The advantage of using relational databases for large corpora

Speed, advanced queries, and unlimited annotation

Mark Davies
Brigham Young University

Relational databases can be used to create large corpora that provide both very good search performance and a wide range of queries. This paper outlines how this approach has been used to create the *Corpus del Español*, which contains 100 million words of text in Spanish texts from the 1200s-1900s. The main databases are composed of n-grams tables (all unique 1, 2, 3, and 4 word sequences) and the associated frequency of all n-grams in each century (historical Spanish) and register (Modern Spanish). These tables are then joined to other tables containing part of speech, lemma, synonyms, and user-defined lists of words and lemma. There is essentially no limit to the amount of annotation that can be added in additional tables (with little or no impact on performance), and the SQL-based queries allow a wide range of searches that are not available with traditional corpora.

Keywords: relational databases, SQL, n-grams, Spanish, historical

1. Introduction

1.1 Competing goals

One of the fundamental challenges in the creation of search engines for large corpora is the need to balance size, annotation, and speed. Some approaches may achieve two of the three goals quite well, but shortcomings with the third factor will severely diminish the usefulness of the corpus. For example, an approach that places several levels of annotation within the actual text of a 100

International Journal of Corpus Linguistics 10:3 (2005), 307–334.
ISSN 1384–6655 / E-ISSN 1569–9811 © John Benjamins Publishing Company

million word corpus and yet does not use adequate indexes would probably be much too slow for most queries, as the search algorithm has to traverse the entire corpus for each query. Likewise, an "off the shelf" product like Microsoft Search will result in very fast queries (one or two seconds to search a 100 million word corpus), but is limited in that it mainly allows queries for just exact words and phrases, with little or no possibility of traditional annotation like lemma and parts-of-speech. Finally, some approaches use extremely elaborate annotation and are effective with a small one million word corpus, but are not scalable and would be completely inadequate for a 100 million word corpus.

1.2 Competing approaches

Let us briefly consider how traditional corpora have attempted to achieve all three of these goals – size, speed, and extensive annotation. Most large corpora use a "horizontal" approach, in which the textual database is composed of sequential words, much as one would find in a printed book. The following, for example, is a short extract from the British National Corpus (BNC):

(1) where all objects (ie including maps, drawings and texts) would be filed electronically (BNC: B2M)

This is the format used in the BNC (see Aston & Burnard 1998; Burnage & Dunlop 1993; Burnard 2000; Garside 1995; Leech, Garside & Bryant 1994), as well as other large 100+ million word corpora, such as ARTFL (Olsen & Hinkelman 1991), and CREA and CORDE from the Real Academia Española (http://www.rae.es). Often the raw text will be supplemented with XML, SGML, or similar markup, as with the following example from the BNC:

(2) <w CJS>where <w DT0>all <w NN2>objects <c PUL>(<w AV0>ie <w PRP>including <w NN2>maps<c PUN>, <w NN2>drawings <w CJC>and <w NN2>texts<c PUR>) <w VM0>would <w VBI>be <w VVN>filed <w AV0>electronically<c PUN>.

Obviously, it would be prohibitively costly in terms of the searching algorithm to traverse the 100+ million word text for each query. Therefore, these corpora make extensive use of separate indexes, which contain pointers to words in the actual textual corpus. For example, the BNC uses more than 22,000 index files (more than 2.0 GB) to speed up the searches.

Nevertheless, these "horizontal" or "linear" schemes still suffer from significant shortcomings. Even with extensive indexes, many queries are still quite slow. For example, with the current version of the BNC, a query to find the most common collocates with a moderate common word like [way] is quite expensive, and is almost prohibitive with a word like [with] or [had]. More seriously, due to the limitations of the indexing schema in the current version of the BNC, it is difficult (and sometimes impossible) to directly query part of speech, such as [had always VVD] or [other AJ0 NN1]. Finally, it is difficult to add additional layers of annotation – such as synonyms or user-defined lexical categories – which would allow users to perform more semantically-oriented queries.

"Corpus query processors" like the IMS Corpus Workbench (CWB) use a very different approach (see Christ 1994; Christ & Schulze 1995; as well as the IMS CWB site at www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/). In this approach, the fundamental units of the corpus (typically words) are stored in columns in sequential rows of a database. Additional annotation can then be added for each of these units, whether it be part of speech, lemma, etc. In most realizations of this approach – as with the IMS CWB – corpus creators typically work through the interface provided by the software. It is the "middleware" (IMS CWB) that actually creates the underlying database and which creates the SQL queries to extract data from the database tables. For corpus creators who use the Unix/Linux OS[1] and who do not need direct access to the database itself, this may provide a nice solution. Those who wish to have complete control over the architecture of the databases and the SQL queries, however, may prefer to create and deploy the databases themselves. This is the approach discussed in this paper.

1.3 A new approach

In this paper, then, we will consider an alternate architecture for large corpora, which is based on the use of relational databases that contain large tables of n-grams and frequency data. As we will see, such an approach allows us to achieve the three goals that we have outlined. First, there is essentially no limitation on the size of the corpora; we have successfully used the approach with both 100 million and 180 million word corpora. Second, the queries are very fast – typically just one or two seconds at the most to query a 100+ million word corpus. Third, the corpora can have any number of levels of annotation. The central n-grams / frequency tables are linked to other tables in the database, such as part of speech, lemma, synonyms, etymologies, translations to other languages, or customized categories that are defined by the end user. Most importantly, because each table has its own clustered index (discussed in more detail in Section 3.5), there is no decrease in performance as new annotation tables are added to the corpus.

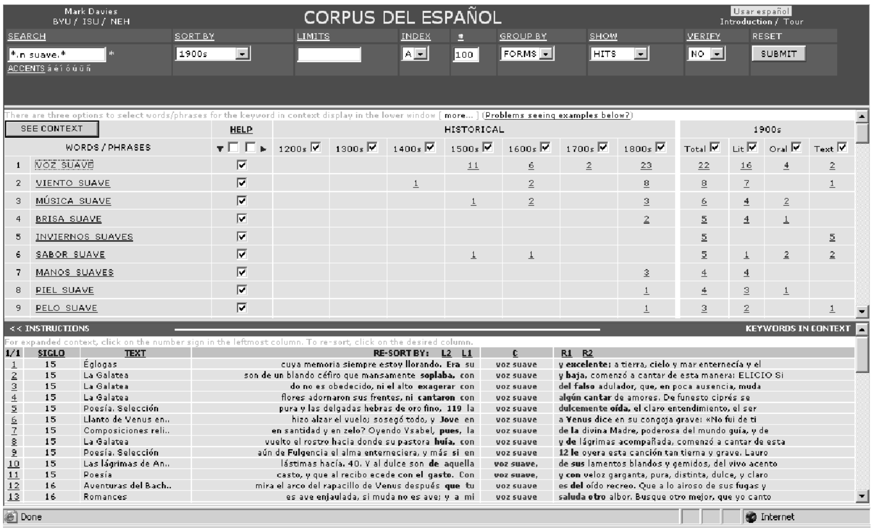
In Section 2 we will very briefly outline an existing corpus that uses the type of relational database architecture that we have proposed. Section 3 considers different ways in which the central n-grams / frequency databases can be constructed, and the advantages and disadvantages of each approach. Section 4 discusses how the central databases can be joined together with an essentially unlimited number of annotation schemas. Finally, Section 5 provides actual examples of some of the more complex queries that can be run against the corpus.

2. The Corpus del Español

2.1 Overview

To provide a concrete example of the relational database architecture, we will consider the 100 million word *Corpus del Español* (www.corpusdelespanol.org), which we have recently created. This corpus was funded by the National Endowment for the Humanities, and has recently been made freely available on the Web. The corpus contains 100 million words of text, composed of 20 million words from the 1200s–1400s, 40 million words from the 1500s–1700s, and 40 million words from the 1800s–1900s. The corpus allows a very wide range of queries, involving substrings, part of speech tags, lemma searches, queries involving synonym information, and searches that incorporate customized lists that were created by the end users.

In the discussion that follows, we should keep in mind what we are discussing when we use the term "corpus". The focus is not on the actual 100 million words of sequential text, i.e. the thousands of text files containing strings like *el hombre fue a la tienda para comprarse cerveza* "the man went to the store to buy himself some beer".[2] Rather, the focus of our discussion are the relational databases comprised of tables that contain n-gram and annotation information for the 100 million words of linear/sequential text. When we use the term "corpus" in this paper, then, we are usually referring to the complete linguistic resource that is made available to the end user. This is a combination of three things: the original textual corpus itself, the n-gram and annotation



Mark Davies
BYU / ISU / NEH

CORPUS DEL ESPAÑOL

Usar español
Introduction / Tour

SEARCH: *.n suave.* *
ACCENTS á é í ó ú ü ñ
SORT BY: 1900s
LIMITS:
INDEX: A
#: 100
GROUP BY: FORMS
SHOW: HITS
VERIFY: NO
RESET
SUBMIT

There are three options to select words/phrases for the keyword in context display in the lower window [more...] (Problems seeing examples below?)

SEE CONTEXT		HELP	HISTORICAL							1900s			
	WORDS / PHRASES	▼ □ □ ▶	1200s ☑	1300s ☑	1400s ☑	1500s ☑	1600s ☑	1700s ☑	1800s ☑	Total ☑	Lit ☑	Oral ☑	Text ☑
1	VOZ SUAVE	☑				11	6	2	23	22	16	4	2
2	VIENTO SUAVE	☑			1		2		8	8	7		1
3	MÚSICA SUAVE	☑				1	2		3	6	4	2	
4	BRISA SUAVE	☑							2	5	4	1	
5	INVIERNOS SUAVES	☑								5			5
6	SABOR SUAVE	☑				1	1			5	1	2	2
7	MANOS SUAVES	☑							3	4	4		
8	PIEL SUAVE	☑							1	4	3	1	
9	PELO SUAVE	☑							1	3	2		1

<< INSTRUCTIONS KEYWORDS IN CONTEXT

For expanded context, click on the number sign in the leftmost column. To re-sort, click on the desired column.

1/1	SIGLO	TEXT	RE-SORT BY: L2 L1	C	R1 R2
1	15	Églogas	cuya memoria siempre estoy llorando. Era su	voz suave	y excelente: a tierra, cielo y mar enternecía y el
2	15	La Galatea	son de un blando céfiro que mansamente soplaba, con	voz suave	y baja, comenzó a cantar de esta manera: ELICIO Si
3	15	La Galatea	do no es obedecido, ni el alto exagerar con	voz suave	del falso adulador, que, en poca ausencia, muda
4	15	La Galatea	flores adornaron sus frentes, ni cantaron con	voz suave	algún cantar de amores. De funesto ciprés se
5	15	Poesía. Selección	pura y las delgadas hebras de oro fino, 119 la	voz suave	dulcemente oída, el claro entendimiento, el ser
6	15	Llanto de Venus en..	hizo alzar el vuelo; sosegó todo, y Jove en	voz suave	a Venus dice en su congoja grave: «No fui de ti
7	15	Composiciones reli..	en santidad y en zelo? Oyendo Ysabel, pues, la	voz suave	de la divina Madre, poderosa del mundo guía, y de
8	15	La Galatea	vuelto el rostro hacia donde su pastora huía, con	voz suave	y de lágrimas acompañada, comenzó a cantar de esta
9	15	Poesía. Selección	aún de Fulgencia el alma enterneciera, y más si en	voz suave	12 le oyera esta canción tan tierna y grave. Lauro
10	15	Las lágrimas de An..	lástimas hacía. 40. Y al dulce son de aquella	voz suave,	de sus lamentos blandos y gemidos, del vivo acento
11	15	Poesía	casto, y que al recibo ecede con el gasto. Con	voz suave,	y con veloz garganta, pura, distinta, dulce, y claro
12	16	Aventuras del Bach..	mira el arco del rapacillo de Venus después que tu	voz suave	es del oído recreo. Que a lo airoso de sus fugas y
13	16	Romances	es ave enjaulada, si muda no es ave; y a mi	voz suave	saluda otro albor. Busque otro mejor, que yo canto

Done
Internet

Figure 1. Web-based interface

tables (the focus of this paper), and the user interface that provides access to these two resources.

The following figure provides an overview of the query interface for the *Corpus del Español*, and the actual search engine will be discussed in much more detail later in the paper.

In order to understand how the interface works, let us consider a concrete example. Suppose that a user wants to know which nouns are most common with the adjective *suave* "soft", and how these collocations have changed over time. The user would enter the following into the web-based form:

(3) [SEARCH] *.n suave.*

A web-based script would transform this into an SQL query (to be discussed below), which would be run against the relational database,[3] and the user would then see a table with all of the matching strings.[4] As the following table indicates, the user will also see the frequency of each string in each of the centuries from the 1200s to the 1900s, as well as three different registers of Modern Spanish (spoken, written-fiction, and written-non-fiction). For the query given above, the output would look similar to the following, with the entries *voz suave* "soft voice" (22 occurrences in the 1900s), *viento suave* "slight breeze" (8 occurrences), and *inviernos suaves* "mild winters" (5 occurrences):

Table 1. N-grams / frequency – web page results

#	PHRASE(S)	12	13	14	15	16	17	18	19	Lit	Oral	Misc
1	*voz suave*				11	6	12	23	22	16	4	2
2	*viento suave*			1		2		8	8	7		1
6	*inviernos suaves*								5			5
7	*manos suaves*							3	4	4		
	...											

Table 2. KWIC display

1/1	CENTURY	TEXT	RE-SORT BY: L-2 L-1	C	_R-1 R-2
11	15	La Galatea	rostro hacia donde su pastora huía, con	voz suave	y de lágrimas acompañada, comenzó a cantar de
18	17	Eusebio	su atención, mucho más cuando oyó una	voz suave	que acompañaba al dulce sonido, y que decía:
31	18	El lobo	media, agüelito? -pregunta con su	voz suave	y melodiosa. El bandido yergue la cabeza. En sus
47	19LIT	Vivir el Final	que a tí te gusten, y te las leeré, sí, con	voz suave	y cariñosa, como esa muchacha llorosa le lee a su
	...	...	...	...	...

Many users will only want to see a list of the matching words and phrases, with their frequency in each period. However, there are also many different options to select certain words and phrases in order to see them in KWIC format. Users can click on a phrase to see it in all historical periods, or they can click on the frequency listing in one of the columns to see that particular word or phrase in that historical period (for example, 12 cases of *voz suave* in the 1700s). Alternatively, they can select multiple words and phrases and multiple historical periods (or registers of Modern Spanish) and see the KWIC display for the multiple selections.

Once the specific phrase(s) and historical period(s) are selected, the user can see these words and phrases in a traditional KWIC display. In this view, users can then re-sort the occurrences by left or right context words, and can also select certain phrases to see in more detail. A sample of the KWIC display is seen in the Table 2 (note the entries have been shortened here in the interest of space).

2.2 Wide range of queries

As was discussed in the introduction, there are at least three competing goals with corpora – size, annotation, and speed. As we have seen, the *Corpus del Español* achieves the goal of robust size, since it includes 100 million words of text, from the 1200s–1900s. The *Corpus del Español* is also very robust in terms

Table 3. Range of queries in the Corpus del Español

Type of search	**Query**	**Result**	**Explanation**
Pattern matching	*cans*	*cansado, descansar, incansable*	words related to the root for "tired"
	*udo +1500s –1800s –1900s	*cabezudo, vedijudo, capilludo*	words ending with the (sometimes pejorative) suffix [-udo], which appear in the 1500s, but not the 1800s or 1900s
Collocations	tan * como	*bueno, lento, fácil*	"as ADJ as" (*good, show, easy*)
	.n duro. +1900s –1500s –1600s	*línea, disco, cabeza,*	"hard N" that appear in the 1900s, but not the 1500s or 1600s (*line, drive, head*)
Lemma	decir.*	*dice, dijeron, diremos*	forms of "to say"
	haber.* 1500s>5 –1900s	*avia, uvo, obiese*	forms of *to have* (+PP) occurring at least five times in the 1500s, but not occurring in the 1900s
Part of speech	*.adv	*pronto, bien, rápidamente*	adverbs (*soon, well, quickly*)
	*.v_inf +1900s –1800s	*detectar, frenar, intercambiar*	infinitives occurring in the 1900s, but not in the 1800s (i.e. new verbs) (*detect, brake, exchange*)
Synonyms / antonyms	!inteligente	*vivo, capaz, agudo ...*	synonyms of "intelligent"
	!hablar.* +1900s –1700s +lemma	*comentar, dialogar, platicar*	Synonyms of *hablar* (*to speak*) that occur in the 1900s but not in the 1700s, grouped by lemma (*to comment, have a dialog with, chat*)
Customized, user-defined lists	estar.* tan [gonzález: emociones].*	*estaba tan alegre* *estoy tan deprimido*	any form of *estar* "to be" + "as" + any form of any word in the [emociones] list created by [gonzález]
Combinations of the preceding	!mandar.* que *.v_subj_ra +1900s –1800s –1700s	*hizo que dijeran* *mandé que volviera*	Any form of any synonym of "to command" + past subjunctive, which occurs in the 1900s, but not in the 1700s or 1800s (*he made them say, I ordered him to come back*)

of the second goal – the levels of annotation. As the following table shows, the *Corpus del Español* offers a very wide range of searches. Note that for every search, it is possible to use the query syntax to limit the results to those that occur at a given frequency in different historical periods and in different registers, which is a result of the underlying n-grams architecture.

The *Corpus del Español* also achieves the third goal, which is speed. In spite of its size and levels of annotation, the corpus is quite fast. For example, none of the queries in the preceding table takes more than two seconds to return the full range of results from the 100 million word corpus. In addition to the three main goals described above, the use of relational databases as the foundation of the corpus architecture has a fourth advantage. Because of the modular architecture for annotation tables, we can add on any additional levels of annotation – etymologies, bilingual dictionaries, or any other resource – and virtually all queries will still take less than two or three seconds.

3. Creating the database of n-grams and frequency

3.1 The n-grams databases

In this section we will consider in some detail the architecture underlying the *Corpus del Español*, and how this relates to size, speed, and levels of annotation. As we will see, the key to robust performance are the n-grams and frequency databases that are at the heart of the corpus, and the links that exist between these databases and the annotation databases – parts of speech, lemma, synonyms, and user-defined tables. Let us first consider, then, the architecture of the central n-grams and frequency databases. As noted, we will again use the *Corpus del Español* as the point of departure for our discussion.

The first step in building the *Corpus del Español* was to create a listing of all of the 1, 2, 3, and 4 word clusters (n-grams) in the entire corpus. In order to create these n-grams databases, we used the WordList function of WordSmith to create a list of all of the unique n-grams – with their associated frequency – for each of the historical periods as well as the different registers of Modern Spanish. These were then imported into the MS SQL Server database, where they were merged together to create the 1, 2, 3, and 4-gram tables. An example of one row from the forty million rows in the 3-grams table is the following:

Table 4. N-grams/frequency table in SQL Server

w1	w2	w3	x12	x13	x14	x15	x16	x17	x18	x19	19-Lit	19-Oral	19-Misc
hora	*en*	*que*	31	0	8	61	55	39	369	92	78	10	4

The columns w1, w2, w3 refer to each of the "slots" in the 3-gram; the columns x12–x19 refer to the frequency of this 3-gram in the 1200s–1900s; and 19-Lit, 19-Oral, and 19-Misc refer to the frequency in these three registers from the 1900s. As might be imagined, the tables are rather large. There are nearly one million distinct 1-grams (i.e types), eleven million distinct 2-grams, forty million distinct 3-grams, and 65 million distinct 4-grams. Yet because each of these relational database tables is indexed, including some clustered indices (to be discussed in more detail later on), the queries on the tables are very fast – usually just one or two seconds.

3.2 Basic "slot-based" queries

Even without annotation, there are a number of useful queries that can be done directly on the n-gram/frequency databases. For example, the user would enter the following into the web-based form, which would generate the following SQL query:

(4) [SEARCH] tan * como
```
select * from x3 where
w1 in ('tan') and
w3 in ('como')
order by x19 desc
```

This query will search the [3-grams] table for all cases where the [w1] column is *tan* "as/so" and the [w3] column is *como* "as", and order the results by the frequency in the [x19] column (the default). This will produce hits like *tan bueno como* "as good as", *tan rápido como* "as fast as", etc.

A second example shows how users can input information into the [SORT] and [LIMITS] fields, to access the frequency information for the different historical periods:

(5) [SEARCH] *aren [SORT] 1200s [LIMITS] +1200s +1400s –1900s
```
select * from x1 where
w1 like ('%aren') and
x12>0 AND x14>0 AND x19= 0
order by x12 desc
```

This will search the [1-grams] table for all cases where the word [w1] ends in [-aren] (a marker for the archaic 3PL future subjunctive), which occur at least once in the 1200s and the 1400s, but which have disappeared by the 1900s, and then sort the results by the frequency in the 1200s. This will result in cases like *fallaren* "that they might lack" and *tomaren* "that they might take".

The final query is somewhat more complex, and shows the use of wildcard and OR operators:

(6) [SEARCH] me/te/nos/le/les quier* *r [SORT] 1800s [LIMITS] +1800s

```
select * from where
w1 in ('me', 'te', 'nos', 'le', 'les') and
w2 like ('quier%') and
w3 like ('%r') and
x18 > 0
order by x18 desc
```

This query will search the [3-grams] table for all records where the first word [w1] is one of the following [*me,te,nos,le,les*] (indirect object pronouns), the second word [w2] has the pattern [*quier-*] (some forms of the verb *querer* "to want"), the third word [w3] ends in [-*r*] (possibly an infinitive), and the string occurs at least once in the 1800s. This will produce strings like *le quiero decir* "I want to tell him" or *nos quiere llamar* "wants-3SG to call us".

Because Spanish has morphology that is both strong and fairly regular, users can employ simple lists of words and word patterns to search for even relatively complex syntactic constructions, as in the example just given. However, at some point it will obviously be necessary to have more complete annotation, including annotation for those lemmata that are not morphologically regular (e.g. *quis** for preterite forms of *querer*), as well as parts of speech that are not predictable in terms of forms (such as nouns and adjectives in Spanish). In the following three sections, we discuss how this annotation can be carried out and the different ways in which it can be stored in the relational databases.

3.3 Traditional part of speech annotation and lemmatization

There are at least three possible approaches to the question of where and how to place part of speech (POS) and lemma information in the relational database. The first approach would probably be the most intuitively familiar to those who are used to working with a morphologically weak language like English, where POS disambiguation is a real challenge. In this approach, we want to have as much contextual information as possible as the text is being tagged. Therefore, before we create the n-grams frequency lists and import them into the database, we tag the corpus in a traditional manner – word by word, with limited "look backward" and "look forward" capabilities. For example, the tagger would convert the English phrase "needs to talk" into the following string (here we give the word form + lemma + POS):

(7) needs_need_vbz to_TO0_to talk_talk_vvi

Once the corpus has been tagged, a script would then process the text to separate the actual word forms from the POS tag and the lemma, insert field delimiters, and place each word on a separate line, as in the following:

(8)	needs	VBZ	need
	to	TO0	to
	talk	VVI	talk

The text file would then be imported into the database, which would have rows like the following:

Table 5. 1-grams table

ID	w1	pos1	lem1	w2	pos2	lem2	w3	pos3	lem3
342	*needs*	VVZ	need						
343	*to*	TO0	to						
344	*talk*	VVI	talk						

We would then run an SQL query, which inputs into the empty fields in each n-grams table the following [n-1] words (for example, word2 and word3 slot in a 3-gram table). This could be done with a simple SQL statement like the following

(9)
```
update a set
a.w2 = b.w1, a.w3 = c.w1,
a.pos2 = b.pos1, a.pos3 = c.pos1,
a.lem2 = b.lem1, a.lem3 = c.lem1
from x3 as a, x3 as b, x3 as c
where a.ID+1 = b.ID and a.ID+2 = c.ID
```

The result would be a table like the following, containing all of the sequential 3-grams in the entire corpus:

Table 6. 3-grams table

ID	w1	pos1	lem1	w2	pos2	lem2	w3	pos3	lem3
342	*needs*	VVZ	need	*to*	TO0	to	*talk*	VVI	talk
343	*to*	TO0	to	*talk*	VVI	talk	*with*	PRP	with
344	*talk*	VVI	talk	*with*	PRP	with	*Bill*	NP0	Bill

Finally, we would use the SQL "GROUP BY" statement to create a separate table with all of the unique 3-grams and their associated frequency in different registers or historical periods (e.g. freq$_x$ and freq$_y$ in the following table):

Table 7. 3-grams table, sorted by lemma, with frequency

ID	w1	pos1	lem1	w2	pos2	lem2	w3	pos3	lem3	freq$_x$	freq$_y$
26	*needs*	VVZ	need	*to*	TO0	to	*talk*	VVI	talk	12	8
27	*needing*	VVG	need	*to*	TO0	to	*send*	VVI	send	6	5
28	*needed*	VVD	need	*to*	TO0	to	*put*	VVI	put	8	5

As with the *Corpus del Español*, this table could then be queried to see which verbs, for example, occur most frequently after a form of [need] followed by [to]. The following query, then, would produce the SQL that would retrieve all three rows from the preceding table:

(10) [SEARCH] need.* to *.VVI

```
select * from x3 where
lem1 in ('need') and
w2 in ('to') and
pos3 in ('VVI')
order by freqx desc
```

With this annotation schema, there is absolutely nothing lost in going from the output of the tagger (Table 7 above) to the relational databases (Tables 5–6). Therefore, by employing a "vertical" corpus as in Tables 5 and 6 – as opposed to a traditional "horizontal" corpus – we maintain all of the advantages of the traditional architecture. Yet, as we will see, we will also gain a number of important advantages that can only be achieved by using a relational database – such as the very fast queries that result from the optimized indexes on each column in the database.

3.4 Annotation and disambiguation via the relational database

The annotation scheme just described works well when the corpus creators themselves are tagging the corpus. It also works well for a previously-tagged corpus. For example, we have recently used this approach to convert the BNC into relational database form (see http://view.byu.edu), using all of the part of speech information that was produced with the CLAWS4 tagger (see Leech et al. 1994).

With the *Corpus del Español*, however, we have used a rather different annotation scheme. As discussed in Section 3.1 above, we used the WordList function of WordSmith to create lists of all of the distinct 1, 2, 3, and 4-word sequences in the 100 million words of text, along with the frequency in each century and register. This information was then imported into the SQL Server database, where the tables were merged together to provide the frequency for each unique n-gram in each historical period. The following is an example of one row from the 3-gram table, for the phrase *no puede ser* "it cannot be". Notice, however, that there is still no POS or lemma information for each of the word forms. This annotation information was in separate tables, as in the two tables following the n-grams table (pos = part of speech):

Table 8. Separate ngrams/frequency and annotation tables

w1	lem1	pos1	w2	lem2	pos2	w3	lem3	pos3	x12	x13	x14	…	x19	19-Lit	19-Oral	19-Misc
no			*puede*			*ser*			105	13	140	…	646	200	339	107

x3

w1	pos
no	adv
puede	v_pres
ser	v_inf
ser	n

pos

w1	lem
no	no
puede	poder
ser	ser

lemma

A SQL "join" operation was then performed to import the POS and lemma information into the main n-grams tables, e.g.

(11) update x3

```
set x3.lem1 = lemma.lem
from x3, lemma
where x3.w1 = lemma.w1
```

This process would be repeated for both the POS and lemma fields of all three "slots" in the 3-grams table.

There is one obvious drawback to this schema. For words that have more than one possible part of speech or lemma, there are two possible entries from

Table 9. Contextual disambiguation, based on n-grams

w1	lem1	pos1	w2	lem2	pos2	w3	lem3	pos3	x12	x13	x14	…	x19	19-lit	19-oral	19-misc
no	no	adv	*puede*	poder	v_pres	*ser*			105	13	140	…	646	200	339	107
de	de	prep	*un*	un	a_ind	*ser*			0	0	2	…	37	6	17	14

the POS or lemma tables that could be input into the n-grams table. For example, in the following table, *ser* can either be a noun (*el ser humano* "the human being") or an infinitive (*de ser humilde* "to be humble"). Which of the two entries from the POS table will be input into the row in the n-grams table?

Fortunately, using SQL updates we can look at the contextual information in the n-grams table itself to insert or modify the POS and lemma values, based on the values in the other word slots. In the case of *ser* "to be" just given, the UPDATE query will assign a value of [v_inf] when [*ser*] is preceded by a modal verb such as *poder* (the first row), and it will assign the value of [n] to [*ser*] when it occurs after an indefinite article (as in the second row). The first UPDATE query is shown in the following example:

(12)

```
update x3
set pos3 = ‘v_inf’
where lem2 = ‘poder’ and w3 = ‘ser’
```

The important point is that with this annotation schema, the tagging takes place not within the textual corpus itself – as with the traditional approach – but rather within the n-grams tables of the relational database.

We should note that the end result of using the database approach to POS tagging and the traditional approach will likely produce similar results. In the traditional "linear" approach, of course, the tagger employs a "look left / look right" algorithm to use contextual words for disambiguation. Our database approach does essentially the same thing. Assuming a 7-gram database, as we attempt to disambiguate words in middle column (call it *w4*), we have access to the words (and their associated POS tags) for the three words to the left (*w1,w2,w3*) and the three words to the right (*w5,w6,w7*). To the extent that the context required for disambiguation is no more than seven words, this approach works fine. In addition, it has the advantage that it is quite fast – updates on tens of millions of words of text can usually be accomplished in one or two seconds. We recently used this approach to carry out the POS tagging and lemmatization for a 20 million word corpus of Modern Spanish. Over a period of 6–8 months, repeated manual checking of the POS tagging carried out within the database itself was at least as accurate as that done with traditional "linear" tagging. (See https://www.fastlane.nsf.gov/servlet/

showaward?award=0214438 for more information on this Modern Spanish project, which has been funded by the US National Science Foundation, and which is being carried out with Douglas Biber.) Nevertheless, for those who prefer to tag texts in the traditional way (within the linear text itself) and then import the tagged text into the database, this option is always available, as discussed in Section 3.3 above.

Researchers who are used to working with a language such as English – which has relatively poor morphology – might object to the idea of having the POS and lemma information in tables where they are removed from their original context – which in our case is the unannotated textual corpus that is stored in 2000 word chunks in the separate "linear / textual" database (see Note 2). The concern might be that there would not be enough information in the n-grams database to successfully disambiguate cases of polysemy. For a language such as Spanish, however, this concern is probably unfounded. Because Spanish is such a morphologically strong language, there are relatively few forms (as with *ser*) that have a high frequency as two different parts of speech or two different lemmata, and which cannot be successfully disambiguated with even a very limited context.

In addition, for 2-gram, 3-gram, and 4-gram searches, the query that is submitted by the end user will often serve to limit the results to just those rows of the data where the words have the desired part of speech or lemma. For example, suppose that the user is searching for cases of a form of *no* + form of *poder* + infinitive (e.g. *no puede ser* "can-not-3SG be" or *no podemos decir* "we cannot say"). S/he would enter the following in the web-based form (note that the period+asterisk after a word signifies lemma, and an asterisk+period before a word signifies part-of-speech):

(13) no poder.* *.v_inf

This search phrase is then translated into the following SQL command, which is run against the x3 (3-grams) table.

(14) select top 300 * from x3 where
w1 = 'no' and
lem2 = 'poder' and
pos3 = 'v_inf'

Although the first row is marked as [v_inf] (and thus matches the query), this is actually redundant. Once we have specified that [lem2] = [poder] ("to be able"), virtually any word in the [w3] slot, which has two possible POS tags,

Table 10. Using contextual disambiguation in query results

id	w1	lem1	pos1	w2	lem2	pos2	w3	lem3	pos3	x12	x13	x14	...	x19	19-lit	19-oral	19-misc
343	*no*	no	adv	*puede*	poder	v_pres	*ser*	ser	v_inf	105	13	140	...	646	200	339	107
344	*de*	de	prep	*un*	un	a_ind	*ser*	ser	n	0	0	2	...	37	6	17	14

will be an infinitive [v_inf]. Therefore the query itself – at least in the case of the 2, 3, and 4-gram tables – often disambiguates things sufficiently to select only the correct rows from the database.

3.5 Annotation – redundancy vs ambiguity

While we initially used the annotation schema described in Section 3.4 for the *Corpus del Español,* we eventually settled on an even more unconventional approach to POS and lemma annotation. As we have seen, with a language like Spanish – where POS ambiguity is much less of a problem than in English – we can probably do the POS tagging and lemmatization right within the relational database itself. Soon it became apparent that it might be possible to remove the annotation even further from the original context. In this approach, the annotation tables (POS and lemma) that are used for the actual queries are separate from the n-grams tables themselves. The following tables show this architecture, in which the n-grams/frequency information is in one table, and the POS and lemma information are in two additional tables. The two tables are never merged together, as they are in Table 10 above.

In this scenario, there would be simple SQL JOIN commands to link the two databases. The same query shown above would be translated into the following SQL command. In this case, the database first sub-queries the POS (x_c) and lemma (x_L) tables, and then feeds the output from these tables into a query of the main n-gram/frequency table (x3):

```
(15) select top 300 * from x3 where
     w1 in (‘no’) and
     w2 in (select w1 from x_L where x1 in (‘poder’)) and
     w3 in (select w1 from x_c where x1 in (‘v_inf’))
```

Because of the very low level of polysemy in Spanish, and because the immediate context can usually be used successfully for disambiguation (as discussed above), there are relatively few cases where the non-contextual annotation presents a problem. For example, if a user searches for [no poder.* *.v_inf], then [*no puede ser*] and other such strings will be retrieved, because *ser* is listed as a [v_inf] in the POS table. And in fact all of these examples of the potentially

Table 11. Separate n-grams/frequency, POS, and lemma tables

w1	w2	w3	x12	x13	x14	x15	x16	x17	x18	x19	19-lit	19-oral	19-misc
no	*puede*	*ser*	105	13	140	518	423	269	892	646	200	339	107

x3

w1	x1
no	adv
puede	v_pres
ser	v_inf
ser	n

pos
(x_c)

w1	x1
no	no
puede	poder
ser	ser

lemma
(x_L)

polysemous *ser* really are in its use as an infinitive – there would be only a very small percentage of cases where *ser* is the noun "(human) being". Likewise, if a user searches for [el *.n humano], then [*el ser humano*] "the human being" will be one of the matching strings that is retrieved, and again virtually all of these cases will be with *ser* as a noun (because of the preceding determiner and the following adjective).

The only problem in the assignment of part-of-speech and lemma will occur in those relatively few cases in which 1) a word is polysemous and 2) both meanings are highly frequent and 3) the search context is not sufficiently rich to disambiguate the multiple meanings. Our experience from more than a year's worth of working with the *Corpus del Español* shows that there are in fact very few cases where serious ambiguity arises, and even in these cases the relatively infrequent spurious phrases can simply be ignored by the end user.

One of the major advantages of placing the annotation in tables that are separate from the context to which they refer deals with redundancy. In the database architecture in which the POS and lemma information is part of the n-grams tables, then redundant annotation occurs every time a word occurs in any slot of any n-gram table. Thus a word like *es* "is" would be annotated as [POS=v_pres; lemma = ser] in each of the 209,160 rows of the 3-grams table where it appears in the [w2] slot – as well as hundreds of thousands of other rows for the other slots of the 3-grams table and the other n-grams tables. By placing the annotation in a separate table, there is exactly one entry for [*es*]. There are secondary benefits from placing the annotation in separate tables, both of which are related to the issue of redundancy. First, because the annotation only occurs in one or two rows of the POS or lemma tables, hundreds or thousands can be updated in a matter of one or two seconds. If a form can occur in hundreds of thousands of rows, on the other hand, then updating the annotation for the large amount of forms becomes more difficult.

A second issue is somewhat more technical in nature, and deals with the physical architecture of database tables. In most databases, only one column in each table can have a "clustered" index, which means that the rows in the table are physically arranged on the hard drive according to the contents of that column. (In a non-clustered index, on the other hand, there are pointers to the data, but the data itself may be spread over the entire hard drive.) For our case, the important point is that if the clustered index is placed on the [w1] column (the first word in the n-gram), then any indices on the annotation columns in the n-grams table cannot be clustered, and queries dealing with POS or lemma will be relatively slow (taking perhaps 15–20 seconds for a 100 million word corpus). By placing POS and lemma annotation in their own tables, each of these tables can contain a clustered index, and text retrieval will be much faster – typically just a second or two for even the most complex queries.

4. Modularity

4.1 Basic POS, lemma, and frequency-based queries

Now that we have discussed the annotation architecture, let us briefly consider some concrete examples of how this information can be used to create some rather powerful queries. In this first section we will briefly outline the interaction of the n-grams, frequency, part of speech, and lemma databases, by providing examples of three searches that involve an increasing degree of complexity. In Section 4.2, we will expand the scope to include other databases such as thesauruses and user-defined tables. The general point of this section is that with a relational database architecture, one can add as many levels of annotation as one wants. Although each new level creates powerful interaction with the other types of annotation, the speed of the search engine is in no way degraded with additional annotation.

To begin with a relatively simple example, suppose that the user wants to find cases of "tough-movement" constructions like *difícil de creer* "hard to believe". The user would enter the following into the web-based form, and this would be converted by the script into the following SQL command:

(16) [SEARCH] difícil.* de *.v_inf

```
select * from x3 where
w1 in (select w1 from x_L where x1 in ('difícil')) and
```

```
w2 in (‘de’) and
w3 in (select w1 from x_c where x1 in (‘v_inf’))
order by x19 desc
```

The sub-queries select the lemma (line 2) and part-of-speech (line 4) information for [difícil] and [v_inf] from lemma (x_L) and part-of-speech (x_c) tables. It then uses this information to search the actual 3-grams table [x3], and orders the results by the frequency of these n-grams in the 1900s (field x19), producing results like *difícil de explicar* “hard-SG to explain” (23 occurrences in the 1900s), *difíciles de encontrar* “hard-PL to find” (12 occurrences), and *difícil de creer* “hard-SG to believe” (12 occurrences). Note that this query takes less than half a second to query the 100 million word corpus.

Let us now consider a somewhat more complex example. Recall that because the n-grams tables include information on the frequency of each of the n-grams in each of the sub-corpora, this frequency information can be used directly as part of the query syntax. For example, a user might want to know which adjectives are used with *situación* “situation” or *condición* “condition” at least two times in the 1900s, but which do not appear in the 1800s. The user would input the following, and this would be converted to the SQL command:

(17) [SEARCH] situación/condición *.adj [SORT] 1900s
[LIMIT] 1900s>1 –1800s

```
select * from x2 where
w1 in (‘situación’, ‘condición’) and
w2 in (select w1 from x_c where x1 in (‘adj’)) and
x19>1 AND x18= 0
order by x19 desc
```

This will produce results like *situación concreta* “concrete situation”, *situación incómoda* “uncomfortable situation”, and *condición existencial* “existential condition”.

Finally, it is possible to group the results by lemma. For example, users might want to know which verbs occur most commonly in the present subjunctive after the permissive verb *dejar* “to allow”. They would enter the following into the search form (note the GROUP BY option), and it will be converted to the SQL command:

(18) [SEARCH] dejar.* que *.v_subj_pres [GROUP BY] lemma

```
select * from x3 where
w1 in (select w1 from x_L where x1 in (‘dejar’)) and
```

w2 in ('que') and
w3 in (select w1 from x_c where x1 in ('v_subj_pres'))

Subsequent queries are then generated by the web-based script, to insert the lemma into the n-grams table. For example, the following query inserts the appropriate lemma into the lemma slot (lem3) for word3:

(19) update t set t.lem3 = lemma.x1 from temp_table as t, lemma where t.w3 = lemma.w1

Finally, these lemma fields are then used to group all of the words in a particular slot, and to sum the frequency of all of the matching lines:

(20) select l1 as w1,l2 as w2,l3 as w3, sum(x12) as x12, ..., sum(x22) as x22
from temp_table
group by l1,l2,l3 order by x19 DESC

The result of all of these SQL queries – which of course are done "behind the scenes", are results like [*dejar que hacer*] "to allow someone to make", [*dejar que caer*] "to let something fall" (i.e. "to drop") and [*dejar que hablar*] "to let someone speak". Even with the multiple SELECT, INSERT, and UPDATE queries, however (which are all hidden from the end user), the search still takes less than one second.

4.2 Semantically-based queries

In the previous section we discussed the way in which part of speech and lemma information can be integrated into the central n-grams / frequency databases, and how all of this information can be used to create rather powerful queries. As we have suggested, however, the relational database approach even allows us to go much further than this. The number of levels and types of additional annotation that we add are essentially unlimited. We can add additional resources like thesauruses, etymological information, translations to other languages, and user-defined databases, all without suffering any hit in terms of performance.

An example of one of these "auxiliary" databases is a table in the *Corpus del Español* that contains the synonyms for 30,000 words. Users can submit a simple query like either of the two that follow:

(21) !romper
!romper.*

A simple query like [!romper] "to break" will retrieve a listing of all of the synonyms for the word (such as *romper, derrotar, deshacer, derribar*, and seventeen other verbs), along with the frequency of each of these words in each of the sub-corpora. In the second case, [!romper.*], the frequency and distribution of all of the forms of all of the synonyms are retrieved, such as *romper, rompe, rompieron, deshizo, deshaga*, etc. These can also be re-grouped by lemma, to show the overall frequency of all of the forms for each of the synonyms. In addition, after seeing a list of synonyms, users can click on any one of the words in the new list, and see a new list of synonyms for that word. This allows them to create "semantic chains" of related words. In essence, this is like a traditional electronic thesaurus, with the important difference that users can see the frequency and distribution of each synonym.

The important point is that this database (and any number of other databases) can easily be linked to the main n-grams / frequency database, to create even more powerful queries. An example of this is the following query:

(22) [SEARCH] !decir.* !chiste.*
[SEARCH] !mandar.* que *.v_subj_ra

In terms of the underlying SQL command, the following refers to the first example [!decir.* !chiste.*]:

```
(23) select * from x2 where
     w1 in (select w1 from x_L where x1 in ('decir', 'mencionar', 'aseverar', 'exponer' ...)) and
     w2 in (select w1 from x_L where x1 in ('chiste', 'ocurrencia', 'chirigota', 'ingeniosidad' ... ))
     order by x19 desc
```

The first query searches for all forms of all synonyms of *decir* "to say" followed by all forms of any synonym of *chiste* "a joke", producing results like *conté chistes* "I told jokes" or *diciéndose burlas* "telling to each other jokes". The second query searches for all forms of all lemmata that are synonyms of *mandar* "to order, command", followed by the subjunctive marker *que* "that", followed by a past subjunctive, and this produces results like *mandé que fueran* "I made them leave" and *hicieron que dijera* "they made her say".

4.3 More semantically-based queries: User-defined lists

Another example of linked tables and unlimited annotation in the *Corpus del Español* are the customized lists of words that users can create, which contain words that are related morphologically, syntactically, or semantically, such as adjectives that describe emotions, words ending in [-azo] (which often denote a blow or strike), or a list of temporal adverbs. After they are created by the users, they are stored in tables where they can later be used as part of the query syntax.

For example, suppose that a user [jones] creates a list called [emotions], which contains a list of verbs of emotion (e.g. *gustar, alegrar, sorprender* "to please, make happy, surprise"). At the most basic level, Jones can simply retrieve this list and see the distribution and use of all of the words in the list. But s/he can also use this list as part of a more complex query, as in the following:

(24) me/nos/te/os/le/les [jones:emotions].* que *.v_subj_pres

This query searches for all cases of:

(25) one of the indirect object pronouns [*me, nos, te, os, le, les*] +
any form of any of the words in the customized [emotions] list created by [Jones] +
que +
present subjunctive

The script then translates this into the following SQL command, which is passed to the database:

(26)
```
select top 300 * from x4 where
w1 in (‘me’, ‘nos’, ‘te’, ‘os’, ‘le’, ‘les’) and
w2 in (select w1 from x_L where x1 in (
(select w1 from jones where x1 = ‘emotions’)
) and
w3 in (‘que’) and
w4 in (select w1 from x_c where x1 in (‘v_subj_pres’))
```

and will return a list like *me gusta que haya* "it pleases me that there is", *le sorprende que tengan* "it surprises him/her that they have".

4.4 Additional levels of annotation

While the n-grams approach that we have discussed here works well with POS, lemma, synonyms, and user-defined lists, some modifications to the underlying database architecture may be necessary to join these tables to other tables containing metadata (e.g. title, author, or text size) or text-critical information such as emendations or transcriptions. This is because the conflation of unique strings of text from different texts into the n-grams tables involves the loss of information for each of the individual occurrences of a particular n-gram. We have used a modified architecture for a relational database version of the British National Corpus (see http://view.byu.edu), which we have recently placed online. With this modified architecture, there is less conflation into n-grams, and we can maintain full annotation for each individual occurrence in the 100 million words of text.

Some types of annotation, however, would be quite difficult or perhaps impossible with our approach. While this approach does allow for wildcards (see Example 6 above), it does not allow for direct use of regular expressions, since these are not a native operator in MS SQL Server. One could devise a web-based script to process the input and convert the regular expression into multiple queries that would be recognized by SQL Server, but this might be overly cumbersome. Finally, hierarchical information – such as treebanks – would most likely not work well with the n-grams approach.

In summary, there can be an unlimited *number* of levels of annotation on the corpus, and most types of annotation can be incorporated – whether parts of speech, or lemma, or synonyms, or translations between languages, or etymologies, or customized lists, and these can all be linked together with simple SQL JOIN commands. It is not apparent how this degree of flexibility or power would be an inherent property of the standard scheme, in which the annotation is usually based within the textual corpus itself. It is also not clear to what degree this would be possible with the alternate CQP (corpus query processor) approach since – as far as we are aware – there are no large, publicly-available corpora based on CQP that possess the range of annotation types that we have incorporated into the *Corpus del Español*.

5. Advanced comparisons using sub-queries

Perhaps the best example of the power of the relational database approach is the way in which these databases allow extremely complex comparisons of competing structures. For example, suppose that a user wanted to know which nouns occur more frequently with the adjective *blanco* "white" than with *negro* "black". With our approach, it would be quite easy to check for this. The user would enter the following into the search form on the web, and this would be converted to the following SQL command:

(27) [SEARCH 1] *.n blanco.* [LIMITS] 1900s > 10 [GROUP BY] lemma
[SEARCH 2] *.n negro.* [LIMITS] 1900s < 5 [GROUP BY] lemma

```
select blanco.freq as blanco,negro.freq as negro,blanco.w1 as w1 from
(
select top 10000 sum(x2.x19) as freq,x_l.x1 as w1
from x2,x_l where
x2.w1 in (select w1 from x_c where x1 in (‘n’)) and
x2.w2 in (select w1 from x_L where x1 in (‘blanco’)) and
x2.w1 = x_l.w1
group by x_l.x1
order by sum(x2.x19) desc
) blanco
left join
(
select top 10000 sum(x2.x19) as freq,x_l.x1 as w1
from x2,x_l where
x2.w1 in (select w1 from x_c where x1 in (‘n’)) and
x2.w2 in (select w1 from x_L where x1 in (‘negro’)) and
x2.w1 = x_l.w1
group by x_l.x1
order by sum(x2.x19) desc
) negro
on blanco.w1 = negro.w1
where blanco.freq > 10 and negro.freq < 5
order by blanco.freq desc
```

This SQL query uses two sub-queries to retrieve the 10000 most frequent nouns with *blanco* and then with *negro*, and then limits the output to just those that occur more than ten times with *blanco* and less than five times with *negro*.

The query takes less than five seconds, and returns a list including *vino* "wine", *guante* "glove", and *diente* "tooth". The opposite query – nouns that occur more frequently with *negro* – takes another four seconds and returns a list containing *agujero* "hole", *humor* "(sense of) humor", and *terciopelo* "velvet".

Likewise, a user could quickly and easily check to see which verbs in Spanish appeared in the preterite much more often than they did in the imperfect. The user would enter the following into the search form on the web, and it would be converted to the following SQL query:

(28)

```
[ SEARCH 1] *.v_pret    [LIMITS] 1900s > 200   [GROUP BY] lemma
[ SEARCH 2] *.v_impf    [LIMITS] 1900s < 40    [GROUP BY] lemma
select pret.freq as pret,impf.freq as impf,pret.w1 as w1 from
(
select top 10000 sum(x1.x19) as freq,x_l.x1 as w1
from x_l,x_c,x1
where x_l.w1=x1.w1 and x_c.w1=x1.w1 and x_c.x1 = ‘v_impf’
group by x_l.x1
order by sum(x1.x19) desc
) impf
left join
(
select top 10000 sum(x1.x19) as freq,x_l.x1 as w1
from x_l,x_c,x1
where x_l.w1=x1.w1 and x_c.w1=x1.w1 and x_c.x1 = ‘v_pret’
group by x_l.x1
order by sum(x1.x19) desc
) pret
on impf.w1 = pret.w1
where pret.freq > 200 and impf.freq < 40
order by pret.freq desc
```

This SQL query uses two sub-queries to retrieve the 10000 most frequent verbs in the imperfect and in the preterite, and then limits the output to just those that occur more than 200 times in the preterite and less than 40 times with the imperfect. The query takes less than five seconds and returns a list that includes "preterite-oriented" verbs like *conquistar* "to conquer", *fallecer* "to die", and *aclarar* "to make clear". The opposite search – verbs that are more frequent in the imperfect – also takes less than five seconds and yields verbs like *soler* "to be in the habit of V-ing", *lucir* "to shine", and *carecer* "to lack".

The subqueries that we have considered here are just a small sampling of those that are available with a relational database like the *Corpus del Español*. It is not difficult to imagine a number of others as well. In any case in which we want to compare the frequency or existence of one word or grammatical construction with that of another, subqueries will be of great value. For example, we could easily determine which nouns occur most frequently with one particular semantic field (such as 20–30 positive emotions), as opposed to the opposite semantic field (20–30 negative emotions). Likewise, we could easily determine which verbs have the highest degree of occurrence in passive or progressive constructions, by comparing the frequency of the verb in the active voice or simple verb (1-gram table) with its frequency in the passive voice or progressive aspect (2-gram table), and joining the queries on the appropriate lemma. .

These types of subqueries are perhaps the best evidence for the power of the relational databases. It is difficult to imagine how these types of queries could be carried out with a traditional "linear" corpus. In the first place, the query syntax would have to allow fast searches directly by part of speech and lemma. It would have to allow frequency information to be accessed directly as part of the query as well. Even if these two conditions were fulfilled (and we are not aware of any other large corpus that has such capabilities), then the output of each of the two subqueries that we have just shown (e.g. preterite and imperfect) would have to be imported into a spreadsheet or database. The two lists would then have to be matched up and compared against each other. All of this would be quite complicated, to be sure, and would also take a significant amount of time. Even with the alternate CQP approach (e.g. IMS CWB), the interface and search engine do not allow these types of comparisons. With our approach, on the other hand, the query involves one simple step and can be completed in less than ten seconds.

6. Conclusion

In summary, we suggest that the use of large n-grams / frequency tables in relational databases provides corpus creators with a number of important advantages over traditional approaches:

1) It allows for extremely fast retrieval, because of the clustered indexes that can be created on the n-grams tables. For example, in the 100 million word *Corpus del Español*, most searches take less than one second, and even the most

complex queries – involving morphological pattern matching, lemma, part-of-speech, and synonyms in the same query – usually take less than five seconds.

2) Because the annotation resides in relational database tables, it can quickly and efficiently be updated, without having to traverse the entire textual corpus and update tags within the text itself.

3) The n-grams / frequency tables allow users to access frequency information directly, such as rank-ordered lists of words with a particular morphological pattern, those synonyms that occur in one register but not in another, or the most common strings for a particular syntactic construction, which occur with a set frequency in different sub-corpora.

4) The n-grams tables also allow for complex comparisons involving sub-queries, such as which nouns occur with only one of two verbs or adjectives, or which words are synonyms of two different verbs (such as *to jump* and *to run*) or two different nouns (such as *floor* and *level*).

5) The n-grams can also be a powerful tool to help annotate the corpus itself and to help build the lexicon, through the use of sub-queries and the collocational frequencies that can easily be extracted from the tables. This is particularly useful in the annotation of corpora that have minimal or non-existent lexicons.

6) Finally, the modular approach, which relies on linked relational databases, allows both the corpus creator and the end user to join together an unlimited number of annotation tables (or user-defined databases) of virtually any complexity, with no decrease in performance.

Notes

1. All CQP of which we are aware (as with the IMS CWB) run on Unix/Linux systems. In theory, one should be able to use these programs on a Windows-based machine by using a "dual-boot" approach. However, as we will see, the hardware and software demands are often quite heavy, in terms of being able to produce fast retrieval (2–3 seconds) from a 100+ million word corpus. With a dual-boot approach, the system resources will of necessity be divided between the two operating systems, which may produce unacceptable results.

2. The 100 million words of text from flat files are imported into MS SQL Server in 1000–2000 word chunks, where they are indexed with the "Full-Text Indexing Service". These tables have essentially no annotation, beyond a unique text and text block identifier, which are linked to other tables containing "metadata": author, date, size, etc. This sequential text is used only to provide the KWIC display like that of Table 2. Once the n-gram and annotation tables are created (the focus of this paper), then all queries involving POS, lemma, synonyms, etc. do not directly involve the textual corpus itself, but rather the n-gram and annotation databases.

3. The *Corpus del Español* uses MS SQL Server as its relational database. Other less expensive, open source relational databases such as MySQL may also provide the needed robustness to handle a databases with tens of millions of rows. On the other hand, desktop programs like Microsoft Access typically are useful only for databases containing about half a million rows or less.

4. The *Corpus del Español* uses Active Server Pages (ASP) for the scripting of the dynamically-created web pages, but other technologies such as PHP would probably work just as well.

References

Aston, G. & Burnard, L. (1998). *The BNC handbook: Exploring the British National Corpus with SARA*. Edinburgh: Edinburgh University Press.

Burnage, G. & Dunlop, D. (1993). Encoding the British National Corpus. In J. Aarts et al. (Eds.), *English language corpora: Design, analysis and exploitation. Papers from the 13th International Conference on English Language Research* (pp. 79–95). Amsterdam: Rodopi.

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

Christ, O. (1994). *The IMS Corpus Workbench Technical Manual.* Institüt für maschinelle Sprachverarbeitung, Universität Stuttgart.

Christ, O. & Schulze, B. (1995). Ein flexibles und modulares anfragesystem für textcorpora. In *Tagungsberichte des Arbeitstreffen Lexikon + Text.* Tübingen: Niemeyer.

Garside, R. (1995). Grammatical tagging of the spoken part of the British National Corpus: A progress report. In G. Leech, G. Myers & J. Thomas (Eds.), *Spoken English on Computer: Transcription, Mark-up and Application* (pp. 161–7). Harlow: Longman.

Leech, G., Garside, R., & Bryant, M. (1994). The large-scale grammatical tagging of text: Experience with the British National Corpus. In N. Oostdijk & P. De Haan (Eds.), *Corpus-based research into language* (pp. 47–63). Amsterdam: Rodopi.

Olsen, M., & Hinkelman, E. (1991). *Problems of Large Literary Databases: Morphological Analysis of ARTFL* (CILS Technical Report). Chicago: University of Chicago.

Author's address

Mark Davies
Associate Professor of Linguistics
Department of Linguistics and English Language
Brigham Young University
4064 JFSB
Provo, Utah 84602-6278
USA
E-mail: Mark_Davies@byu.edu