WORD FREQUENCY AND KEYWORD EXTRACTION

AHRC ICT Methods Network Expert Seminar on Linguistics
8 September 2006, Lancaster University, UK

Word frequency in context: Alternative architectures for examining related words, register variation, and historical change

Mark Davies, *Brigham Young University, USA.*

Keywords
mega-corpora, relational database, n-gram, integration, frequency, slot-based queries, collocates.

Abstract
The use of relational databases that are composed of the frequency of n-gram in a given corpus allows users to quickly and easily examine word frequency. Perhaps the first large corpus to use such an approach was the 100 million word *Corpus del Español*, which was created in 2002 (www.corpusdelespanol.org/). This was followed by two BNC-based 100 million word corpora that were modelled on the same architecture: *Phrases in English* (pie.usna.edu) and *Variation in English Words and Phrases* (*VIEW*; view.byu.edu), as well as a 40 million word *Corpus of Historical English* (view.byu.edu/che)

The relational database/n-grams architecture allows simple word frequency queries such as the following (all of which can be carried out on a 100 million word corpus in 1-2 seconds):

- Overall frequency of a given word, set of words, phrase, or substring in the corpus
- 'Slot-based' queries; e.g. the most common nouns one 'slot' after *mysterious*, or z-score ranked words immediately preceding *chair*
- Wide-range collocates; e.g. the most common nouns within a ten word window (left or right) of *string* or *broken*

In addition, however, the architecture that we have used for *VIEW* and the *Corpus of Historical English* allows several other types of queries that cannot be carried out directly with competing architectures (e.g. *SARA/XARA*, the *IMS Corpus Workbench*, or the *Phrases in English* architecture), including the following:

- Comparison of frequency with related words; e.g. nouns occurring immediately after *utter* but not after *complete* or *sheer*, or adjectives within ten words of *woman* but not *man*
- One simple query to find the frequency of words in separate databases, such as user-defined, customized lists (clothing, emotions, technology terms, etc) or synsets from WordNet
- Register variation; e.g. all verbs or all words ending in *ble or all three-word lexical bundles that are more common in academic texts than in fiction, or in legal or medical texts
- Historical variation; e.g. words, phrases, or collocates of a given word or part of speech, which are more common in the 1900s than in the 1800s

Finally, even within the relational database/n-grams approach, there are competing architectures that favour certain types of queries over others, and we will briefly consider some of these issues.

Introduction

Since the advent of 'mega-corpora' that are 100 million words in size or larger, there have been challenges in terms of economically extracting large amounts of data. For example, several years ago it was sufficient to create a query engine that would perform a linear scan of the text, and such an architecture might return the results from a one million word corpus in 1-2 seconds. Using that same architecture, however, a similar query on a 100 million word corpus might take 100-200 seconds. As a result of these performance issues, in the last 10-15 years a number of alternate architectures have been developed. These include the use of large numbers of indices that contain offset values for each word in the corpus and the use of large hash operations to find nearby words (e.g. SARA/XAIRA: Burnage and Dunlop 1993, Burnard 2000) and the relational database architecture of the IMS Corpus Workbench (Christ 1994).

During the past five years, we have employed a modified relational database architecture for a number of corpora that we have created. In contrast to the IMS Corpus Workbench approach, however, these corpora rely heavily on an 'n-gram' architecture, which will be one of the major topics of this paper. These corpora include the 100 million word Corpus del Espanol (www.corpusdelespanol.org) and a new architecture and interface for the 100 million word British National Corpus (view.byu.edu), both of which will be discussed herein.

As with some other competing architectures, this relational database/n-gram approach allows queries like the following:

- the overall frequency of a given word or phrase in the corpus (*mysterious, blue skies*)
- the frequency words with a given substring (*able, *heart*, etc)
- queries involving part of speech or lemma (e.g. *utter* NN1, *as* ADJ *as, ADV VVD* '*barely realized'*)

Unlike some other architectures, however, our approach is quite fast. Any of the preceding queries on a 100 million word corpus would take less than one second.

In addition, as we will see, our approach allows a number of types of query that would be difficult or impossible to carry out directly in one step with competing architectures. These include – but certainly are not limited to – the following:

- Comparison of frequency with related words; e.g. nouns occurring immediately after *utter* but not after *complete* or *sheer*, or adjectives within ten words of *woman* but not *man*
- One simple query to find the frequency of words in separate databases, such as user-defined, customized lists (clothing, emotions, technology terms, etc.) or synsets from WordNet
- Register variation; e.g. all words ending in **icity*, or all verbs, or all three-word lexical bundles, which are more common in academic texts than in works of fiction, or in legal or medical texts
- Historical variation; e.g. all words, phrases, or collocates of a given word or part of speech, which are more common in the 1900s than in the 1800s

In the discussion that follows, we will first present the basic architecture (relational databases and n-grams) and provide concrete examples of some of the types of queries that this architecture allows. We will then discuss shortcomings of this architecture, and consider how these issues have been handled in some of the newer interfaces that we have created, such as the VIEW interface for the BNC (http://view.byu.edu).

A simple n-gram architecture

Let us first consider the 'first-generation' approach to relational databases and n-grams, which was used for the Corpus del Espanol (http://www.corpusdelespanol.org) that we created in 2001-02 (see Davies 2005a, 2005b), and the subsequent BNC-based 'Phrases in English' database and interface that was based on the same architecture and which was created by William Fletcher in 2003 (http://pie.usna.edu).

In this approach, one uses a program to create the n-grams of a given corpus, such as the WordList module of WordSmith (Scott 2004) or the KFN-gram program (http://miniappolis.com/KWiCFinder/kfNgramHelp.html). For example, with WordSmith, one would simply create separate lists of all of the 1-grams, 2-grams, 3-grams, etc. in the corpus, and then import these into a relational database. In the case of 3-grams for the BNC, for example, a small section of the 3-grams table would look like the following:

Table 1. Example of 3-grams where lem1 = 'break' and word2 = 'the'

FREQ	WORD1	WORD2	WORD#
106	breaking	the	law
98	break	the	law
56	broke	the	silence
53	break	the	news
46	broke	the	news
40	break	the	deadlock
24	broken	the	law
23	break	the	habit

Each unique three-word string in the corpus appears in the database, with its associated frequency. For example, in the BNC *breaking the law* occurs 106 times, *broke the law* occurs 56 times, and so on.

It is also possible to create frequency tables that include POS (part of speech) and lemmatization information as well. In this case, the table might look like the following:

Table 2. Example of 3-grams where LEM1 = 'break' and WORD2 = 'the'

FREQ	WORD1	LEM1	POS1	W2	LEM2	POS2	W3	LEM3	POS3
106	breaking	break	VVG	the	the	AT0	law	law	NN1
98	break	break	VVI	the	the	AT0	law	law	NN1
56	broke	break	VVD	the	the	AT0	silence	silence	NN1
53	break	break	VVI	the	the	AT0	news	news	NN1
46	broke	break	VVD	the	the	AT0	news	news	NN1
40	break	break	VVI	the	the	AT0	deadlock	deadlock	NN1
24	broken	break	VVN	the	the	AT0	law	law	NN1
23	break	break	VVI	the	the	AT0	habit	habit	NN1

With these n-gram/frequency tables, it is a relatively simple process to use SQL queries to extract the desired data. For example, to extract the 100 most common singular nouns (NN1) in the BNC, the SQL query would be the following:

(1)
```
SELECT TOP 100 * FROM [TABLE_NAME] WHERE
POS1 = 'NN1'
ORDER BY FREQ DESC
```

To select the 100 most common three-word strings where the first word is a form of *break,* the second word is *a* or *the*, and the third word is a noun, the SQL query would be the following, and the results would be those seen in Table 2 above:

(2)
SELECT TOP 100 * FROM [TABLE_NAME] WHERE
LEM1 = 'BREAK' AND
WORD2 = 'THE' AND
POS3 LIKE 'NN%'
ORDER BY FREQ DESC

Either of these two queries would take less than half a second to retrieve the 100 most frequent matching words or strings from the 100 million word corpus. This is the approach used in our Corpus del Espanol and in the Phrases in English interface, and it represents an early approach to the use of n-grams.

Accounting for register or historical variation

There is a serious problem, however, associated with a strict n-gram architecture. Once the frequencies are calculated for each unique n-gram in the corpus, one then loses all contextual information for that n-gram – in other words, in which part of the corpus each of these n-grams occur. For example, Table 2 above shows that *breaking the law* occurs 106 times in the corpus, but at this point we have no idea how many of these are in the SPOKEN texts, or FICTION, or NEWSPAPERS. Therefore, using this approach it would probably be impossible to find the most frequent words or phrases in a given register, or to compare the frequency of words or phrases in two competing (sets of) registers.

There is a way around the lack of context for each n-gram, however. One could calculate the overall n-gram frequency for a set of different registers (as in Table 2), and then create n-gram frequency tables for each register individually. One would then 'merge' the information from the 'register' tables into the overall frequency table, which would contain separate columns (for each n-gram) showing the frequency in each register. In other words, the resulting table might look like the following

Table 3. Example of 3-grams where LEM1 = 'break' and WORD2 = 'the'

WORD1	LEM1	POS1	…	W3	LEM3	POS3	REG1	REG2	REG3
breaking	break	VVG	…	law	law	NN1	x1	y1	z1
break	break	VVI	…	law	law	NN1	x2	y2	z2
broke	break	VVD	…	silence	silence	NN1	x3	y3	z3
break	break	VVI	…	news	news	NN1	x4	y4	z4
broke	break	VVD	…	news	news	NN1	x5	y5	z5
break	break	VVI	…	deadlock	deadlock	NN1	x6	y6	z6
broken	break	VVN	…	law	law	NN1	x7	y7	z7
break	break	VVI	…	habit	habit	NN1	x8	y8	z8

This is in fact the approach taken in the construction of the 100 million word Corpus del Espanol. For each n-gram, there are columns that show the frequency of the string in each century from the 1200s to the 1900s (x12-x19 below). There are also separate columns containing the frequency of the n-gram in each of the three registers SPOKEN, FICTION, and NON-FICTION in the 1900s. The following is a small section of bi-grams table, containing a few of the n-grams that match the query NOUN + lemma DURO 'hard N':

Table 4. '*Hard* N' (N *duro*) in the Corpus del Español, by century

WORD1	WORD2	x12	…	x17	x18	x19	x19SP	x19FIC	x19NF
MANO	DURA	0	…	0	10	22	6	9	7
LÍNEA	DURA	0	…	0	0	12	0	3	9
SER	DURO	0	…	1	9	10	7	1	2
CUELLO	DURO	1	…	0	0	10	9	1	0
MADERA	DURA	0	…	0	5	10	7	0	3
CARA	DURA	0	…	0	3	10	6	4	0

The advantage of using such an approach should be readily apparent. Because each n-gram has the associated frequency in each of the different historical periods and the different registers, this frequency information can be accessed directly as part of the query. For example, in the case of the Corpus del Espanol, we can find which nouns occur with *duro* 'hard' in the 1900s, but not in the 1800s. The SQL query would look something like the following (simplified here from how it would appear in the actual database):

(3)
```
SELECT TOP 100 * FROM [TABLE_NAME] WHERE
POS1 = 'NOUN' AND
LEM2 = 'DURO' AND
X19 <> 0 AND
X18 = 0
ORDER BY X19 DESC
```

This gives us results like *línea dura* 'hard line', *disco duro* 'hard drive', and *años duros* 'hard years', etc.

In spite of the advantages of this approach, one problem is that it is quite costly to run the SQL UPDATE commands that copy the frequency information (for tens of millions of n-grams) from each of the separate tables (1200s, 1500s, 1900s-FIC, etc) into the main n-gram tables. In addition, this approach may only be practical when there are a limited number of frequency columns, such as the eleven columns in the Corpus del Espanol (1200s-1900s, and three additional registers for the 1900s). In the case of the BNC, on the other hand, there are nearly 70 different registers, according to the categorization made by David Lee (see http://opinion.nucba.ac.jp/~davidlee/devotedtocorpora/home/BNC_WORLD_INDEX.ZIP).

In summary, the frequency information from each sub-corpus can be quite valuable, in terms of being able to compare between different historical periods, different registers, and so on. However, because of the difficulty in creating such tables, they are not used in some other competing architectures and interfaces. As a result, with these approaches, it is only possible to look at word and phrase frequency across the entire corpus.

The issue of size

In addition to the problem of 'granularity' in terms of frequency in sub-registers, another problem with a strict n-gram approach has to do with the size of the tables. While there are only about 800,000 rows in the [1-grams] table of the BNC (i.e. 800,000+ unique types in the corpus), this increases to about 11 million unique [2-grams] and 40 million unique [3-grams], and it would move towards 90-95 million unique 7-grams for the 100 million words.

The problem with this approach, then, is that the n-gram tables become quite unmanageable in terms of size. Even with efficient clustered indexes on the tables, it may take 10-15 seconds to return the results from a particularly difficult query, such as the most frequent n-grams for [*the NN1 that*]. The second problem is that once we add up all of the different n-gram tables (1-grams + 2-grams + 3-grams, etc), we soon find that the total number of rows in these tables is larger than the total number of words in the corpus, especially if we include the 4-grams through 7-grams.

As a result of the size issue, the approach taken in the construction of the Phrases in English database and interface (http://pie.usna.edu) is to include just those n-grams that occur three times or more. By eliminating from the tables all n-grams that occur just one or two times, the size of the tables is reduced dramatically – by 75% in the case of the 3-grams, and even more for 4-grams through 7-grams. Therefore there is a large performance gain by eliminating all n-grams that occur just once or twice. Unfortunately, with this approach, one also completely loses the ability to analyze these less-common strings. If one is interested in only the highly-frequent n-grams, this is probably not much of problem. But for detailed comparison of sub-corpora, or to see which phrases are entering into or leaving the language, it is problematic to exclude 75% or more of all n-gram types.

As we have seen, there are two basic problems with a strict n-gram approach. First, in terms of sub-corpora, we either have to ignore the context in which each n-gram appears (register, historical period, etc), or else we have to create tables that are quite difficult to construct, by merging in the frequency for each sub-corpus into a column in the main n-gram table. Second, in terms of size and speed, we either create tables that eliminate n-grams that occur just one or two times (and thus lose 75% or more of all unique n-grams), or we have a number of separate n-gram tables (2-grams, 5-grams, etc) whose combined size in terms of rows is probably much larger than the total number of words in the corpus.

An alternate architecture – all sequential n-grams

There is a 'second-generation' approach, however, that avoids both of the problems with a strict n-gram approach. In this architecture, we create one single database table that has as many rows as the total number of words in the corpus. Each row contains a sequential word in the corpus (along with part of speech and other information, if desired). Most likely, these sequential words will be one of the central columns of the table. This column is then surrounded on the left and right by a number of 'contextual' columns, to create a 'context window' for each word in the corpus.

For example, the following table represents the main table in our BNC/VIEW database (http://view.byu.edu). This table contains about 100 million rows (one for each sequential word in the BNC), with each word in corpus (column *word4*) surrounded by three words to the left (*word1*-*word3*) and three words to the right (*word5*-*word7*), as well as the 'word offset' ID and the text from which the n-gram is taken (for the sake of brevity, only *word3*-*word5* are shown here):

Table 5. Sequential n-gram 'tokens'

ID	text	…	word3	pos3	word4	pos4	word5	pos5	…
50891887	EAA	…	a	AT0	small	AJ0	group	NN1	…
50891888	EAA	…	small	AJ0	group	NN1	of	PRF	…
50891889	EAA	…	group	NN1	of	PRF	people	NN0	…
50891890	EAA	…	of	PRF	people	NN0	at	PRP	…
50891891	EAA	…	people	NN0	at	PRP	work	NN1	…
50891892	EAA	…	at	PRP	work	NN1	,	PUN	…
50891893	EAA	…	work	NN1	,	PUN	there	EX0	…
50891894	EAA	…	,	PUN	there	EX0	will	VM0	…
50891895	EAA	…	there	EX0	will	VM0	be	VBI	…
50891896	EAA		will	VM0	be	VBI	significant	AJ0	

In addition to this main sequential n-gram table, there is also a small table that contains 'meta-data' for each of the 4000+ texts in the BNC. For example, the following shows just a few of the columns of information for a handful of entries from this table, centred on the [EAA] text:

Table 6. Text meta information table

text	register	topics	title
EA8	W_commerce	leadership; commerce	Making it happen…
EA9	W_commerce	hotel management; tourism	The hotel receptionist…
EAA	W_soc_science	-	Managing people at…
EAJ	W_soc_science	public law; politics	Public law and political...
EAK	W_nat_science	scientific research	Nature. London: Macmillan…

Let us now consider how these tables can be used to create quick, powerful queries on even large corpora such as the BNC.

Basic frequency data for words, phrases, substrings, and collocations

Basic queries, even on large corpora such as the 100 million British National Corpus are quite fast with the new architecture. They also have the added benefit of including *all* matching strings – not just the most frequent n-grams, as is the case with some other architectures. For example, to find the most frequent nouns following *break* + *a/the* + NOUN (*break the news, break a promise*), the user would enter the following string via the web-based interface. This would be converted to the following SQL command, which would produce the results seen in Table 2 above.

(4)
QUERY: break a/the [nn*]

```
SQL:
SELECT TOP 100
COUNT(*),WORD4,POS4,WORD5,POS5,WORD6,POS6
FROM [TABLE_NAME] WHERE
WORD4 = 'BREAK' AND
WORD5 IN ('THE','A') AND
POS6 LIKE 'NN%'
GROUP BY WORD4,POS4,WORD5,POS5,WORD6,POS6
ORDER BY COUNT(*) DESC
```

Any combination of substrings, words, or part of speech in a seven-word window can be used for the query.

In addition to seeing just raw frequencies, one can also see a ‘relevancy-based‘ display, which is based on a modified z-score. With a simple raw-frequency sorting ([SORT BY: RELEVANCE] in the search form), a query like [* chair] would produce results like *a chair, the chair, his chair*, etc. However, if the user selects [SORT BY: RELEVANCE] in the search form, then the results will be *high-backed chair, sedan chair, wicker chair, swivel chair*, etc. To produce this results set, the script 1) finds the raw percentage for all strings matching [* chair] 2) finds the overall frequency of the words in the [*] slot (e.g. *the, wooden, swivel*, etc), and then 3) divides the first figure by the second. In this case, though, the three-step query still only takes about one half of a second to return the results from the 100 million word corpus.

Comparing related words

A major advantage of storing the n-grams in a relational database is that this architecture lends itself well to frequency comparisons between sets of words. For example, it is possible to find all of the collocates of a given WORD 1 and the collocates of a different WORD 2, and then compare these two lists within the database itself. For example, a user can see the difference between two or more synonyms by finding the collocates that occur with one but not with the other, and all of this can be carried out in just one or two seconds.

As a concrete example, assume that a non-native speaker of English is interested in the difference between *utter, sheer*, and *absolute*. The user would enter the following into the web-based search form:

(5)
QUERY: {utter/sheer/absolute} [n*]

The curly brackets around the three synonyms indicates that the user wants to find the collocates for each of these three words, and then 'group' these collocates together for each individual word. The results would look like the following:

Table 7. Grouping by synonyms:

%	+	-	SHEER
1.00	60	--	weight
1.00	31	--	force
1.00	26	--	luck
1.00	23	--	quantity
1.00	13	--	cliff
1.00	12	--	cliffs
1.00	11	--	coincidence
1.00	10	--	enjoyment

%	+	-	UTTER
1.00	19	--	confusion
1.00	5	--	condemnation
1.00	5	--	devastation
1.00	5	--	disregard
1.00	5	--	helplessness
1.00	4	--	loneliness
1.00	3	--	dejection
1.00	3	--	ruthlessness

%	+	-	ABSOLUTE
1.00	98	--	majority
1.00	84	--	terms
1.00	54	--	zero
1.00	53	--	minimum
1.00	51	--	value
1.00	39	--	egalitarianism
1.00	33	--	right
1.00	27	--	price

This would indicate to the language learner, for example, that *sheer* occurs with *weight* (60 tokens), *force* (31 tokens), and *luck* (26), but that none of these words occur with either *utter* or *absolute*. *Utter*, on the other hand, is the only one of the three synonyms that occurs with *confusion* (10 tokens), *condemnation* (5) and *devastation* (5), and *absolute* is the only synonym with *majority* (98), *terms* (84) and *zero* (54).

To process this query, the script carries out a number of steps: 1. find all nouns following *sheer*; 2. find all nouns following *utter*; 3. find all nouns following *absolute*; 4. stores all of the results for these three queries in a temporary table; 5. runs three separate, sequential SQL queries to find the most frequent collocates for each of these words, which do not occur with either of the other two. (It is also possible to sort by raw frequency with each word, rather than WORD1 vs WORD2 / WORD3, as shown above)

Notice that comparisons such as these might be possible, but they would certainly be much more cumbersome with other architectures. For example, users of those interfaces could carry out a query with WORD 1, and then WORD 2, and so on. They would then copy and paste the results into separate tables of a database, and then (assuming some skill in SQL), carry out cross-table JOINs to determine the relative frequency with the different words. In this case however, the process would probably take at least 3-4 minutes, whereas with our interface it takes only about 1.2 seconds.

Integration with other databases

Another important advantage of the relational database approach is that the central n-gram tables can be integrated with other relational databases. For example, the frequency information from the BNC/VIEW databases can be joined with semantic information from other databases such as WordNet (Fellbaum 1998, Landes et al 1998), or with personalized lists (relating to semantic fields) created by the user.

Let us take a concrete example, which we have already discussed elsewhere (Davies, forthcoming). In order to add a strong semantic component to the VIEW/BNC database, we have imported into a separate database the entire contents of WordNet – a semantically-based hierarchy of hundreds of thousands of words in English. The 'synsets' that make up WordNet indicate roughly equivalent meanings (synonyms), more specific words (hypernyms) and more general words (hyponyms) for a given word, as well as 'parts' (meronyms) of a larger group (e.g. parts of a body) or the larger groups to which a certain word belongs (holonyms; e.g. leg > table / body).

At the most basic level, a user can simply submit as a query something like the following:

(6)
QUERY: [=sad]

The script sees the equal sign and interprets this as a query to find all synonyms of *sad*. The script then retrieves all of the matching words from WordNet and finds the frequency of each of these words in the BNC. Users can also limit the hits by part of speech, and can also intervene before seeing the frequency listing to select just certain synsets (or meanings) from WordNet.

Table 8. WordNet/BNC integration: frequency of synonyms of *sad*

1	SORRY	10767
2	SAD	3322
3	DISTRESSING	359
4	PITIFUL	199
5	DEPLORABLE	144
6	LAMENTABLE	69

The WordNet data can also be integrated into more advanced queries. A user can – with one simple query – compare which nouns occur with the different synonyms of a given word. For example, the following query finds all cases of a noun following a synonym of *bad*:

(7)
QUERY: [=bad] [nn*]

In less than four seconds, the user then sees something similar to the following. (The format on the web interface is somewhat different to the abbreviated listing shown here).

Table 9. Synonyms of [bad] + NOUN

PHRASE	FREQ	PHRASE	FREQ
disgusting thing	16	severe shortage	26
disgusting way	5	severe weather	43
distasteful species	5	severe winter	49
evil empire	10	terrible accident	26
evil eye	24	terrible blow	13
evil influence	9	terrible danger	14
foul language	29	terrible feeling	19
foul mood	21	terrible mistake	48
foul play	72	terrible shock	41
foul temper	14	wicked grin	6
severe blow	44	wicked people	13
severe burn	28	wicked thing	27
severe damage	59	wicked way	14
severe drought	30	wicked witch	12
severe illness	23		

Such collocational data can be very useful for a language learner, who is probably unsure of the precise semantic range of each adjective. The type of listing given above, which shows the most common nouns with each of the adjectives, can easily permit the language learner to make inferences about the semantic differences between each of the competing adjectives. For example, s/he would see that *severe illness* occurs but *wicked illness* does not, and that *terrible mistake* is common, whereas *foul mistake* is not.

Finally, it is again worth noting that such a query would be extremely difficult with an architecture that does not allow user access to other databases that can be tied into the main frequency/n-gram databases. Without our approach, users would likely have to create the list of synonyms in another program, perhaps run queries with each of these words individually, and then collate and sort the results in a spreadsheet. In our approach, all of this is done by the web-based script in less that two seconds.

Register-based queries

As was discussed previously, one serious shortcoming of many other approaches is that it is either difficult or impossible to compare the results of different sub-corpora – for example, different registers or different historical periods. With the newer BNC/VIEW architecture, however, this is both quite simple and quite fast.

Recall that with the 'sequential n-gram' architecture, each word in the corpus appears on its own line in the database:

Table 10. Sequential n-gram table

ID	text	…	word3	pos3	word4	pos4	word5	pos5	…
50891887	EAA	…	a	AT0	small	AJ0	group	NN1	…
50891888	EAA	…	small	AJ0	group	NN1	of	PRF	…
50891889	EAA	…	group	NN1	of	PRF	people	NN0	…
50891890	EAA	…	of	PRF	people	NN0	at	PRP	…
50891891	EAA	…	people	NN0	at	PRP	work	NN1	…
50891892	EAA	…	at	PRP	work	NN1	,	PUN	…
50891893	EAA	…	work	NN1	,	PUN	there	EX0	…
50891894	EAA	…	,	PUN	there	EX0	will	VM0	…
50891895	EAA	…	there	EX0	will	VM0	be	VBI	…
50891896	EAA		will	VM0	be	VBI	significant	AJ0	

As shown previously, there is also a small table that contains 'meta-data' for each of the 4000+ texts in the BNC:

Table 11. Text meta information table

text	register	topics	title
EA8	W_commerce	leadership; commerce	Making it happen…
EA9	W_commerce	hotel management; tourism	The hotel receptionist…
EAA	W_soc_science	-	Managing people at…
EAJ	W_soc_science	public law; politics	Public law and political...
EAK	W_nat_science	scientific research	Nature. London: Macmillan…

To limit the query to one particular register (or set of registers) or to compare between registers, the script relies on a SQL JOIN between these two tables. It looks for all rows that match the n-gram table (Table 10 above), all texts that match the specific register(s) (Table 11 above), and then limits these hits to just those that have a particular text in common (e.g. EAA above). Examples of the types of register-based queries might be queries to find which nouns, or verbs, or three word lexical bundles, or collocates with *chair* occur more in one register (e.g. FICTION) than in another (e.g. ACADEMIC).

For example, suppose that a user wants to find which verbs are more common in legal texts than in academic texts generally. S/he would simply select the following in the web-based form:

(8)
QUERY: [vvi]
REGISTER 1 [w_ac_polit_law_edu] (from pull-down menu)
REGISTER 2 [ACADEMIC]

This searches for all infinitival lexical verbs (VVI) in law texts (w_ac_polit_law_edu), and then all infinitival verbs in ACADEMIC texts as a whole, and then compares the two sets of words. The words – representing verbs that are highly frequent in a legal context – are found in the following table:

Table 12. Searching by register; lexical verbs in legal texts

WORD/PHRASE	TOKENS REG1	TOKENS REG2	PER MIL IN REG1 [4,640,346 WORDS]	PER MIL IN REG2 [10,789,236 WORDS]	RATIO
SUE	331	10	71.33	0.93	76.96
CERTIFY	27	2	5.82	0.19	31.39
ADJOURN	26	2	5.60	0.19	30.23
NOTIFY	38	3	8.19	0.28	29.45
WAIVE	38	3	8.19	0.28	29.45
DISCLOSE	190	16	40.95	1.48	27.61
OVERRULE	23	2	4.96	0.19	26.74
PROHIBIT	53	5	11.42	0.46	24.65
PLEAD	62	6	13.36	0.56	24.03

Again, note that such comparisons would possibly be quite cumbersome for competing architectures. To the degree that these architectures allow queries by sub-corpora (and not all do), users could carry out a query for [VVI] in Register 1 and then a subsequent query in Register 2. They would then copy and paste the results into separate tables in a database, and then carry out cross-table JOINs to determine the relative frequency in the two registers. Again, however, the process would probably take at least 3-4 minutes, whereas with our interface it takes only about two seconds.

Expanding collocate searches

The architecture described to this point works well for determining the frequency of words, substrings, phrases, and 'slot-based' searches (e.g. ADJ + *world*, synonym of [*tired*] + NOUN, etc). One fundamental disadvantage of the architecture, however, is that it is limited by its very nature to strings that occur within a seven word window. This is because that is the number of columns in the 'sequential n-gram' table (see Table 5 above). Therefore, if one wanted to find all the nouns that occur within a wider context of a given adjective, for example, it would not be possible with this architecture.

Recently however, we changed the searching algorithm to allow for the recovery of collocates from much wider context – up to ten words to the left and to the right of a given word. There are several steps in the script that carries out such queries. Let us take a concrete example to see how this would work. Assume that we are looking for adjectives that are collocates – ten words to the left or right – of the word *woman*. First, the script finds all of the ID values (=word offset values in the corpus) for *woman* – more than 22,000 occurrences in the 100 million word corpus – and these are stored in a temp table. We then find all of the adjectives in the corpus whose ID value is between ten more and ten less than the ID values in the temp table. These are placed in a second temp table, and a SQL command then finds the most common words in this table. Overall, the script takes about 5-6 seconds to run, and yields adjectives like *young, old, beautiful, married,* etc.

As with the 'slot-based' queries (e.g. ADJ + *woman*) these collocates can then be compared to the collocates of a competing word, such as *man*, to determine with collocates are used with *woman* much more than with *man* (e.g. *childless, pretty, pregnant, distraught, fragile, desirable*), or more with *man* than with *woman* (e.g. *honourable, reasonable, military, modest, rational*).

Likewise, one can compare collocates across registers, to look for possible polysemy with a given word. For example, one could look for adjectives with *chair* that are more common in fiction than in academic text (e.g. *small, hard, rocking, asleep*) or which are more common in academic text than in fiction (e.g. *senior, philosophical, established, powerful*). Again, this script only takes about 1.5 seconds to produce the list of contrasting collocates.

Conclusion

We hope to have demonstrated two fundamental facts in this paper. First, it is often insightful and advantageous to look at word frequency in context – by register, in historical terms, in syntagmatic terms (collocations) and paradigmatic terms (a given word contrasted with competing words that fill the same slot, such as synonyms). Second, the highly-structured relational databases lend themselves well to the comparison of contexts. Word frequency, then, can be analysed not just as

the overall frequency of a given word or lemma in a certain corpus, but rather as the frequency of words in a wide range of related contexts.

References

Burnage, G. and D. Dunlop, 'Encoding the British National Corpus', in Aarts, J., et al (ed). *English Language Corpora: Design, Analysis and Exploitation. Papers from the 13th International Conference on English Language Research*. (Amsterdam: Rodopi, 1993), 79–95.

Burnard, L., *Reference Guide for the British National Corpus: World Edition* (Oxford: Oxford University Computing Services, 2000).

Christ, O., *The IMS Corpus Workbench Technical Manual* (Stuttgart: Institut fur Maschinelle Sprachverarbeitung, Universitat Stuttgart, 1994).

Davies, M., 'The Advantage of Using Relational Databases for Large Corpora: Speed, Advanced Queries, and Unlimited Annotation', *International Journal of Corpus Linguistics,* 10 (2005), 301–28.

Davies, M. 'Advanced Research on Syntactic and Semantic Change with the Corpus del Español', in Pusch, C., et al (eds), *Romance Corpus Linguistics II: Corpora and Diachronic Linguistics* (Tübingen: Guntar Naar, 2005), 203–14.

Davies, M. 'Semantically-Based Queries with a Joint BNC/WordNet Database', in Facchinetti, R. (ed), *Corpus Linguistics twenty-five years on* (Amsterdam: Rodopi, forthcoming).

Fellbaum, C., (ed.), *WordNet: An Electronic Lexical Database*. (Cambridge, MA: MIT Press, 1998).

Fletcher, W. 'Exploring Words and Phrases from the British National Corpus' *Phrases in English*, [website], (2005) <http://pie.usna.edu>, accessed June 2006.

Landes S., C. Leacock, and R. Tengi, 'Building Semantic Concordances', in Fellbaum, C. (ed), *WordNet: An Electronic Lexical Database* (Cambridge, MA: The MIT Press, 1998), 199–216.

Scott, M., *WordSmith Tools,* Version 4 (Oxford: Oxford University Press, 2004).