Full-text corpus data

Database is one of the three data formats. When you purchase the data, you purchase rights to all three formats, and you can download whichever ones you want.

This format is composed of three tables:

1. [corpus]: one line for each word in the corpus, showing: the textID, the ID (offset value: 293, 294, 295, etc), and then an integer value for the word (wordID).
2. [lexicon]: information on each wordID: word (*walked*), lemma (*walk*), and part of speech (*vvd*).
3. [sources]: information on each textID: genre or country, source, title, etc,

This format allows you to carry out extremely powerful queries on the corpora. You can search by word form, lemma, or part of speech. You can also limit the search to just certain texts in the corpus via JOINs on the [sources] table (e.g. just Magazines-Sports in COCA, Great Britain in GloWbE, fiction from the 1870s in COHA, a particular website in NOW, or pages with a given word in the title in Wikipedia), . You can also modify the [lexicon] table to add whatever additional features you would like for each wordID, or modify the [sources] table to create your own sub-corpora.

corpus table

textID	ID	wordID
2040250	110933753	848
2040250	110933754	3
2040250	110933755	560
2040250	110933756	459620
2040250	110933757	6891
2040250	110933758	10
2040251	110933779	7140251
2040251	110933780	11850
2040251	110933781	187678
2040251	110933782	26
2040251	110933783	19957
2040251	110933784	19
2040251	110933785	41
2040251	110933786	64
2040251	110933787	160
2040251	110933788	4
2040251	110933789	4155

lexicon table

wordID	word	lemma	PoS
71186	swab	swab	nn1
77653	swag	swag	nn1
36155	swagger	swagger	nn1
62562	Swahili	swahili	jj_nn1
44807	Swain	swain	np1
13782	swallow	swallow	vvi
37384	swallow	swallow	vv0
44376	swallow	swallow	nn1@
73593	swallow	swallow	nn1@_vv0
16873	swallowed	swallow	vvd
22661	swallowed	swallow	vvn
36293	swallowed	swallow	vvd_vvn
23040	swallowing	swallow	vvg
39247	swallows	swallow	vvz
57594	swallows	swallow	nn2
14960	swam	swim	vvd
15768	swamp	swamp	nn1

sources table

COCA

textID	genre	sub-genre	year	sourceTitle	textTitle
2040250	MAG	Religion	2007	Source_A	Title_L
2040251	MAG	Sports	2007	Source_B	Title_M
2040252	NEWS	Financial	2012	Source_C	Title_N
2040253	ACAD	Science	1993	Source_D	Title_O
2040254	FIC	MovieScript	1997	Source_E	Title_P
2040255	SPOK	CNN	2001	Source_F	Title_Q
2040256	NEWS	Financial	2012	Source_G	Title_R

COHA

textID	Year	genre	sourceTitle	textTitle
728282	1837	FIC	Source_A	SampleTitle_N
728283	1872	FIC	Source_B	SampleTitle_0
728284	1904	NF	Source_C	SampleTitle_P
728285	1938	MAG	Source_D	SampleTitle_Q
728286	1959	NEWS	Source_E	SampleTitle_R
728287	1987	MAG	Source_F	SampleTitle_S

GloWbE

textID	country	genre	url	textTitle
3282569	AU	BLOG	SampleURL_A	SampleTitle_N
3282570	IN	BLOG	SampleURL_B	SampleTitle_0
3282571	US	GENL	SampleURL_C	SampleTitle_P
3282572	GB	BLOG	SampleURL_D	SampleTitle_Q
3282573	IE	GENL	SampleURL_E	SampleTitle_R
3282574	NZ	GENL	SampleURL_F	SampleTitle_S
3282575	SG	BLOG	SampleURL_G	SampleTitle_T

There are similar tables for NOW, Wikipedia, and Spanish

NOTE: The database format assumes that you know SQL (Structured Query Language), and that you can create the tables, populate them with the downloaded data, and (most importantly) run the SQL queries to extract the data. Please do not use this format unless you are well-acquainted with databases and SQL. The following are just a handful of SQL queries that you can run on the data, but there is of course no limit to what you can do.

1. Find 1000 most frequent nouns in (COCA) ACAD-Science

```
select count(*),lex.word
from lexicon as lex, sources, corpus where
sources.genre = 'ACAD' and sources.sub-genre = 'Science' and
sources.textID = corpus.textID and
lex.pos like 'nn%' and
lex.wordID = corpus.wordID
group by lex.word
order by count(*) desc
```

2. Find top 500 strings of get V-ed (e.g. *got married, gets paid*)
(using "runtime" self-join on corpus; much faster with multi-column table; see below)

```
select count(*),lex1.word, lex2.word
from lexicon as lex1, lexicon as lex2, corpus as corpus1, corpus as corpus where
lex1.lemma = 'get' and
lex2.pos like 'v_n%' and
lex1.wordID = corpus1.wordID and
lex2.wordID = corpus.wordID and
corpus.ID = corpus1.ID + 1
group by lex1.word, lex2.word
order by count(*) desc
```

3. Find top 500 3-grams, with *point* in the second position
(using "runtime" self-join on corpus; much faster with multi-column table; see below)

```
select count(*),lex1.word, lex2.word, lex3.word
from lexicon as lex1, lexicon as lex2, lexicon as lex3, corpus as corpus1, corpus as corpus, corpus as corpus3 where
lex2.word = 'point' and lex2.wordID = corpus.wordID and
lex1.wordID = corpus1.wordID and
lex3.wordID = corpus3.wordID and
group by lex1.word, lex2.word, lex3.word
order by count(*) desc
```

Note: rather than using self-joins (as in #2 and 3 above) the architecture for the corpora from English-Corpora.org has tables like that shown below. The [w5] column here corresponds to the [wordID] column in the [corpus] table above, but a massive self-join has been done on this table (as the corpus was created; not as each query is run) to create "adjacent" [w1]-[w4] and [w6]-[w9] columns. As a result, the four preceding and four following words are already on the row when one searches [w5]. With the full-text data, you can create similar tables yourself.

w1	w2	w3	w4	**w5**	w6	w7	w8	w9
43	3	858	5	**432**	3319	9	132	2876
3	858	5	432	**3319**	9	132	2876	3643
858	5	432	3319	**9**	132	2876	3643	5
5	432	3319	9	**132**	2876	3643	5	1729
432	3319	9	132	**2876**	3643	5	1729	72
3319	9	132	2876	**3643**	5	1729	72	43
9	132	2876	3643	**5**	1729	72	43	21887
132	2876	3643	5	**1729**	72	43	21887	746929
2876	3643	5	1729	**72**	43	21887	746929	676
3643	5	1729	72	**43**	21887	746929	676	62900

This allows for much faster queries (than self-joins at SQL runtime). For example, to find the most frequent collocates for a given word, the SQL query would be:

4. Find top 200 noun collocates of *break* as a verb, in the four "slots" after *break* (columns w6-w9 above).

```
select top 200 count(*),w6 from (
SELECT x.w6 FROM corpus as x, lexicon as x1 where x1.lemma like 'break' and x1.pos like 'v%' and x.w5 = x1.wordID UNION ALL
SELECT x.w7 FROM corpus as x, lexicon as x1 where x1.lemma like 'break' and x1.pos like 'v%' and x.w5 = x1.wordID UNION ALL
SELECT x.w8 FROM corpus as x, lexicon as x1 where x1.lemma like 'break' and x1.pos like 'v%' and x.w5 = x1.wordID UNION ALL
SELECT x.w9 FROM corpus as x, lexicon as x1 where x1.lemma like 'break' and x1.pos like 'v%' and x.w5 = x1.wordID
) a, lexicon b where
b.pos like 'nn%' and
a.word2 = b.wordID
```

```
group by a.w6
order by count(*) desc
```

On a fairly fast machine, this will only take about two seconds for COCA (440 million words)