to be used to indicate training texts.
['prepath', 'filename', 'doctype']
1953
target column name : doctype @ 2
Text types : {'wine', 'sres', 'leaflet', 'beer', 'wc', 'ares', 'tedtalk'}
Number of texts = 1953
Number of tokens= 3215180
Shortest = 56 tokens.
median size = 1310
mean size = 1646.28
Longest = 21954 tokens.
ares 522805
beer 4196
leaflet 305939
sres 134809
tedtalk 2150340
wc 93239
wine 3852
reference category :  tedtalk
ares 409
beer 34
leaflet 289
sres 163
tedtalk 998
wc 31
wine 29
snippets = 27724
ares
beer
leaflet
sres
tedtalk
wc
wine
gram listing written on file C:\formulib\op\formtest_list.txt
gram dump written onto file C:\formulib\op\formtest_gram.dat
C:\formulib\p3\outgrams.py done on Tue May 31 16:47:25 2016
after 53.70358 seconds.
```

Most of this screen output is merely to reassure a user that the program is doing something. The outputs that matter more are formtest_gram.txt and formtest_list.txt. A short edited extract from formtest_gram.txt is listed below. This shows the first 16 and the last 10 entries for category ares (UN General Assembly resolutions) followed by the first 13 entries (6-grams) for category beer (beer bottle back-label texts).

```
# ares 522805 3346191
```

```
1 (6, 394, 42, ('resolution', 'adopted', 'by', 'the', 'general', 'assembly'))
2 (6, 356, 35, ('the', 'report', 'of', 'the', 'secretary', 'general'))
3 (6, 309, 34, ('adopted', 'by', 'the', 'general', 'assembly', 'on'))
4 (6, 303, 34, ('the', 'general', 'assembly', 'on', 'the', 'report'))
5 (6, 303, 33, ('general', 'assembly', 'on', 'the', 'report', 'of'))
6 (6, 303, 30, ('by', 'the', 'general', 'assembly', 'on', 'the'))
7 (6, 303, 29, ('assembly', 'on', 'the', 'report', 'of', 'the'))
8 (6, 297, 33, ('the', 'general', 'assembly', 'at', 'its', 'fifty'))
9 (6, 282, 30, ('to', 'the', 'general', 'assembly', 'at', 'its'))
10 (6, 223, 33, ('the', 'charter', 'of', 'the', 'united', 'nations'))
11 (6, 199, 31, ('of', 'the', 'secretary', 'general', 'on', 'the'))
12 (6, 193, 34, ('report', 'of', 'the', 'secretary', 'general', 'on'))
13 (6, 193, 26, ('for', 'the', 'period', 'from', '1', 'july'))
14 (6, 187, 30, ('of', 'the', 'report', 'of', 'the', 'secretary'))
15 (6, 183, 40, ('official', 'records', 'of', 'the', 'general', 'assembly'))
16 (6, 172, 36, ('to', 'include', 'in', 'the', 'provisional', 'agenda'))

[.... many lines omitted to save space ....]

71 (3, 215, 16, ('the', 'economic', 'and'))
72 (3, 215, 14, ('general', 'on', 'the'))
73 (3, 214, 17, ('convention', 'on', 'the'))
74 (3, 214, 11, ('with', 'a', 'view'))
75 (3, 212, 17, ('to', 'the', 'convention'))
76 (3, 212, 14, ('in', 'this', 'regard'))
77 (3, 211, 9, ('a', 'view', 'to'))
78 (3, 210, 22, ('assembly', 'recalling', 'its'))
79 (3, 210, 14, ('of', 'the', 'present'))
80 (3, 209, 18, ('and', 'social', 'council'))
# beer 4196 25164
1 (6, 12, 35, ('avoid', 'alcohol', 'if', 'pregnant', 'or', 'trying'))
2 (6, 12, 33, ('if', 'pregnant', 'or', 'trying', 'to', 'conceive'))
3 (6, 12, 32, ('alcohol', 'if', 'pregnant', 'or', 'trying', 'to'))
4 (6, 8, 40, ('recommend', 'adults', 'do', 'not', 'regularly', 'exceed'))
5 (6, 8, 34, ('regularly', 'exceed', 'men', '4', 'units', 'daily'))
6 (6, 8, 34, ('adults', 'do', 'not', 'regularly', 'exceed', 'men'))
7 (6, 8, 32, ('not', 'regularly', 'exceed', 'men', '4', 'units'))
8 (6, 8, 31, ('units', 'daily', 'women', '3', 'units', 'daily'))
9 (6, 8, 30, ('exceed', 'men', '4', 'units', 'daily', 'women'))
10 (6, 8, 29, ('do', 'not', 'regularly', 'exceed', 'men', '4'))
11 (6, 8, 27, ('4', 'units', 'daily', 'women', '3', 'units'))
12 (6, 8, 25, ('men', '4', 'units', 'daily', 'women', '3'))
13 (6, 7, 42, ('uk', 'chief', 'medical', 'officers', 'recommend', 'adults'))

[.... even more lines omitted ....]
```

The first line, like other lines beginning with a hash sign ('#'), signals the start of data for a new category of texts. The three items that follow the hash sign are the name of the category, the number of tokens and the number of characters in that category. Here we see that category ares (UN General Assembly resolutions) contains 522805 tokens (words or numbers) comprising 3346191 characters. The beer sample is much smaller: it contains 4196 tokens totalling 25164 characters. It should be noted that these character counts are made after tokenization and include a single space between each token.

The lines shown after the header lines contain the n-grams themselves. We requested 80 of each size and used the sizes of 3 (shortest) and 6 (longest). Line 1 thus contains the most frequent 6-gram

"resolution adopted by the general assembly"

which occurs 394 times in the 409 files given as input and contains 42 characters (including the spaces between words).

The data from ares goes down to the 80th most frequent 3-gram ("and social council") which occurs 209 times. After that begins the data from the beer subsection of the bottlab files. Here the 2 most frequent 6-grams are

"avoid alcohol if pregnant or trying"

and

"if pregnant or trying to conceive"

both of which occur 12 times.

This pair of items hints at one of the main problems with simple n-gram lists, namely that many n-grams of a fixed size are obviously fragments of a longer sequence. In this case the repeated phrase is happens to be an 8-gram

"avoid alcohol if pregnant or trying to conceive"

fragments of which will be found at n-gram sizes from 6 down to 3. (An essential objective of formulex.py and the subsequent programs is to deal with exactly this problem, but description of that will be postponed till the next section.)

This _gram.txt file is meant to be re-read easily by subsequent programs, so it isn't in a particularly convenient format for humans. However, it is useful to be able to interpret its contents in case you want to check the results on your data before proceeding further.

Slightly more readable is the _list.txt output file. Part of formtest_list.txt, generated by the example above, is listed below.

```
Tue May 31 16:46:24 2016
parafile: C:\formulib\parapath\formtest.txt
metafile: c:\formulib\mets\training.txt
miniglen: 3
maxiglen: 6
topgrams: 80
dropsubs: 0
maxtops : 20
1953 7

Stop-word listing :
1 27413 the 98.88
2 26501 and 95.59
3 26093 of 94.12
4 26004 to 93.8
5 23680 in 85.41
6 23285 a 83.99
7 20499 that 73.94
8 17506 is 63.14
9 15803 this 57.0
10 15219 it 54.89
11 15171 for 54.72
12 14809 you 53.42
13 13609 on 49.09
14 13000 with 46.89
15 12277 i 44.28
16 12192 so 43.98
17 11337 have 40.89
18 11256 are 40.6
19 11207 we 40.42
20 10506 as 37.89
```

```
n-grams by category :

ares 522805 tokens
1 3377 3 18 the united nations
2 2014 3 20 the general assembly
3 1989 3 21 the secretary general
4 1884 3 13 of the united
5 1873 4 21 of the united nations
6 1178 3 13 report of the
7 956 3 13 the report of
8 953 4 17 the report of the
9 885 4 24 the secretary general to
10 885 3 20 secretary general to
11 764 3 21 implementation of the
12 762 3 18 in accordance with
13 760 4 30 requests the secretary general
14 760 3 22 requests the secretary
15 738 3 21 the implementation of
16 690 3 10 as well as
17 635 3 14 by the general
18 629 5 33 requests the secretary general to
19 627 3 14 adopted by the
20 615 4 23 by the general assembly
21 589 3 16 of the secretary
22 585 4 24 of the secretary general
23 585 3 21 united nations system
24 576 4 25 the united nations system

[.... many lines omitted ....]
```

The first block of this listing gives a date-line and some parameter information. Miniglen and maxiglen are the minimum and maximum n-gram lengths (2 and 5 by default).

The next block lists the stop-words used, 20 having been requested. These are compiled by taking the whole input corpus and subdividing it into fixed-length chunks that I call 'snippets', with a default snippet-size of 115 tokens. Then the percentage of snippets in which it occurs is found for each token, and the tokens are sorted in descending order of this percentage. The first maxtops (here 20) of these are then taken as stop-words. In the subsequent listing any n-gram consisting entirely of stop-words will be ignored; otherwise they have no further effect, although they are listed for reference. From this listing we can see that 'the' occurs in more than 98 percent of all snippets. At position 20 'as' occurs in 37.89 percent of all snippets. In this case all 20 items are high-frequency function words, common in most kinds of English. Of course a different selection of texts would probably give a somewhat different collection of stop-words.

The next blocks show the actual n-grams for each category. Here the ordering is simply by frequency, so n-grams of various sizes are intermixed. To save space only the first 24 lines from the ares sample are shown here. From these we see that the first triplet "the united nations" (18 characters in length) occurs 3377 times in this category. At positions 7 and 8 we find that the 3-gram "the report of" occurs 956 times while the 4-gram "the report of the" occurs 953 times, which implies that only thrice out of 956 occasions did anything other than "the" follow "the report of" in this corpus. Similarly, lines 9 and 10 show that the 3-gram "secretary general to" is always preceded by "the".

Again, multiple fragments of what are presumably longer fixed phrases occur at various points, so this listing, although it does convey a general impression of the key topics in each category, gives only limited insight into how pervasive fixed formulaic sequences are in the texts concerned. That is the task of the next program, formulex.py.

Running formulex.py

When you execute this program, you should see something like the following output on screen.

```
C:\formulib\p3\formulex.py 1.5 Tue May 31 17:18:59 2016
command-line args. = 1
prepath : C:\formulib\p3
working folder:  C:\formulib\p3
script usage:  python C:\formulib\p3\formulex.py <parafile>
please give parameter file name : formtest
Paths to search for parameter file :
['C:\\formulib\\parapath',          'C:\\formulib\\p3',          '..',          '.',
'C:\\Users\\Richard.lounge-pc\\parapath', 'C:\\Users\\Richard.lounge-pc']
 formtest
trying to open : C:\formulib\parapath\formtest.txt
C:\formulib\parapath\formtest.txt opened for reading.
c:\formulib\mets\training.txt to be used to indicate training texts.
['prepath', 'filename', 'doctype']
1953
target column name : doctype @ 2
Text types : {'ares', 'tedtalk', 'beer', 'sres', 'wine', 'leaflet', 'wc'}
Number of texts = 1953
Number of tokens= 3215180
Shortest = 56 tokens.
median size = 1310
mean size = 1646.28
Longest = 21954 tokens.
ares 522805
beer 4196
leaflet 305939
sres 134809
tedtalk 2150340
wc 93239
wine 3852
reference category :  tedtalk
ares 409
beer 34
leaflet 289
sres 163
tedtalk 998
wc 31
wine 29
snippets = 27724
N-grams to be read from C:\formulib\op\formtest_gram.dat :
2157
2150 n-grams read from C:\formulib\op\formtest_gram.dat
['prepath', 'filename', 'doctype']
1206
target column name : doctype @ 2
Number of test texts = 1206
Number of test tokens= 1989267
listing written onto file C:\formulib\op\formtest_food.txt
formdump written onto file C:\formulib\op\formtest_flab.dat
C:\formulib\p3\formulex.py done on Tue May 31 17:20:24 2016
after 81.28188 seconds.
```

Most of this is of only passing interest, but it is worth pointing out that this program has three main input sources. One is the set of texts files specified by the trainmet parameter, which is normally the same set as used to generate the n-gram file. The second is an n-gram file generated by outgrams.py, normally from those texts specified in the metafile parameter. The third is optional, but is present in this case as those texts specified in testing.txt, the value of the testmeta parameter.

The most important output from this run is the file formtest_food.txt, part of which is reproduced below.

```
Tue May 31 17:18:59 2016
parafile: C:\formulib\parapath\formtest.txt
metafile: c:\formulib\mets\training.txt
miniglen: 3
maxiglen: 6
topgrams: 80
1953 7

Category coverage% (characters, tokens) by frequent n-grams :

0 ares              15.7643   16.8610
1 beer              29.3236   29.1230
2 leaflet           13.0373   14.6248
3 sres              18.2463   19.1797
4 tedtalk            3.6273    4.7620
5 wc                 5.0934    6.0962
6 wine              18.8720   18.7175

Document coverage% (characters, tokens) by frequent n-grams :

   1    983   160   60.87  60.00  beer         saltaire_blonde.txt
   2    995   165   59.84  58.18  beer         saltaire_cascade.txt
   3   1009   166   58.61  57.23  beer         saltaire_pride.txt
   4   1011   166   58.40  54.82  ares         A_RES_56_290-en.txt
   5   1042   164   57.43  57.93  ares         A_RES_57_329-en.txt
   6    606   103   49.42  49.51  wine         fabcab.txt
   7    526    85   47.25  49.41  beer         low_alcohol_czech_lager.txt
   8    568    90   44.99  45.56  beer         marstons_burton_bitter.txt
   9   1030   165   44.81  48.48  sres         S_RES_14892003-en.txt
  10    913   163   44.31  39.88  ares         A_RES_56_222-en.txt
  11    681   116   43.99  43.10  sres         S_RES_13802001-en.txt
  12    543    90   43.93  44.44  beer         corona_extra.txt
  13   1761   290   43.25  45.17  sres         S_RES_15552004-en.txt
  14    525    96   42.78  41.67  wine         long_slim_chile.txt
  15   2238   386   42.65  46.89  ares         A_RES_55_259-en.txt
  16    551    89   42.57  41.57  beer         marstons_double_drop.txt
  17    618   104   42.00  42.31  wine         coop_chilean_merlot.txt
  18    718   118   41.72  40.68  sres         S_RES_15052003-en.txt
  19    625   105   41.05  42.86  wine         coop_shiraz_rose.txt
  20    823   139   41.02  41.01  sres         S_RES_13942002-en.txt
  21    699   119   40.00  40.34  beer         abbot_ale.txt
  22   1934   298   39.90  39.93  ares         A_RES_56_238-en.txt
  23   1300   210   39.58  43.33  ares         A_RES_56_234-en.txt
  24   1870   317   39.12  39.75  ares         A_RES_57_313-en.txt
  25   1009   169   37.72  37.87  ares         A_RES_56_512-en.txt
  26    841   136   37.17  41.91  wine         coop_fairtrade_chenin_blanc.txt
  27   1786   281   37.05  37.72  ares         A_RES_57_310-en.txt
  28   1463   244   36.61  35.66  ares         A_RES_56_209-en.txt
  29    709   123   36.06  34.96  beer         mcewans_amber.txt
  30   1097   177   35.97  35.59  ares         A_RES_56_211-en.txt
  31    797   133   35.96  36.09  beer         wychwood_goliath.txt
  32   2448   396   35.48  35.35  ares         A_RES_56_233A-A_RES_56_233-en.txt
  33   2465   417   35.44  38.85  leaflet      Ossopan_Granules.txt
  34    920   158   35.40  34.81  beer         wychwood_golden_ale.txt
  35   1197   187   35.14  35.83  ares         A_RES_55_229-en.txt
  36   1500   239   34.78  36.82  sres         S_RES_14922003-en.txt
  37   2472   379   34.78  35.88  ares         A_RES_56_46-en.txt
  38   1045   180   34.61  38.33  leaflet      Persantin.txt
  39    985   157   34.38  34.39  ares         A_RES_55_72-en.txt
  40   1964   327   34.05  32.42  ares         A_RES_56_274A-A_RES_56_274-en.txt

[.... many lines omitted ....]

1941   8704  1265    0.80   1.19  leaflet      Primacor.txt
1942   4102   709    0.78   0.85  tedtalk      1206PhilipZimbardo.txt
1943  15480  2838    0.76   0.99  tedtalk      0500C.K.Williams.txt
1944  10354  1543    0.73   1.04  leaflet      Cordarone_X_Injection.txt
1945  14210  2879    0.70   0.83  tedtalk      1715JoshuaPrager.txt
1946   1845   321    0.54   0.93  tedtalk      1173MayaBeiser.txt
1947   2509   481    0.48   0.62  tedtalk      0814DerekSivers.txt
1948   9496  1637    0.41   0.55  tedtalk      1601GeorgetteMulheir.txt
1949   1919   355    0.00   0.00  tedtalk      1737MalcolmLondon.txt
1950    762   152    0.00   0.00  tedtalk      0988DavidByrne,Ethel+ThomasDolby.txt
1951   1496   289    0.00   0.00  tedtalk      0119Stew.txt
1952    794   155    0.00   0.00  tedtalk      0117NatalieMacMaster.txt
1953    463    80    0.00   0.00  wine         oxford_landing_2011_merlot.txt
```

```
:: Processing files from test metafile c:\formulib\mets\testing.txt :
Category coverage% (characters, tokens) by frequent n-grams :

0 ares              14.6338   15.6725
1 beer              20.3588   20.7117
2 leaflet           12.7932   14.2807
3 sres              15.6157   16.4139
4 tedtalk            3.5699    4.6923
5 wc                 4.2501    5.0465
6 wine              15.5344   15.3014

Document coverage% (characters, tokens) by frequent n-grams :

   1    587    97   49.57  49.48  beer         youngs_hummingbird.txt
   2    452    72   46.90  51.39  beer         cumberland_corby_blonde.txt
   3    596   102   46.64  45.10  sres         S_RES_13212000-en.txt
   4    925   152   46.16  43.42  ares         A_RES_56_284-en.txt
   5    940   154   45.74  42.86  sres         S_RES_15042003-en.txt
   6    962   151   45.32  46.36  ares         A_RES_56_49-en.txt
   7    782   131   44.63  45.80  sres         S_RES_14852003-en.txt
   8   2121   350   42.15  41.71  ares         A_RES_56_233B-en.txt
   9    947   146   41.82  41.10  ares         A_RES_56_276-en.txt
  10    593   100   41.15  41.00  beer         wells_bombadier.txt
  11   1482   245   39.54  41.63  sres         S_RES_13232000-en.txt
  12   1032   168   39.53  39.29  ares         A_RES_56_273-en.txt
  13   1660   281   38.31  37.37  ares         A_RES_55_244-en.txt
  14   1251   207   38.13  37.20  ares         A_RES_56_230-en.txt
  15   1199   186   37.86  38.17  ares         A_RES_56_270-en.txt
  16   2654   397   37.26  35.01  sres         S_RES_14132002-en.txt
  17    657   105   36.99  33.33  wine         one_tree_lake_shiraz.txt
  18    636   105   36.79  37.14  beer         lancaster_blonde.txt
  19    717   111   35.70  38.74  wine         coop_chianti_2013.txt
  20   1336   211   35.70  36.02  ares         A_RES_56_42-en.txt
  21   3539   633   35.12  36.81  leaflet      Cafergot_Tablets.txt
  22   1142   188   34.59  34.57  sres         S_RES_12942000-en.txt
  23    667   106   34.03  33.02  sres         S_RES_13472001-en.txt
  24   1486   244   33.38  32.79  ares         A_RES_56_133-en.txt
  25   1338   217   33.18  32.72  ares         A_RES_57_312-en.txt
  26   1093   177   33.12  34.46  sres         S_RES_15572004-en.txt
  27    918   160   33.01  31.87  beer         marstons_amber_ale.txt
  28   2522   416   32.59  32.45  ares         A_RES_57_311-en.txt
  29   1674   251   32.56  33.47  ares         A_RES_56_239-en.txt
  30   2095   334   32.17  31.14  ares         A_RES_55_225B-en.txt
  31    716   122   32.12  31.15  sres         S_RES_14882003-en.txt
  32   2254   347   32.08  33.72  ares         A_RES_55_215-en.txt
  33   4148   612   31.70  30.72  sres         S_RES_15632004-en.txt
  34   1376   220   31.69  33.18  ares         A_RES_55_281-en.txt
  35    726   122   31.68  31.15  sres         S_RES_13812001-en.txt
  36   2685   434   31.66  33.64  ares         A_RES_55_220C-en.txt
  37   2143   326   31.45  32.21  ares         A_RES_56_271-en.txt
  38   1270   205   31.42  30.73  ares         A_RES_56_264-en.txt
  39   3372   573   31.35  33.68  leaflet      Sanomigran_Tablets.txt
  40   2397   383   31.16  32.38  ares         A_RES_56_95-en.txt

[.... many lines omitted ....]

1189  16344  2954    0.97   1.39  wc           Savrola3.txt
1190   9193  1652    0.95   1.21  tedtalk      1019BartWeetjens.txt
1191   3409   643    0.94   0.93  tedtalk      1068SuheirHammad.txt
1192  13930  2441    0.88   1.31  tedtalk      1118DavidChristian.txt
1193   6710  1172    0.88   1.11  tedtalk      0524BenKatchor.txt
1194   3933   737    0.81   0.81  leaflet      Serevent_Diskhaler.txt
1195   3529   676    0.77   0.89  tedtalk      0235SiegfriedWoldhek.txt
1196   1587   310    0.69   0.97  tedtalk      1500UsmanRiaz+PrestonReed.txt
1197  21034  3699    0.59   0.92  wc           SpionKop.txt
1198  10231  1790    0.52   0.67  tedtalk      1476BeebanKidron.txt
1199   6198   946    0.45   0.63  leaflet      Adenocor.txt
1200   6936  1135    0.20   0.26  leaflet      Calciparine.txt
1201   1194   238    0.00   0.00  tedtalk      1740JohnLegend.txt
1202    618   109    0.00   0.00  tedtalk      1172OnyxAshanti.txt
1203   1191   229    0.00   0.00  tedtalk      0639ImogenHeap.txt
1204    530   106    0.00   0.00  tedtalk      0115RachelleGarniez.txt
1205    746   133    0.00   0.00  tedtalk      0099JillSobule.txt
1206    469    80    0.00   0.00  wine         oxford_landing_2011_sauvignon_blanc.txt

N.B. category tedtalk used if class label previously unseen.
```

This listing starts with a reminder of some of the more important parameter settings. Next comes a listing showing the proportions of each text category covered by the frequent n-grams as well as the proportion of each document covered in this way. Two percentages are given for each category or document: first character coverage, then token coverage. This is a kind of self-test, since the texts are being scored using the n-grams generated from themselves.

Next, if a testmeta parameter is specified, as is done here, the same information is given for the texts in the test metafile. In this example there were 1953 text files in the training set and 1206 in the test set. Only the 40 highest scoring and some low-scoring documents from both corpora are shown above, to save space. The second ranking represents a genuine test, since the texts are being processed by n-grams derived from a different group of documents, though with the same class label.

The essential idea underlying this program is that the high-scoring items are highly repetitive and therefore formulaic while the low-scoring items are less repetitive, thus less formulaic. Moreover we can quantify this attribute as a percentage.

Taking the first 2 lines from the second part of the ranking (the holdout or testing subsample)

```
1    587    97    49.57  49.48  beer         youngs_hummingbird.txt
2    452    72    46.90  51.39  beer         cumberland_corby_blonde.txt
```

as an example, the output format is as follows. The first column is its rank (highest coverage by frequent sequences). The next 2 columns give the number of characters and tokens in the document. The next pair of numbers indicates the actual percentage covered, first in terms of characters, then by tokens. The next 2 strings give the name of the files concerned, preceded by their category label.

Thus the 2 texts most covered by frequent n-grams from their category were both from the beer subcorpus. In terms of characters, 49.57% of the total length of the text of Youngs Hummingbird was covered by frequent 3:6-grams; and 49.48% of the tokens were so covered.

At the foot of such a listing, items with zero coverage generally deserve further investigation. For example, here one of the TED talks (0115 by Rachelle Garniez) turns out on inspection to be a song in French with just a few English words of introduction. Anomaly detection of this kind can be useful for various purposes.

How the program works

The basic idea behind this program is very simple. The trouble with n-gram lists is that they tend to contain multiple fragments of longer sequences, losing track of what might be considered the natural length of the sequences from which they are derived. So formulex.py tries to put them back together by going back over the original texts to find out exactly which passages are covered by the items in the frequent n-gram list. The key concept here is **coverage**. The important point is that a text sequence is either covered or not: the number of n-grams that match a particular sequence of tokens doesn't matter, just whether any do or none.

To give an illustration of the coverage process, let's suppose that you have gathered a corpus of political propaganda in which the phrase

> "securing a better future for hardworking families"

is repeated ad nauseam. (Yes, they do tend to write "hardworking" rather than "hard-working" or

"hard working". I checked.)

This could be regarded as a frequent 7-gram, but with the system's default settings, the longest n-grams to be saved would be 5-grams. Thus the n-gram list would probably include

"securing a better future for"
"a better future for hardworking"
"better future for hardworking families"

as well as shorter subsequences, probably going down to 2-grams such as "better future" and "hardworking families". Suppose further that the program is processing the sentence tabulated below (shown vertically, for convenience).

(word) token	**match count**	**covering n-gram(s):**		
we	0			
are	0			
committed	0			
to	0			
securing	1	securing		
a	2	a	a	
better	3	better	better	better
future	3	future	future	future
for	3	for	for	for
hardworking	2		hardworking	hardworking
families	1			families
throughout	0			
britain	0			

To keep things manageable, we've ignored the 4-, 3- and 2-grams, which might well increase the totals in the column labelled "match count", but the main point is that 'coverage' will be determined by whether this figure is greater than zero or not. The total number of matches isn't taken into account for this purpose. (It could have value in certain contexts: see discussion of flicshow.py below.)

Sticking just to these 13 words (87 characters, including single spaces between words) and three 5-grams, the coverage would be 48/87 characters and 7/13 tokens, i.e. just the words that have a nonzero entry next to them under "match count". This would appear as 55.17% and 53.85% in the output. These two percentages tend to be highly correlated, meaning that similar conclusions are likely to be drawn from either. I placed the character-coverage first, and used that as the primary sort key, because I believe it is likely to be a slightly more sensitive indicator.

To summarize, the program works out coverage of tokens in this manner for each file separately using the n-grams from the same category as the text concerned (or the largest category if the text has an unseen class label) and also aggregates the coverage for each category. The texts are listed in descending order of character coverage.

To return to the category coverage in the output above, the four categories ares, beer, leaflet, sres and wine all have values over 12 percent. As expected, the UN resolutions, the bottle labels and the drug leaflets are all rather repetitive. The less formulaic categories are TED talks and Winston Churchill's writings (mostly speeches). These have character coverages between 3.56 and 4.25

percent when 3:6-grams from the training subset are applied to the holdout sample.

The largest difference between self-test and holdout mode is in the beer category, but this is a comparatively tiny corpus, so would be expected to show more variability.

Examining the formulexicon

As well as rating both texts and text categories by formulaic coverage, formulex.py produces another output file (formtest_flab.txt in this example) which is intended to give a glimpse of what might be regarded as the 'formulexicon' of each category. An extended extract from formtest_flab.txt follows.

```
[....]
2 leaflet 289 305939 1758713
 0.3122   323   16   3  tell your doctor
 0.2354   345   11   3  if you have
 0.2252   233   16   3  your doctor will
 0.2183   349   10   3  if you are
 0.2098   246   14   3  you are taking
 0.2083   229   15   3  your doctor may
 0.1825   107   29   5  ask your doctor or pharmacist
 0.1814   145   21   3  the active ingredient
 0.1660   139   20   3  taking your medicine
 0.1632    33   86  16  if you have any questions or are not sure about anything ask your
doctor or pharmacist
 0.1465   184   13   3  should not be
 0.1443    94   26   5  what you should know about
 0.1410   124   19   3  high blood pressure
 0.1361    63   37   8  is one of a group of medicines called
 0.1331   195   11   3  do not take
 0.1271   149   14   3  by your doctor
 0.1238   198   10   3  do not use
 0.1199   111   18   4  do you suffer from
 0.1183   130   15   3  the expiry date
 0.1164    16  127  26  remember this medicine is for you only a doctor can prescribe it for
you never give it to someone else it may harm them even if
 0.1154    70   28   6  out of the reach of children
 0.1138    69   28   4  tell your doctor immediately
 0.1035    10  181  33  please read this carefully before you start to take your medicine if
you have any questions or are not sure about anything ask your doctor or pharmacist the name
of your medicine is
 0.1026    41   43   8  in a safe place where children cannot reach
 0.1006    61   28   6  the name of your medicine is
 0.0979   123   13   3  used to treat
 0.0952    54   30   6  if your doctor decides to stop
 0.0938   110   14   3  you should not
 0.0917    26   61  13  it may harm them even if their symptoms are the same as yours
 0.0867    25   60   9  this leaflet does not contain the complete information about
 0.0843    19   77  15  if the answer to any of these questions is yes tell your doctor or
pharmacist
 0.0842    74   19   3  consult your doctor
 0.0836    98   14   3  to your doctor
 0.0819    96   14   3  are you taking
 0.0812    68   20   4  any of the following
 0.0812    51   27   4  before taking your medicine
 0.0810    89   15   3  your doctor has
 0.0799    19   73  12  please read this leaflet carefully before you start to take your
medicine
 0.0798    78   17   4  if you are taking
 0.0782    86   15   3  ask your doctor
 0.0778    37   36   7  tell your doctor as soon as possible
 0.0764    48   27   6  on the back of this leaflet
 0.0739    52   24   5  your doctor has told you
 0.0732     9  142  25  a summary of the information available on your medicine if you have
any questions or are not sure about anything ask your doctor or pharmacist
 0.0723    41   30   5  tell your doctor or pharmacist
 0.0719    79   15   3  it is important
 0.0717    97   12   3  any of these
 0.0702    65   18   4  it is important to
 0.0701   112   10   3  you do not
 0.0675    54   21   4  to take your medicine
 0.0657    68   16   4  is used to treat
 0.0648    95   11   3  do you have
 0.0648    57   19   5  if you are not sure
```

```
 0.0619    64   16   3  this medicine is
 0.0606    41   25   4  your doctor or pharmacist
 0.0605    38   27   6  if your doctor tells you to
 0.0604     9  117  24  if your doctor tells you to remember this medicine is for you only a
doctor can prescribe it for you never give it to
 0.0600    44   23   5  as soon as you remember
 0.0597    42   24   5  your doctor tells you to
 0.0587    43   23   5  tell your doctor if you
 0.0555    61   15   4  if you have any
 0.0541    34   27   5  a group of medicines called
 0.0523    40   22   4  check with your doctor
 0.0520    15   60  14  if you forget to take a dose take it as soon as you remember
 0.0509    14   63  14  for you only a doctor can prescribe it for you never give it to
 0.0503    52   16   3  with your doctor
 0.0489    43   19   4  as soon as possible
 0.0478    35   23   3  your doctor immediately
 0.0474    49   16   3  your medicine is
 0.0474     7  118  20  information about your medicine if you have any questions or are not
sure about anything ask your doctor or pharmacist
 0.0471    46   17   3  reach of children
 0.0469     5  164  33  remember this medicine is for you only a doctor can prescribe it for
you never give it to someone else it may harm them even if their symptoms are the same as
yours
 0.0461     9   89  18  remember this medicine is for you only a doctor can prescribe it for
you never give it to
 0.0445    29   26   6  it as soon as you remember
 0.0445    27   28   5  to your doctor or pharmacist
 0.0444    39   19   4  if you are pregnant
 0.0430     3  251  49  if you have any questions or are not sure about anything ask your
doctor or pharmacist remember this medicine is for you only a doctor can prescribe it for you
never give it to someone else it may harm them even if their symptoms are the same as yours
 0.0428    47   15   3  you suffer from
 0.0422    53   13   4  to any of the
 0.0407     5  142  27  or are not sure about anything ask your doctor or pharmacist remember
this medicine is for you only a doctor can prescribe it never give it to
3 sres 163 134809 851798
 0.4001    71   47   8  decides to remain actively seized of the matter
 0.2967   133   18   3  in accordance with
 0.2914    73   33   5  requests the secretary general to
 0.2744   123   18   3  the united nations
 0.2614    17  130  20  the relevant principles contained in the convention on the safety of
united nations and associated personnel adopted on 9 december
 0.2454    95   21   4  of the united nations
 0.2219   105   17   3  the importance of
 0.2184   124   14   3  the parties to
 0.2170    42   43   8  2002 adopted by the security council at its
 0.2134   101   17   3  the government of
 0.2120    86   20   3  the security council
 0.1956   119   13   3  th meeting on
 0.1860    36   43   8  2004 adopted by the security council at its
 0.1860    36   43   8  2003 adopted by the security council at its
 0.1834    71   21   3  the secretary general
 0.1831    40   38   7  decides to remain seized of the matter
 0.1831    39   39   7  of the democratic republic of the congo
 0.1807    27   56   9  having considered the report of the secretary general of
 0.1740    57   25   4  the implementation of the
 0.1648    36   38   7  the report of the secretary general of
 0.1566    23   57   7  the security council reaffirming its previous resolutions
 0.1550    88   14   3  in this regard
 0.1550    30   43   8  2001 adopted by the security council at its
 0.1545    28   46   6  the security council recalling its resolutions
 0.1472    57   21   3  the implementation of
 0.1409    30   39   7  in the democratic republic of the congo
 0.1362    29   39   7  the date of adoption of this resolution
 0.1323    23   48   6  the committee established pursuant to resolution
 0.1317    66   16   3  member states to
 0.1313    43   25   4  the government of lebanon
 0.1310    62   17   3  in particular the
 0.1266    49   21   4  by the united nations
 0.1240    96   10   3  as well as
 0.1233    42   24   4  referred to in paragraph
 0.1212    43   23   3  the secretary general's
 0.1136    22   43   8  of the democratic republic of the congo and
 0.1127    40   23   4  of the security council
 0.1124    29   32   6  decides to extend the mandate of
 0.1115    25   37   6  the president of the security council
 0.1107    41   22   4  in accordance with the
```

```
 0.1085    66   13   3  to ensure the
 0.1080     10    91   13  the security council recalling its previous resolutions and the
statements of its president
 0.1078    54   16   3  the situation in
 0.1059    22   40   6  requests the secretary general to submit
 0.1038    17   51   7  the special representative of the secretary general
 0.1019    28   30   5  the government of sierra leone
 0.1014    54   15   3  support for the
 0.0986    30   27   4  adoption of this resolution
 0.0983    31   26   5  its strong support for the
 0.0981    38   21   3  implementation of the
 0.0944    67   11   3  in order to
 0.0939    32   24   4  the secretary general to
 0.0930    18   43   5  the international security assistance force
 0.0916    30   25   4  in bosnia and herzegovina
 0.0916    30   25   3  the committee established
 0.0904    11   69  12  acting under chapter vii of the charter of the united nations decides
 0.0898    51   14   3  to continue to
 0.0873    12   61  11  acting under chapter vii of the charter of the united nations
 0.0870    19   38   7  adopted by the security council at its
 0.0863    49   14   3  to the council
 0.0852    22   32   4  international peace and security
 0.0839    65   10   3  set out in
 0.0839    13   54   8  the president of the international tribunal for rwanda
 0.0838    21   33   5  the measures imposed by paragraph
 0.0818    17   40   6  of the international tribunal for rwanda
 0.0817    29   23   4  the measures imposed by
 0.0817    12   57   8  permanent judges of the international tribunal for rwanda
 0.0808     8   85  13  the implementation of the peace agreement and the situation in bosnia
and herzegovina
 0.0805    49   13   3  in the region
 0.0789    32   20   3  of the international
 0.0778    17   38   6  the security council recalling all its
 0.0764    21   30   4  the secretary general's report
 0.0761    36   17   3  the peace process
 0.0758    19   33   6  the statement of its president of
 0.0758    19   33   5  determining that the situation in
 0.0757    43   14   4  as well as the
 0.0738    17   36   7  of the charter of the united nations
 0.0729    27   22   3  bosnia and herzegovina
 0.0723    22   27   5  to cooperate fully with the
 0.0710      5  120  21  a threat to international peace and security in the region acting
under chapter vii of the charter of the united nations
4 tedtalk 998 2150340 11526504
 0.0866  1109    8   3  a lot of
 0.0766   465   18   3  thank you applause
 0.0691   613   12   3  in the world
 0.0624   654   10   3  one of the
 0.0509   391   14   3  we're going to
 0.0475   322   16   3  around the world
 0.0464   486   10   3  we need to
 0.0453   180   28   5  thank you very much applause
 0.0444   394   12   3  i'm going to
 0.0429   494    9   3  i want to
 0.0427   410   11   3  and i think
 0.0411   395   11   3  some of the
 0.0407   313   14   3  percent of the
 0.0387   343   12   3  a little bit
 0.0369   354   11   3  part of the
 0.0357   343   11   3  you can see
 0.0331   347   10   3  out of the
 0.0328   315   11   3  a couple of
 0.0325   312   11   3  in order to
 0.0323   233   15   3  you're going to
 0.0321   264   13   3  the fact that
 0.0311   299   11   3  that we can
 0.0310   357    9   3  and i was
 0.0309   274   12   3  this is what
 0.0307   272   12   3  i don't know
 0.0305   293   11   3  there was a
 0.0298   286   11   3  look at the
 0.0295   162   20   4  in the united states
 0.0292   280   11   3  in terms of
 0.0285   299   10   3  when i was
 0.0283   272   11   3  i wanted to
 0.0283   233   13   3  and of course
 0.0283   181   17   3  the united states
```

```
 0.0282   296   10   3  be able to
 0.0282   295   10   3  there is a
 0.0276   245   12   3  this kind of
 0.0273   286   10   3  and it was
 0.0267   237   12   3  of the world
 0.0262   275   10   3  all of the
 0.0255   196   14   3  to think about
 0.0254   244   11   3  and you can
 0.0253   324    8   3  it was a
 0.0250   222   12   3  that you can
 0.0250   240   11   3  you want to
 0.0248   260   10   3  we want to
 0.0241   278    9   3  a kind of
 0.0238   229   11   3  back to the
 0.0238   249   10   3  and i said
 0.0237   228   11   3  is going to
 0.0234   245   10   3  and if you
 0.0227   238   10   3  and a half
 0.0214   224   10   3  to do with
 0.0213   189   12   3  are going to
 0.0210   202   11   3  going to be
 0.0210   121   19   4  thank you very much
 0.0209   142   16   4  at the same time
 0.0209   127   18   4  for the first time
 0.0197   284    7   3  to be a
 0.0196   161   13   4  to be able to
 0.0194   140   15   4  a little bit of
 0.0180   173   11   3  i'd like to
 0.0178   187   10   3  by the way
 0.0177   146   13   3  it's going to
 0.0171   152   12   3  not going to
 0.0169   150   12   3  i think that
 0.0169   130   14   4  if you want to
 0.0166   137   13   3  to talk about
 0.0158   114   15   4  a lot of people
 0.0157   113   15   4  one of the most
 0.0156   100   17   4  it turns out that
 0.0155   137   12   3  it turns out
 0.0153   126   13   3  in the middle
 0.0151   109   15   4  and you can see
 0.0151   193    8   3  to do it
 0.0148    90   18   4  all over the world
 0.0147    89   18   5  if you look at the
 0.0144   208    7   3  the u s
 0.0136   112   13   4  i want you to
 0.0131    89   16   4  and this is what
 0.0131    89   16   4  and i'm going to
5 wc 31 93239 519204
 0.0832    24   17   3  the united states
 0.0768    21   18   3  the prime minister
 0.0740    16   23   4  the right hon gentleman
 0.0722    15   24   3  his majesty's government
 0.0651    26   12   3  of the world
 0.0624    36    8   3  i do not
 0.0578    20   14   3  of the british
 0.0576    13   22   3  the conservative party
 0.0539    14   19   4  at the present time
 0.0508    22   11   3  there is no
 0.0501    20   12   3  in the world
 0.0482    25    9   3  it is not
 0.0468     9   26   4  the committee of ministers
 0.0462    16   14   3  that we should
 0.0462    15   15   3  would have been
 0.0462    15   15   3  in this country
 0.0458    17   13   3  which we have
 0.0451    13   17   3  of the government
 0.0447     8   28   5  the leader of the opposition
 0.0445    21   10   3  of all the
 0.0439    19   11   3  at any rate
 0.0439    12   18   4  the house of lords
 0.0433    15   14   3  in the country
 0.0416     9   23   5  of the house of commons
 0.0403    19   10   3  be able to
 0.0401    13   15   3  in south africa
 0.0376    15   12   3  of the house
 0.0364     9   20   4  of the united states
 0.0360    17   10   3  one of the
```

```
 0.0351    14   12   3  which we are
 0.0347    15   11   3  will not be
 0.0347    15   11   3  what is the
 0.0347    15   11   3  it would be
 0.0347    15   11   3  in order to
 0.0347    12   14   3  of the country
 0.0339    11   15   4  in spite of the
 0.0339     8   21   4  of the united nations
 0.0337     7   24   4  great britain and france
 0.0325    13   12   3  we have been
 0.0324    14   11   3  in the last
 [....]
```

This extract shows all 80 of the sequences produced from three of the text types: the patient leaflets (leaflet), the UN Security Council data (sres) and the TED transcripts (tedtalk), followed by the first 40 items from the Churchill sample (wc). The contrast between highly stylized writing and spoken discourse (albeit largely rehearsed) emerges rather forcefully. In addition, it serves to introduce the notion that the system can find what I call 'collocades', i.e. chains or cascades of collocations.

Each group of data lines is preceded by a 1-line header such as

```
2 leaflet 289 305939 1758713
```

which indicates that category number 2 is the patient leaflets consisting of 289 files containing 305939 tokens amounting to a total of 1758713 characters. Then come the sequences themselves. For example, in the sres group

```
 0.1825   107   29   5  ask your doctor or pharmacist
```

comes in 7th place, while

```
 0.1632    33   86  16  if you have any questions or are not sure about anything
ask your doctor or pharmacist
```

comes in tenth place.

Here we have a pair of sequences that are clearly related, a 5-gram that also appears as a segment of a 16-gram. Their relationship may not be obvious, but can be elucidated from the figures given.

The numbers 0.1825 and 0.1632 are percentages. They indicate that 0.1825 percent of the entire 1758713 characters in the leaflet corpus consists of repetitions of the phrase "ask your doctor or pharmacist", which occurs 107 times and is 29 characters long. However, the figure of 107 occurrences only refers to those occurrences of this 5-gram that are not included within a longer sequence. Specifically, that excludes the 33 occurrences within the 16-gram "if you have any questions or are not sure about anything ask your doctor or pharmacist". As it happens, the _gram.dat output file shows that the 5-gram "ask your doctor or pharmacist" occurs 319 times altogether in this sample of 289 leaflets, ignoring whether or not it is part of a longer collocade. In the listing above this particular 5-gram can be found as part of 6 other collocades in this list, though the total of their occurrences do not add to 319, implying that it appears in other contexts, albeit less frequently.

Similar remarks apply to sequences such as

```
 0.2120    86   20   3  the security council
 [....]
 0.1860    36   43   8  2004 adopted by the security council at its
 0.1860    36   43   8  2003 adopted by the security council at its
 [....]
 0.1566    23   57   7  the security council reaffirming its previous resolutions
 [....]
```

```
 0.1550    30   43   8  2001 adopted by the security council at its
 0.1545    28   46   6  the security council recalling its resolutions
```

from the sres subcorpus. These lines, in effect, show the most common contexts of the sequence "the security council" (which, ignoring context, occurs as a 3-gram 508 times, as can be found from the _gram.dat file).

I have considered writing a program to collate such linked sequences, but in the general case that would be tricky and more appropriate in a concordance package, so at present this listing can be regarded as pointing to what sequences to examine using a conventional concordance program.

As for formulaic language, it certainly identifies some prefabricated phrases. Taking only the first-ranked item from the sres corpus

```
0.4001    71   47   8  decides to remain actively seized of the matter
```

we find an 8-gram that covers more than 0.4 percent of the whole corpus. If that isn't a formulaic phrase, I don't know what is! The figures show that this sequence comprising 47 characters or 8 tokens occurred 71 times. It might seem that 0.4001 percent is quite a small number, but if you look further down the list you'll see that such a common phrase as "as well as" only accounts for 0.1240 percent of the characters in this text type; and, for instance, in the Churchill sample (wc) "one of the" accounts for 0.0360 percent. So for an 8-gram to take as much as 0.4001 percent of the text in a corpus is quite a feat.

The longest of these Security Council sequences is a 21-gram "a threat to international peace and security in the region acting under chapter vii of the charter of the united nations" which occurs 5 times accounting for 0.0710 percent of the entire corpus. This is a good example what I call a 'collocade', several collocated subsequences which happen to be consecutive. A look at the n-gram list suggests it can be broken down as "a threat to international peace and security" + "in the region" + "acting under chapter vii of the charter of the united nations".

The way of counting employed by formulex.py can take some getting used to, but it does mean that the sequences shown are mutually exclusive. It also prevents longer prefabricated phrases from being swamped by the elements of which they are composed, which is an important objective of this software.

The tedtalk category presents a sharp contrast. All but 2 of its 80 high-ranking sequences are 3-grams or 4-grams. Short markers of spoken language abound, such as "a lot of", "a kind of" and "i'm going to". (The last could perhaps have been represented as a pair, "i'm gonna", with a less formal transcription convention.) Even one of the 5-grams, "thank you very much applause", is artificially inflated by one token as a result of a particular transcription convention. (The other 5-gram is "if you look at the".) This is a generic corpus covering a wide variety of topics, so even after forbidding n-grams entirely composed entirely of stop-words we're left with very basic short chunks of syntax. In contrast with the UN sequences, no obvious topic focus can be discerned.

Running flicshow.py

After examining a list of high-frequency sequences of various lengths, including collocades, it is natural to wonder not just how often they occur, but where. A conventional KWIC (Keyword in Context) listing doesn't really answer this need, so I have written flicshow.py to offer an alternative approach.

The main input parameter for flicshow.py is metaflic, which specifies a metafile giving the text files

to be processed. The other parameters are typically the same as for formulex.py. This program also uses the n-gram file produced by outgrams.py.

The program processes each of the files specified by metaflic using the same mode of calculating coverage as in formulex.py. It places them in a subdirectory called html of whichever output folder has been specified by the outpath parameter. These output files are in html and should be viewed with a browser such as Mozilla Firefox or Microsoft Edge.

In these output files the portions covered by the n-grams of the relevant category (i.e. the category of the text being processed, or the largest category if it has an unknown label) are highlighted by being colour-coded, while the rest of the text is printed in black. The html file for Artesian Dark Ale from the beer subcorpus is reproduced below. This comes from the holdout sample, i.e. was not part of the collection used to create the n-gram file.

artesian dark ale see amber smell fruity spicy hoppy taste fruity biscuit malt spice this delicious ale is hand crafted using chalk filtered mineral water from the artesian well deep below our brewery amber in colour it is a complex full flavoured beer combining aromas of fruits and spices with rich biscuity malts drink responsibly
uk chief medical officers recommend men do not regularly exceed 4 units daily and women 3 units daily drinkaware co uk
for the facts 1.9 uk units brewed bottled by shepherd neame ltd faversham kent me13_7ax england
contains barley malt
ingredients water barley hops and malt
return for refund where applicable product of england
10 cent refund at collection depots when sold in south australia standard drinks 1.5 beer
500 ml alcohol
3.8 percent volume brewed bottled by shepherd neame ltd faversham kent me13_7ax england
best before end see neck of bottle allergy advice contains barley malt
gluten www shepherdneame co uk

The colour scheme used by flicshow.py is as follows.

Coverage score	Colour
6+	purple
5	red
4	orange
3	green
2	blue
1	cyan
0	black

The coverage score for each token, and hence its colour, can be computed in 2 ways, depending on the value of the parameter huemode. If huemode is 0, it will be the size of the longest n-gram of the category concerned in the gramdat file that covers that particular token. If huemode is 1, the score will be the sum total of n-grams, of any size, from the _gram.dat file that covers that token. Huemode 0, the default, is used in all he examples shown here.

The way this operates can be illustrated by the penultimate line above

best before end see neck of bottle allergy advice contains barley malt

which is a collocade that starts with a 5-gram in red, but then has 2 tokens in orange, followed by five more in green. Orange is used for a score of four, so it might seem strange that only 2 tokens are in orange: the reason is that "see neck of bottle" is a frequent 4-gram but "best before end see neck" is in the top 80 5-grams and each token gets the score associated with the longest n-gram that applies to it. Thus "see neck" receives a higher score than "of bottle". Conversely, the 5 tokens in "allergy advice contains barley malt" appear in green, appropriate to a score of 3 because each of these five tokens is covered by no n-gram longer than 3, such as "allergy advice contains" and "contains barley malt".

This multiple colouration of collocades might almost be considered a kind of parsing: it hints at the way the phrases are prefabricated.

Although the Artesian Dark Ale file was not part of the 'training data', its colouring illustrates a relatively high degree of coverage by formulaic sequences (29.23% by characters, 29.94% by tokens). An even higher level of coverage (35.12% and 36.81%) is illustrated by the Cafergot text from the patient leaflet test sample, below.

what you should know about
cafergot tablets
please read this carefully before you start to take your medicine
even
if you have
taken cafergot before
if you have any questions or are not sure about anything ask your doctor or pharmacist
chemist
the name of your medicine is
cafergot ergotamine with caffeine it should be taken at the start of a migraine attach to relieve the symptoms and stop the attack getting worse things to remember about cafergot make sure at is safe for you to take cafergot see inside leaflet
take your medicine
exactly as
your doctor tells you
remember to keep a record of how many tablets you take in a day and in a week
do not take
cafergot every day it
you start to take
any other medicine make sure
your doctor or pharmacist
knows cafergot can sometimes cause side effects you can find these listed inside this leaflet keep your medicine away from children you will find more about your tablets inside this leaflet
your medicine is
called cafergot the ergotamine in this medicine affects the tension of the blood vessels causing your migraine attack your tablets also contain caffeine this helps the ergotamine to be absorbed into the body more quickly so you get quicker relief from your migraine
before taking your medicine do not take
cafergot

if you are pregnant
or become pregnant
or if you are
breast feeding a baby
tell your doctor or pharmacist if you suffer from any of the following
kidney or liver disease circulation or heart problems
or if you are taking
medicine for
high blood pressure taking your medicine it is important to take your medicine
correctly take only the amount directed
by your doctor
the label will tell you how much to take and how often if it doesn't or
you are not sure ask your doctor or pharmacist
remove the tablet from the foil as shown in the picture always take the tablets with water and swallow them whole
it is important to
keep a record of how many tablets you take
you should not
use more than 4 tablets in 24 hours
do not take
cafergot every day leave a gap of at least 4 days before taking a further dose
you should not
use more than 8 tablets in a week
if you are not
getting relief from your migraine
do not take
more tablets
tell your doctor if you
find your headaches occur more often
tell your doctor
overdose if you accidentally take too much of your medicine
tell your doctor immediately
or go to your nearest casualty department after
taking your medicine
there is no need to worry if you develop an upset stomach but
tell your doctor
at your next visit if you develop numbness or tingling in your fingers or toes
tell your doctor immediately
storing your medicine keep your tablets
in a safe place where children cannot reach
them your tablets could harm them it
your doctor decides to stop
your treatment return any leftover medicine to the pharmacist only keep them
if your doctor tells you to
what's in your medicine cafergot tablets are round and white each one contains 1 mg ergotamine tartrate and 100 mg caffeine further information

remember this medicine is for you only a doctor can prescribe it for you never give it to
anyone
else it may harm them even if their symptoms are the same as yours this leaflet does not contain
complete
information about your medicine if you have any questions or are
unsure
about anything ask your
pharmacist cafergot is a registered trade mark sandoz pharmaceuticals frimley business park frimley camberley surrey gu16 5 sg pl 0101 5 c23r sandoz products ireland limited airton road tallaght dublin 24 pa 13 5 1

By contrast, TED talk 1716, which also was absent from the examples used to generate the n-gram file, shows a much lesser degree of coverage.

when i was
14 years old i had low self esteem i felt i was not talented at anything one day i bought a yo yo when i tried my first trick it looked like this i couldn't even do the simplest trick but it was very natural for me because i was not dextrous and hated all sports but after one week of practicing my throws became more like this a bit better i thought the yo yo is something for me to be good at
for the first time in
my life i found my passion i was spending all my time practicing it took me hours and hours a day to build my skills up to the next level and then four years later
when i was
18 years old i was standing onstage at the world yo yo contest and i won i was so excited yes i did it i became a hero i may get many sponsors
a lot of
money tons of interviews and be on tv i thought laughter but after coming back to japan totally nothing changed in my life i realized society didn't value my passion so i went back to my college and became a typical japanese worker as a systems engineer i felt my passion heart and soul had left my body i felt i was not alive anymore so i started to consider what i should do and i thought
i wanted to
make my performance better and to show onstage how spectacular the yo yo could be to change the public's image of the yo yo so i quit my company and started a career as a professional performer i started to learn classic ballet jazz dance acrobatics and other things to make my performance better as a result of these efforts and the help of many others it happened i won the world yo yo contest again in the artistic performance division i passed an audition for cirque du soleil today i am standing on the ted stage with the yo yo in front of you applause what i learned from the yo yo is if i make enough effort with huge passion there is no impossible could you let me share my passion with you through my performance applause water sounds music applause music music applause applause music applause applause music applause

It is easy to see at a glance that this file contains large sections in black that are not covered by the frequent n-grams, while the highlighted chunks are short and mostly content-free. In fact only 3.27% (by characters) or 4.27% (by tokens) of this text are covered by the most frequent n-grams in tedtalks.

Running taverns.py (Textual Affinity Values Employing Repeated N-gram Sequences)

Inspecting individual texts can be a valuable opportunity to get close to the data, but in a typical corpus there is a huge amount of data to be inspected. The program taverns.py works in bulk mode and thereby gives an indication of which particular files might deserve the kind of close attention given to the output of flicshow.py. It goes a step further than formulex.py, using the same method, by computing coverage of each text specified in the testmeta file not only by the n-grams of its own category, but by those of all the categories in the n-gram file. Thus, in effect, it ranks each text file according to how typical it is of each category, including its own.

In addition, having done this, it performs text classification by assigning each text to the category giving it the highest coverage score. It is not intended primarily as a text classifier, but the results in classification often shed light on the relationships between the text types involved. In particular, it is sometimes interesting to see how texts from outside the classes used to form the n-gram dictionary are classified.

Its main output file is the _ales.txt file (Affinity Listing Employing Sequences). The initial portion of this file generated by using the formtest parameter file follows below.

```
Wed Jun  1 14:12:19 2016
parafile: C:\formulib\parapath\formtest.txt
gramfile: ..\op\formtest_gram.dat
testmeta: c:\formulib\mets\testing.txt
miniglen: 3
maxiglen: 6
topgrams: 80
contrast: 0
1206 7


Ranking by coverage of sequences from ares
   1    925   152   46.16  43.42  ares         A_RES_56_284-en.txt
   2    962   151   45.32  46.36  ares         A_RES_56_49-en.txt
   3   2121   350   42.15  41.71  ares         A_RES_56_233B-en.txt
   4    947   146   41.82  41.10  ares         A_RES_56_276-en.txt
   5   1032   168   39.53  39.29  ares         A_RES_56_273-en.txt
   6   1660   281   38.31  37.37  ares         A_RES_55_244-en.txt
   7   1251   207   38.13  37.20  ares         A_RES_56_230-en.txt
   8   1199   186   37.86  38.17  ares         A_RES_56_270-en.txt
   9   1336   211   35.70  36.02  ares         A_RES_56_42-en.txt
  10   1486   244   33.38  32.79  ares         A_RES_56_133-en.txt
  11   1338   217   33.18  32.72  ares         A_RES_57_312-en.txt
  12   2522   416   32.59  32.45  ares         A_RES_57_311-en.txt
  13   1674   251   32.56  33.47  ares         A_RES_56_239-en.txt
  14   2095   334   32.17  31.14  ares         A_RES_55_225B-en.txt
  15   2254   347   32.08  33.72  ares         A_RES_55_215-en.txt
  16   1376   220   31.69  33.18  ares         A_RES_55_281-en.txt
  17   2685   434   31.66  33.64  ares         A_RES_55_220C-en.txt
  18   2143   326   31.45  32.21  ares         A_RES_56_271-en.txt
  19   1270   205   31.42  30.73  ares         A_RES_56_264-en.txt
  20   2397   383   31.16  32.38  ares         A_RES_56_95-en.txt
```

Here the format is the same as with formulex.py, but it should be noted that percentage coverage of is not being computed for each file by reference to its own n-gram list but, in this section, all coverage scores are computed by reference to the ares n-grams. Nevertheless, all the top 20 are from the ares category. (The highest of another category, a Security Council resolution, comes at rank 75.)

By contrast, the last 30 entries in this ranking are files that have essentially nothing in common with the General Assembly resolutions: none of the General Assembly (ares) n-grams match anywhere in them.

```
1177   1528   251    0.00   0.00  wine         alta_vista_premium_2011.txt
1178    659   113    0.00   0.00  wine         2011_old_man_creek.txt
```

```
1179    587    97    0.00   0.00  beer         youngs_hummingbird.txt
1180    766   128    0.00   0.00  beer         wentworth_imperial_ale.txt
1181    593   100    0.00   0.00  beer         wells_bombadier.txt
1182   1049   174    0.00   0.00  beer         tribute_cornish_pale_ale.txt
1183    652   107    0.00   0.00  beer         timothy_taylors_landlord.txt
1184    861   148    0.00   0.00  beer         theakston_lightfoot.txt
1185   1101   186    0.00   0.00  beer         spitfire.txt
1186   1238   195    0.00   0.00  beer         samuelsmiths_organic_fruitbeer.txt
1187    730   118    0.00   0.00  beer         samuel_adams_boston_lager.txt
1188    621   102    0.00   0.00  beer         sainte_etienne_lager.txt
1189    416    67    0.00   0.00  beer         marstons_english_pale_ale.txt
1190    918   160    0.00   0.00  beer         marstons_amber_ale.txt
1191    636   105    0.00   0.00  beer         lancaster_blonde.txt
1192    483    82    0.00   0.00  beer         greens_supreme_golden_ale.txt
1193    543    88    0.00   0.00  beer         golden_pippin.txt
1194    704   110    0.00   0.00  beer         flying_scotsman.txt
1195    452    72    0.00   0.00  beer         cumberland_corby_blonde.txt
1196   1002   156    0.00   0.00  beer         coop_czech_lager.txt
1197    756   129    0.00   0.00  beer         castle_rock_harvest_pale.txt
1198   1015   167    0.00   0.00  beer         caledonian_flying_scotsman.txt
1199    822   134    0.00   0.00  beer         bud_strong.txt
1200    997   173    0.00   0.00  beer         boadicea_golden_ale.txt
1201    743   129    0.00   0.00  beer         bass_trademark_no1.txt
1202    566    91    0.00   0.00  beer         banks_bitter.txt
1203    836   145    0.00   0.00  beer         badger_tangle_foot.txt
1204    958   157    0.00   0.00  beer         artesian_darkale.txt
1205    483    86    0.00   0.00  beer         adnams_ghost_ship.txt
1206   1099   191    0.00   0.00  beer         9hop_kent_pale_ale.txt

ares mean rank = 159.73
beer mean rank = 1192.5
leaflet mean rank = 861.78
sres mean rank = 311.9
tedtalk mean rank = 763.59
wc mean rank = 574.38
wine mean rank = 1167.5
```

Also printed at the foot of each category's ranking list are the average ranks for all input categories. This gives a rough index of dissimilarity between groups. It shows that ares is indeed the closest category to itself, and that sres is also relatively close, as might be expected. The categories most distant from ares are beer and wine bottle labels and patient information leaflets.

There are seven blocks of ranked lists in this format, one for each n-gram category, which will not be reproduced here. (The full 9774 lines of output can be seen in the file formtest_ales.txt on the op subfolder.)

After these the results of using the method in classification mode are appended. This is shown below, with a large chunk from the middle omitted to save space.

```
Results in classification mode :

      relative   actual       categories
rank  coverage% coverage%     pred : true      docname
   1    100.00    20.86    leaflet + leaflet   Metrogel.txt
   2    100.00    18.61    leaflet + leaflet   Proctosedyl_Suppositories.txt
   3    100.00    16.22    leaflet + leaflet   Fucidin_Cream.txt
   4    100.00    13.57    leaflet + leaflet   Levophed_Injection.txt
   5    100.00    13.30       wine + wine      coop_explorers_vineyard.txt
   6    100.00     9.06       beer + beer      theakston_lightfoot.txt
   7    100.00     8.80    tedtalk + tedtalk   1083AhnTrio.txt
   8    100.00     7.67       beer + beer      flying_scotsman.txt
   9    100.00     4.82       wine + wine      ferreira_port_tawny.txt
  10    100.00     3.90    tedtalk + tedtalk   0325NellieMcKay.txt
  11    100.00     3.47    tedtalk + tedtalk   0296NellieMcKay.txt
  12    100.00     3.02    tedtalk + tedtalk   1508GabrielBarcia-Colombo.txt
  13    100.00     2.73    tedtalk + tedtalk   0383Rives.txt
  14    100.00     2.12    tedtalk + tedtalk   0109EddiReader.txt
  15    100.00     1.58    tedtalk + tedtalk   0287NellieMcKay.txt
  16    100.00     1.46    tedtalk + tedtalk   0551CarolynPorco.txt
  17    100.00     1.41    tedtalk + tedtalk   0849ThomasDolby.txt
  18    100.00     0.45    leaflet + leaflet   Adenocor.txt
```

```
  19     99.08    20.12    leaflet + leaflet   Zaditen_Elixir.txt
  20     99.04    35.12    leaflet + leaflet   Cafergot_Tablets.txt
  21     99.04    15.67    leaflet + leaflet   Norprolac.txt
  22     98.97    25.57    leaflet + leaflet   Sandocal_1000.txt
  23     98.36    13.49    leaflet + leaflet   Neoral_Soft_Gelatin_Caps.txt
  24     98.32    15.20    leaflet + leaflet   Sandimmun_Capsules.txt
  25     98.31     7.00    tedtalk + tedtalk   0823NatalieMerchant.txt
  26     98.14    29.83    leaflet + leaflet   Trifyba.txt
  27     97.70    19.63    leaflet + leaflet   Distaclor_Suspension.txt
  28     97.68    11.19    leaflet + leaflet   Tagamet_Effervescent_Tabs.txt
  29     97.61    19.20    leaflet + leaflet   Diclomax_SR.txt
  30     97.56    17.22    leaflet + leaflet   Cordarone_X_Tablets.txt
  31     96.99    16.26    leaflet + leaflet   Sporanox_Capsules.txt
  32     96.85    29.98    leaflet + leaflet   Zaditen_Tablets.txt
  33     96.77    15.45    leaflet + leaflet   Dobutrex.txt
  34     96.72    20.79    leaflet + leaflet   Distaclor_MR.txt
  35     96.70     9.26    leaflet + leaflet   Tagamet_Infusion.txt
  36     96.63    16.31    leaflet + leaflet   Monomax_SR_40.txt
  37     96.49    16.30    leaflet + leaflet   Merbentyl_Syrup.txt
  38     96.07    10.75    leaflet + leaflet   Ecostatin_Cream.txt
  39     96.03    25.45    leaflet + leaflet   Syntopressin.txt
  40     96.02    18.49    leaflet + leaflet   Immukin.txt

[.... many lines omitted ....]

1177     47.70     7.83       ares + ares      A_RES_56_94-en.txt
1178     47.62     0.78         wc - tedtalk   1476BeebanKidron.txt
1179     47.34    11.27       ares + ares      A_RES_56_35-en.txt
1180     47.33    19.58       ares + ares      A_RES_56_69-en.txt
1181     47.06    12.31       wine + wine      paris_street.txt
1182     46.84    11.54       ares + ares      A_RES_55_38-en.txt
1183     46.64    11.79       ares + ares      A_RES_56_154-en.txt
1184     46.52    15.33       ares + ares      A_RES_55_175-en.txt
1185     46.24     1.59    tedtalk + tedtalk   1721LiuBolin.txt
1186     45.72     1.45    tedtalk + tedtalk   0464JoseAntonioAbreu.txt
1187     45.45    11.76       ares + ares      A_RES_55_11-en.txt
1188     45.22     1.78    tedtalk + tedtalk   1003StefanWolff.txt
1189     45.06     2.53       sres + sres      S_RES_13822001-en.txt
1190     44.62     0.95    tedtalk + tedtalk   1019BartWeetjens.txt
1191     44.37     2.11    tedtalk + tedtalk   1645BoghumaKabisenTitanji.txt
1192     43.51     7.20       ares + ares      A_RES_55_235-en.txt
1193     43.36     1.28    tedtalk + tedtalk   1800EleanorLongden.txt
1194     42.72     1.80    tedtalk + tedtalk   1121RogerEbert.txt
1195     40.98    10.07       sres - ares      A_RES_55_55-en.txt
1196     40.91     1.91       wine + wine      le_provenance_cotes_de_provence.txt
1197     40.30     1.61    tedtalk + tedtalk   0558LizColeman.txt
1198     38.78     1.54    tedtalk + tedtalk   1252NathalieMiebach.txt
1199     38.01    17.57       ares + ares      A_RES_57_271-en.txt
1200     34.91     1.50    tedtalk - leaflet   Serevent_Diskhaler.txt
1201      0.00     0.00    tedtalk - wine      oxford_landing_2011_sauvignon_blanc.txt
1202      0.00     0.00    tedtalk + tedtalk   1740JohnLegend.txt
1203      0.00     0.00    tedtalk + tedtalk   1172OnyxAshanti.txt
1204      0.00     0.00    tedtalk + tedtalk   0639ImogenHeap.txt
1205      0.00     0.00    tedtalk + tedtalk   0115RachelleGarniez.txt
1206      0.00     0.00    tedtalk + tedtalk   0099JillSobule.txt

Confusion matrix :

Truecat =            ares    beer leaflet    sres tedtalk      wc    wine
Predcat : ares        284       0       0       0       0       0       0
Predcat : beer          0      27       0       0       0       0       1
Predcat : leaflet       0       0     170       0       1       0       0
Predcat : sres          7       0       0     112       0       0       0
Predcat : tedtalk       0       0       1       0     551       0       1
Predcat : wc            0       0       1       0       5      24       0
Predcat : wine          0       1       0       0       0       0      20

Kappa value =  0.9787
Precision (%) by category :
ares        100.0
beer        96.4286
leaflet     99.4152
sres        94.1176
tedtalk     99.6383
wc          80.0
wine        95.2381
Recall (%) by category :
```

```
ares        97.5945
beer        96.4286
leaflet     98.8372
sres        100.0
tedtalk     98.9228
wc          100.0
wine        90.9091

cases = 1206
cases with unseen category labels = 0
hits =  1188
percent hits = 98.51

Number of non-null instances = 1200
Correct decisions in such cases = 1183
Percent of such cases correct = 98.58
```

Below are copied the first and last lines of the classification ranking to illustrate the format.

```
   1    100.00    20.86    leaflet + leaflet   Metrogel.txt
1206      0.00     0.00    tedtalk + tedtalk   0099JillSobule.txt
```

Here the first item, which can be regarded as the most confident attribution, has a relative percentage coverage of 100 and an absolute coverage of 20.86 percent. This means that the 3:6-grams from the category giving the highest coverage score to this file (leaflet) covered 20.86% of the characters in this text. Furthermore, no other category covered any of this (rather short) text file, since the relative coverage was 100 percent. The string "leaflet + leaflet" means that the predicted category (before the plus sign) was bottlab as was the true category (after the plus sign).

The 1206th item concerns a TED talk (number 0099 by Jill Souboule) which wasn't covered by any 3:6-grams from any of the seven text categories. (It is in the teds subdirectory, so you can have a look to see why). The reason it was given a plus-sign (correct assignment) is because in the event of zero coverage, the most common class is assigned, which in this example was tedtalk. At ranks 1200 and 1201 are a couple of actual mistakes, which are marked with a minus sign. (Categories unseen in the training n-grams would receive a question mark.)

Following the main listing is a confusion matrix along with some associated statistics. This shows quite a good level of classification accuracy, 98.51 percent correct overall, or 98.58 percent excluding null cases with zero coverage. In other words, these text types are quite distinct from each other in their usage of repeated token sequences. This success rate is achieved on a genuine holdout sample. Even the beer and wine labels are relatively well distinguished, as are the General Assembly and Security Council resolutions. With text types based on content, especially if those text types are relatively formulaic, this method does well. For investigating authorship, however, and more generally with text types of similar registers &/or varied content, it is better to use shorter n-grams, e.g. 1:4-grams, if text classification is your main objective. 1-grams, of course, are those old standbys, single tokens (usually words).

To Conclude: Corpora and Collocades

Many researchers have wrestled with the problem of improperly fragmented n-grams, or, to put it another way, the question "how long is a string of pieces?". The method embodied in formulex.py is, I believe, novel, though it has much in common with the "Serial Cascading Algorithm" described in O'Donnell (2011) and attributed to Catherine Smith. Another approach is reported by Pezik (2015) based on subsumption rates and the logarithm of the relative frequencies of the component tokens in each n-gram. However, the formulex algorithm is simpler and has no fixed upper limit on the length of the sequences produced. Thus it leads to nontrivially differing results.

As for what have here been termed 'collocades', they are obviously akin to the 'collocational chains' discussed in Daudaravičius & Marcinkevičienė (2004) and further elaborated by Gries & Mukhurjee

(2010). However, Gries & Mukhurjee use a rather complex statistical criterion, 'lexical gravity' -- related to the idea of 'lexical attraction' introduced by Yuret (1998) -- to define them, and remain within the context of n-gram lists. It seems to me that the idea of textual 'coverage' is simpler to compute and, in my view, easier to explain; and again yields somewhat different results. Thus I believe it is worth having a separate term for such subsequences.

Of course, eventually, it would be highly desirable to find a way of integrating clearly related elements of the formulexicon such as

"please read this leaflet carefully before taking your tablets"

and

"please read this leaflet carefully before you take your medicine"

into a form that reveals their relatedness (a kind of micro-grammar), but that is a challenge for another day....

Acknowledgements

Thanks to Lukasz Grabowski for drawing me into the maelstrom that is research into formulaic language (;-) and for providing the Patient Information Leaflets and informing me where to find the cord corpus. I am also grateful to Lukasz Grabowski and Phoenix Lam for persuading me that the output of flicshow should be multi-coloured rather than just dichromatic, which definitely is an improvement. Thanks also to the Python team for creating and maintaining such a splendid programming language.

And 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).

Bouayad-Agha, N. & Kilgarriff, A. 1999. “Duplication in Corpora” In *Proceedings of the 2nd CLUK Colloquium.* Colchester, Essex, 11-12 Jan 1999. Available at: http://www.kilgarriff.co.uk/Publications/1999-BouayadAghaKilg-CLUK.pdf

Chomsky, N. 1972. *Language and Mind* [enlarged edition]. New York: Harcourt Brace Jovanovich.

Daudaravičius, V. & Marcinkevičienė, R. 2004. Gravity counts for the boundaries of collocations. *International Journal of Corpus Linguistics*, 9 (2), 321–348.

Gries, Stefan Th. and Joybrato Mukherjee. 2010. Lexical gravity across varieties of English: An ICE-based study of n-grams in Asian Englishes. *International Journal of Corpus Linguistics* 15 (4): 520–548.

O'Donnell, M.B. 2011. The adjusted frequency list: A method to produce cluster-sensitive frequency lists. *ICAME Journal*, 35, 117-134.

Pezik, P. (2015). Using n-gram independence to identify discourse-functional lexical units in spoken learner corpus data. *International Journal of Corpus Linguistics*, 1(2), 242-255. doi 10.1075/ijclr.1.2.03pez.

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

Wray, A. 2002. *Formulaic language and the lexicon*. Cambridge: Cambridge University Press.

Yuret, D. 1998. *Discovery of linguistic relations using lexical attraction*. Ph.D. dissertation, Department of Computer Science and Electrical Engineering, MIT.

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 also need a third column, called "doctype" by default. This gives each text a category label.

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

As an example, the flicmeta metafile (formulib\mets\flicmeta.txt) is listed below.

```
prepath filename       doctype
C:\formulib\samples\ares\     A_RES_56_259-en.txt    ares
C:\formulib\samples\ares\     A_RES_56_59-en.txt     ares
C:\formulib\samples\ares\     A_RES_55_20-en.txt     ares
C:\formulib\samples\ares\     A_RES_56_24-en.txt     ares
C:\formulib\samples\cord\     cord2402.txt   cord
C:\formulib\samples\cord\     cord2140.txt   cord
C:\formulib\samples\pils\     Sustanon.txt   leaflet
C:\formulib\samples\pils\     Trasidrex.txt  leaflet
C:\formulib\samples\sres\     S_RES_14912003-en.txt  sres
C:\formulib\samples\sres\     S_RES_14672003-en.txt  sres
C:\formulib\samples\teds\     0918JulianAssange.txt  tedtalk
C:\formulib\samples\teds\     0246TodMachover+DanEllsey.txt tedtalk
C:\formulib\samples\wc\       SpionKop.txt   wc
C:\formulib\samples\wc\       wc060731.txt   wc
C:\formulib\samples\wc\       wc400820.txt   wc
C:\formulib\samples\wc\       wc411226.txt   wc
C:\formulib\samples\wc\       wc281024.txt   wc
C:\formulib\samples\wc\       wc461118.txt   wc
C:\formulib\samples\wc\       MarlPref3308.TXT       wc
c:\formulib\samples\teds\     1548MarkForsyth.txt    tedtalk
c:\formulib\samples\   FictZolaE_Germinal_VII2_EN.txt       queried
c:\formulib\samples\   MarxMarxK_ComMan_01_EN.txt    queried
```

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 **outgrams**.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.
dropsubs	0	This is normally 0, but if it is 1 then during output onto the n-gram file any n-gram that is a subsequence of one that has already been written will be skipped. (Rather drastic in its effects.)
gramdata	jobname with "_gram.dat" appended	File specification of file onto which n-gram data will be written.
gramlist	jobname with "_list.txt" appended	This specifies the file onto which n-grams ordered by frequency only (with various sizes intermingled) for each category of text will be written. Default value is the jobname followed by "_list.txt".
jobname	[outgrams]	This gives the job a name. Any text string can be the value. It isn't necessary but it is advisable 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. If none is given, the program's name will be used as jobname.
maxiglen	5	This specifies the longest n-gram to be considered. It can't be more than 9.
maxtops	20	This specifies the number of "stop-words" to be found and used. The only effect of having more than zero is that any n-gram consisting only of stop-words will be omitted. (See main text.)
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.
miniglen	2	This specifies the shortest n-gram size that will be considered. The minimum is 1, which wouldn't make sense when looking at formulaic language, though (if using taverns.py as a classifier) might make sense in the context of authorship attribution.
outfile	outgrams.txt	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.
snipsize	115	This gives the size of a text block, in tokens, to be used when calculating pervasiveness to discover stop-words.
targvar	doctype	This should be the name of the column in the metafile containing the category labels.
topgrams	100	This specifies how many of the highest-scoring n-grams (of each

		size from maxiglen down to miniglen) will be written to the gramdata file for each text category.
trainmet	[None]	This should be the full path specification of a metafile that indicates the text files that belong to the training corpus, along with their category memberships. If both metafile & trainmet are specified, trainmet takes priority.
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.

Parameters specific to **formulex**.py

Parameter	**Default value**	**Function**
gramdata	jobname with "_gram.dat" appended	Full filepath name where n-grams have been written by outgrams.py.
outforms	100	Number of collocades to be written on the "_flab.txt" file.
testmeta	[None]	This should be the full path specification of a metafile that indicates the text files that of a test or holdout corpus, along with their category memberships. (Optional.)

Parameters specific to **flicshow**.py

Parameter	**Default value**	**Function**
gramdata	jobname with "_gram.dat" appended	Full filepath name where n-grams have been written by outgrams.py.
huemode	0	Colour-coding option: 0 indicating scoring by longest covering n-gram, 1 indicating covering by number of n-grams covering each token.
metaflic	[None]	This should be the full path specification of a metafile that indicates the files to be processed and written as html flic files (Formulaic Language In Context) onto a subfolder called html of the outpath folder.

Parameters specific to **taverns**.py

Parameter	**Default value**	**Function**
contrast	0	Set this to 1 to make the program create an aggregate table of n-grams from other categories in contrast to each category's common n-gram list and use coverage by items from this anti-list to cancel coverage by that category's own n-grams. (Doesn't seem to have a worthwhile effect in practice.)

Appendix 3 : Sample Corpora

ARES.
These 700 documents are resolutions of the United Nations General Assembly sessions 54 to 57, four dated 1999 and the rest from the years 2000-2003. They were extracted from the publications of the United Nations, collated by DFKI GmbH (Deutsches Forschungszentrum für Künstliche Intelligenz), available from www.euromatrixplus.eu/multi-UN/. A total of 720 documents were extracted but the 20 shortest were excluded.

BOTTLABS.
This metafile describes a small corpus of "back label" texts from beverage bottles (mostly wine or beer). It is being collected by the author and will be updated periodically with each release of formulib.

CORD.
5,768 documents from the EU Commission Community Research & Development Information Service (CORDIS). Obtained from
http://psi.amu.edu.pl/en/index.php?title=Parallel_Corpora
by selecting all texts of 196 words or more from the English collection.

PILS.
The patient information leaflets (PILs) 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 and 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 selection as near-duplicates.

SRES.
This corpus contains 275 resolutions passed by the United Nations Security Council in the period 2000-2004. The texts were extracted from the publications of the United Nations, collated by DFKI GmbH (Deutsches Forschungszentrum für Künstliche Intelligenz) www.dfki.de , available from www.euromatrixplus.eu/multi-UN/. The SRES texts cover a slightly longer period than that of ARES since fewer Security Council resolutions are issued each year.

TEDS.
1555 transcripts in English of talks given as part of the TED initiative (www.ted.com). Obtained from collection held at WIT3 website https://wit3.fbk.eu .

WC.
The 55 texts by Winston Churchill (WC) cover a period of over 60 years and thus deal with a wide range of topics. Most have a political or historical focus, but as Churchill was a Nobel laureate in literature we assume his command of English was more creative than average. The texts comprise 45 speeches sourced from http://www.winstonchurchill.org/learn/speeches/speeches-of-winston-churchill as well as his Nobel prize acceptance speech, 2 individual chapters and 2 prefaces from his four-volume biography of Marlborough, a chapter from his 1897 novel Savrola and some occasional pieces of journalism found through Project Gutenberg.
https://www.gutenberg.org/browse/authors/c