Annotation without lexicons: an alternative to the standard bootstrapping approach

Mark Davies
Illinois State University

1. Introduction

A fundamental problem facing the creators of corpora for less-common languages – or the older stages of an established language – is the lack of suitable lexicons to annotate the corpus. Typically, the annotation of these languages involves "bootstrapping", which refers to the process of starting with the most frequent forms (or some of the morphologically most predictable forms, or both) and progressively annotating the corpus as one also constructs the lexicon. As the annotation process continues, one can deal with less common or less regular forms, because of the predictive capacity of the increasingly robust lexicon and the increasingly rich, annotated context in the corpus itself . (For papers dealing with recent approaches to bootstrapping for a wide range of languages, see Rocio et al 1999, van Eynde et al 2000, Simov et al 2002, Cucerzan et al 2002, Ghani et al 2002, Moreno et al 2003).

A standard approach to corpus annotation is to recursively traverse the entire textual corpus itself, searching for textual patterns, and then adding annotation to the corpus. This can be done either by large-scale pattern matching and replacement (using regular expressions or a similar schema), or by processing sequential chunks of text in a buffer (where only a small window of text is used to disambiguate part of speech (POS) and lemma). The important point is that the annotation typically takes place directly in the textual corpus itself.

In this paper, we will outline an alternative schema that was used to annotate the Corpus del Español (www.corpusdelespanol.org) – a 100 million word corpus of Spanish texts from the 1200s-1900s. The annotation of this corpus was done without directly dealing with or seeing the actual textual corpus itself. Rather, the annotation was done on tables containing all of the distinct 1, 2, 3, and 4-word sequences (n-grams) in the entire corpus, along with the frequency of each of these n-grams in each historical period and modern register of Spanish. As we will see, this non-traditional approach affords a number of important advantages, both in terms of the flexibility and speed of the search engine for the corpus.

For the purposes of this paper, we will focus primarily on the method of annotating the older stages of the language – the texts from the 1200s through the 1500-1600s. The lexicon that we created for Modern Spanish was able to annotate the texts from the 1800s-1900s quite well, but was progressively less useful for older stages of the language. For example, only 40% of the types from the 1700s appear in the Modern Spanish lexicon, and this decreases to 33% for the 1500s and 16% for the 1200s. In other words, most of the types from older historical periods were from a different "language", as far as the lexicon was concerned. We will see, however, that by using an approach based on n-grams tables in relational databases, we were still able to annotate tens of thousands of distinct word forms from the oldest stages of the language in a matter of just a few hours.

2. Corpus architecture and design

Before discussing in detail the way in which the annotation is carried out with the aid of large relational databases of n-grams, let us briefly consider the overall organization of the Corpus del Español. The actual textual corpus for the 100 million word corpus is stored as 1000-2000 word chunks of text in a Microsoft SQL Server 7.0 database. This textual corpus itself is not annotated in any way, apart from a code that indicates the source of each block of text. However, it is indexed with SQL Server "Full-Text Indexing", which is similar to the standard Microsoft Search engine. This indexing scheme allows exact words and phrases to be found fairly quickly – usually less than one second to query the entire corpus and return the relevant examples. The important limitation, however, is that the Full-Text search engine for SQL Server only works well with exact words and phrases. Even wildcard searches are problematic, and certainly there is no capability for customized annotation of any sort.

The annotation for the Corpus del Español resides in relational databases, which are completely separate from the textual corpus itself. These databases are composed of different tables for all of the 1, 2, 3, and 4-grams in the corpus. These tables also include the frequency for each of these n-grams in each of the centuries from the 1200s-1900s, as well as the different registers of Modern Spanish. The data for these tables was generated from the textual corpus itself using the WordList function of WordSmith. This program was run separately for several three to four million word blocks of text, and then merged together in the SQL Server tables.

As might be imagined, the tables are rather large, since they include all of the distinct 1, 2, 3, and 4-grams in the entire corpus. 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. An example of one of the forty million 3-grams from the corpus is the following:

Table 1. N-grams/frequency table

w1	w2	w3	x12	x13	x14	x15	x16	x17	x18	x19	19-Lit	19-Oral	19-Misc
son	las	cosas	38	16	77	67	16	19	33	68	24	40	14

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. Each of these relational database tables is indexed, including some clustered indices (to be discussed in more detail later on), all of which leads to very fast retrieval.

3. Queries without annotation

As this level, however, there is still no annotation per se – only the textual corpus and the n-grams tables. Even at this level, however, there are some useful queries that can be run against the database. For example, a user can input any of the following queries into the web-based form:

Table 2. Queries without annotation

QUERY	SORT BY	LIMITS
tan * como	1900s	
*ización	1900s	1900s>5 –1800s
quer* lo/la/los/las *r	1200s	+1200s -1500s

The first 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. This will return strings like *tan bueno como* "as good as", *tan rápido como* "as fast as", etc. The second query will search the 1-grams table for all of the records where the word in the [w1] column ends in [-ización], the value in the [x19] column is more than 5, and the value in the [x18] column is 0 (meaning that the word appears for the first time in the 1900s). This will return strings like *privatización, globalización*, and *urbanización*. The final example will return strings like *queriendo lo fazer* "wanting to do it" and *queremos las dezir* "we want to say them", which represent cases of a form of *querer* "to want" followed by a direct object pronoun, followed by an infinitive. This query will search the 3-grams table for all records where the [w1] column is has the pattern [quer-], the w2 column is one of the following (lo, la, los, las), the word in the [w3] column ends in [-r], the [x12] column is greater than 0, and the x15 column is 0 (meaning that the phrase is found primarily in Old Spanish).

The simple search syntax of the third query in the table above is transformed via web-based scripting into the following SQL statement, which is then run against the database, and which returns the results in less than one half of a second.

```
select top 300 * from x3 where
w1 like (‘quer%') and
w2 in ('lo', 'la', 'los', 'las') and
w3 like ('%r') and
x12>0 and x15= 0
```

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 Table 2 above. However, at some point it will obviously be necessary to have more complete annotation, including annotation for those lemma 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). The major focus of this paper, then, is the way in which this can be done using collocational and frequency information from the n-grams tables themselves, and the challenge that this presents for languages (or stages of a language) for which we do not have a lexicon.

4. Annotating the corpus: inheriting information from related lexicons

Before turning to the basic question of how to enable "bootstrapping" via n-grams tables in relational databases, however, let us first consider a related and somewhat less difficult scenario. Imagine that there is a lexicon for the modern stage of a particular language, but that the need exists to annotate an older stage of the same language. Obviously, some of the forms from the modern language will be applicable to older stages, but this lexicon will become progressively less useful the farther back one goes. For example, as we have previously mentioned, 40% of the types from the 1700s in the Corpus del Español appear in the Modern Spanish lexicon, and this decreases to 33% for the 1500s and 16% for the 1200s. To the degree that there is similarity in types between the older and newer stages, however, perhaps the best strategy for annotating the corpus is simply to use the database to "inherit" features from related forms in the modern language.

Let us briefly consider how this "inheritance" of annotation features has been carried out with the Corpus del Español. First, we used the frequency information in the n-grams tables to identify the highest frequency forms from older stages of the language, for which lemma and/or POS has not been applied from the Modern Spanish lexicon. For example, a simple SQL query like the following would produce a rank-ordered list of the 300 most common words from the 1500s (x15 column) in the x1 table (single words), where the word (w1) is the same as a word in the lemma table (x_l), which does not have a lemma assigned (column x1) (a similar query could be run for other centuries, as well as for the POS table [x_c]):

```
select top 100 x1.x15,x1.w1
from x1,x_l
where x_l.x1 is null and x_l.w1 = x1.w1
order by x1.x15 desc
```

This list of the most common unannotated forms for a particular historical period can be INSERTed into another table called "unannotated". The corpus creator would then manually go through this list and type into an adjacent column the modern forms that correspond to the older, unannotated forms, in those cases where there is such a correspondence. For example, in this query from the Corpus del Español, some of the unannotated forms that appear are *començó, cavallos*, and *hazían*, which correspond to the modern forms *comenzó, caballos*, and *hacían* "3SG-started, horses, 3PL-made". Once the modern forms are entered into the "unannotated" table along with the modern form, we use a simple SQL UPDATE command to "copy" the POS and lemma values for the modern forms and apply these to the older forms.

In addition to looking at the unannotated forms with the highest frequency, we can also search for forms according to regularized phonetic or morphological changes between the modern language and older stages of the language. For example, there was a regularized shift from [-zi-] to [-ci-] in Spanish, and the following query will find the 1000 most common, unannotated forms from the 1500s that have the pattern [-zi-], and which correspond to an annotated [-ci-] word in the modern lexicon, such as the older forms *haziendo, juizio*, and *vezinos*.

```
select top 1000 x1.x15,x1.w1
from x1,x_l
where x_l.x1 is null and x_l.w1 = x1.w1
and x1.w1 like '%zi%' and patindex(x1.w1,'%zi%') in
(select patindex(w1,'%ci%') from x1
where w1 like '%ci%')
order by x1.x15 desc
```

Once these older forms are INSERTed into an "unannotated" table, a simple REPLACE command can be used to place the modern form into another column, and a subsequent UPDATE query would copy the modern Spanish POS and lemma values to the older forms. In this way, with a knowledge of some of the basic phonetic, morphological, and orthographic changes in the language, it is possible to annotate thousands of forms from older stages of the language in a matter of a few hours.

5. POS annotation with n-grams/frequency information and pattern matching

In the previous section we assumed a more optimistic scenario, in which there is some type of related lexicon that can be applied to our corpus. Let us now turn to the more pessimistic scenario, in which there is no lexicon at all and we are simply "working from scratch". The only assumption here is that we have the n-grams/frequency tables and the relational database structure that we have previously discussed, but there are no other annotation tools available to us.

In order to assign POS, at the most basic level we will simply use SQL commands to select the most common word forms that have a certain morphological pattern. For example, the following SQL query selects those forms that end in [-ADO/-ADA/-ADOS/-ADAS], which is the typical marker of the past participle:

```
select top 100 x12, w1
from x1
where w1 like '%do' or w1 like '%da' or w1 like '%dos' or w1 like '%das'
order by x12 desc
```

The one problem with such queries, however, is that they also retrieve many items that are morphologically similar, but which do not in fact belong to the desired grammatical category. For example, the preceding query would retrieve (-DO) *quando, grado, mando*; (-DA) *espada, nada, cada*, (-DOS) *todos, dos*; and (-DAS) *espadas, todas* – none of which are past participles.

Fortunately, we can use SQL sub-queries to limit forms that have overly-generalized patterns (e.g. -DO, -DA, etc), by comparing them to other forms with which they share a predictable morphological relationship. For example, the past participle of [-AR] verbs is (nearly always) formed by removing the [-AR] of the infinitive, and replacing it with [-ADO]. We can therefore include in the SQL query a sub-query that checks to see whether the "root" of an –ADO form (i.e. remove the –ADO) can also be found as the root for an infinitive (i.e. remove the –AR):

```
select top 100 x12, w1
from x1
where w1 like '%ado' and len(w1) >=3 and
left(w1,len(w1)-3) in
(
select left(w1,len(w1)-2)
from x1 where
w1 like '%ar' and len(w1) >=2
and x12 > 10
)
order by x12 desc
```

Using this sub-query, we eliminate spurious [–ADO] forms like *grado, obispado*, and *prelado*, because there are no corresponding infinitives with the (hypothetical) form **grar, *obispar*, and **prelar*.

In addition, we can use sub-queries to find derived forms of grammatical categories. For example, in searching for past participles ending in [–ADA] (fem sg), we can have the sub-query check to see whether the corresponding [–ADO] (m sg) form also exists (and at a certain frequency, such as 10 occurrences in the 1200s):

```
select top 100 x12, w1
from x1
where w1 like '%ada' and
left(w1,len(w1)-1) in
(
select left(w1,len(w1)-1)
from x1 where
w1 like '%ado' and
x12 > 10
)
order by x12 desc
```

This would eliminate spurious "[-DA] past participles" like *cada* "each" and *espada* "sword" because there are no corresponding [–DO] past participles like the hypothetical forms **cado* and **espado*.

6. POS annotation with n-grams and collocation information

In the preceding examples we have seen how POS annotation can be done by looking for morphological patterns (prefixes, suffixes, etc), and how this can be enhanced through the use of sub-queries, which test for the existence of related forms. In all of these cases, however, we have been dealing with queries that look at individual word forms. The real strength of the n-grams approach comes when we begin to limit queries based on the occurrence of given words within a certain n-gram sequence.

For example, suppose that we want to identify the 1000 most common infinitives in the Spanish of the 1200s. We could do a simple search on the 1-gram table in which we search for words (w1) ending in [-R] (the marker of the infinitive), and produce a rank-ordered list. The following query would select these forms, but would also produce false entries like *por, quier, mayor, muger,* etc.

```
select top 1000 x12, w1
from x1
where w1 like ‘%r’
order by x12 desc
```

An alternative strategy – and one that relies on the inherent strengths of an n-grams approach – would be to search the 2-grams table for words that end in [-AR / ER / IR / YR], but which also occur after some of the most common forms of one of the three auxiliary verbs *poder, querer*, and *deber*.

```
select top 1000 w2,sum(x12)
from x2 where
(w1 like 'quer%' or w1 like 'quis%' or w1 like 'quier%' or w1 like 'pod%' or w1 like
'pued%' or w1 like 'dev%' or w1 like 'deb%' or w1 like 'deu%') and
w2 like '%r'
group by w2
having sum(x12) > 2
order by sum(x12) desc
```

This query is much more accurate than the 1-gram query shown previously (producing only one false hit in the top 200 word forms), and takes less than one second to run.

The following are some additional searches that show how the n-grams tables can be used to annotate parts of speech. In each case, we want to identify a two or three word syntactic environment in which nearly all of the words in one of the “slots” belong to a particular syntactic category. For example, suppose that we want to identify the 2000 most common nouns in the 1200s. A syntactic environment in which these would occur is [indef art] + __ + [que] (*un omne que* “a man that”, *vna casa que* “a house that”, etc). The following query – which takes less than one second to run – selects the 2000 most frequent words in slot 2 (w2) in the 3-grams table (x3), which occur more than two times in that slot and which are preceded by a word (w1) that is (*una, una, vn, vna*) and which are are followed by *que* “that” in the third (w3) slot:

```
select top 2000 w2,sum(x12)
from x3 where
w1 in ('un', 'una','vn', 'vna') and
w3 = 'que'
group by w2
having sum(x12) > 2
order by sum(x12) desc
```

Likewise, the following query – which can be run against the entire 100 million word corpus in less than one second – is an attempt to define a syntactic environment for adjectives. It selects all words in the third slot (w3) of the 3-grams table (x3), in which the first slot (w1) is one of several high-frequency forms of *ser* “to be”, and the second slot (w2) is a form of *muy* “very” or *tan* “so”:

```
select w3,sum(x12)
from x3 where
w1 in ('es','era','será','sera','fue') and
w2 in ('muy','mui','muy'',’tan’)
group by w3
having sum(x12) > 2
order by sum(x12) desc
```

Finally, let us return to the category of past participle, which we considered in the previous section. Recall that in the case of pattern matching with individual 1-grams, it was fairly difficult to morphologically define a past participle (words that typically end in –ADO /-ADOS / -ADA / –ADAS), because of the number of false hits like *cada, dos*, and *espada*. By using the 2-grams or 3-grams table, however, we can constrain the search much better, by selecting only those forms that occur after one of the auxiliary verbs *haber* “to have”*,* or *estar* or *ser* “to be”, which is the real evidence for a form being a past participle In the following query, we search the 2-grams table (x2) to select those forms (w2) that end in [–ADO /-ADOS / -ADA / –ADAS], which are preceded (w1) by some of the highest frequency patterns matching forms of *haber*, *ser* and *estar*.

```
select top 1000 w2
from x2 where
(w1 in (‘ha’,’has’,’han’) or w1 like ‘ha__a%’ or like 'esta_a%' or w1 in
('fue','fueron','era',’eran’)) and
(w2 like '%ado' or w1 like '%ada' or w1 like '%ados' or w1 like '%adas')
group by w2
having sum(x12) >= 2
order by sum(x12) desc
```

As a final note regarding POS annotation, we should mention the progressive nature of the process. As with other methods of bootstrapping, our queries will become progressively more refined and powerful. For example, once we have accurately defined the 500 or 1000 most frequent nouns, we can then use the category [noun] to look for the members of other categories. In other words, it is not necessary for us to continually be doing queries “from scratch”, as in the examples given here.

7. Lemmatization with n-grams, frequency, collocates, and pattern matching

In addition to POS tagging, the n-grams/frequency tables can also be used to help with lemmatization. Probably the most obvious application is to search for word forms that have a certain morphological pattern. For example, the following query will list the sixty most frequent words (w1) in the 1200s, where the word pattern is ‘quer*’ (*querya, querremos*), ‘qu_er*’ (*quiere, qujeren*) or ‘quis*’ (*quisyere, quise*), which are all Old Spanish forms of *querer* “to want”:

```
select top 100 x12,w1 from x1
where w1 like 'quer%' or w1 like 'qu_er%' or w1 like 'quis%'
order by x12 desc
```

We may determine, however, that a number of the morphologically similar word forms actually belong to another lemma. In the example above, for example, a number of the matching forms belong to the Old Spanish verb *querellar* “to quarrel”. To further limit the forms, we may wish to provide a syntactic context in which only the forms of *querer* “to want” will be found, such as those forms that precede an infinitive. The following query does this by using a sub-query to limit the [*que*-] forms to just those that occur immediately before an infinitive (w2; [-AR/-ER/-IR/-YR] for Old Spanish) in the table of 2-grams (x2), which is a syntactic environment in which *querellar* “to fight” would likely not occur:

```
select top 100 x12,w1
from x1
where (w1 like 'quer%' or w1 like 'qu_er%' or w1 like 'quis%')
and w1 in
(
select w1 from x2
where (w1 like 'quer%' or w1 like 'qu_er%' or w1 like 'quis%') and
(w2 like '%ar' or w2 like '%er' or w2 like '%ir' or w2 like '%yr')
)
order by x12 desc
```

By combining pattern matching (e.g. ‘qu_er*’), collocational information for multi-word n-grams tables, and frequency information for each of these n-grams, it is possible to lemmatize hundreds or thousands of words in just a few hours, even when there is no lexicon available.

Finally, we should once again keep in mind the progressive and cumulative nature of the annotation process. The more we have defined a particular grammatical category or a particular lemma, the easier it will be to find members of additional categories or lemma. For example, in attempting to find the members of the lemma [*querer*] “to want”, we have looked at n-grams in which the word in the following “slot” ends in [-AR / -ER / -IR / -YR] (in Old Spanish). Yet at certain point we will have refined the [V_INF] category to the point that we can use it directly in the queries. Likewise, once we have identified the forms of a particular lemma, we can then use that lemma directly in subsequent queries. For example, rather than using ‘quer*’ and ‘qu_er*’ to define [*querer*], at a certain point we can refer to the lemma [*querer*] directly as we search for other lemma and grammatical categories. This is of course similar to more traditional methods of bootstrapping, which perform progressively more refined annotation on the textual corpus itself.

8. The advantage of n-grams databases vs. traditional approaches

In the preceding sections, we have shown how forms can be annotated, based on information from n-grams and frequency in relational databases. This is a much less orthodox approach than the standard approach, which is to use regular expressions (or other pattern matching schemes) to search for strings and patterns in the textual corpus itself, and then insert the annotation into the textual corpus. However, there are certain clear advantages to the relational database / n-gram approach, as we discuss in this section.

First, our approach is probably much more economical and efficient than the standard approach. As has been mentioned several times, even the most complex queries on tens of millions of records in the n-grams tables take only one or two seconds to run. In the standard approach, each query may involve a new traversal of the entire textual corpus. This is much less of a problem for small corpora (one million words or less), but may quickly become prohibitive for corpora containing tens of hundreds of millions of word of text.

Second, the relational database approach is able to take advantage of sub-queries, which allows it to perform complex multi-stage tests on the data. For example, in order to search for a 1SG form of the present tense (*fablo* "I speak"), the query could have a number of sub-queries: check to see if the form ends in [-O], check whether the verbal root is also the root for a relatively high frequency infinitival-like form (i.e. with the [-AR/-ER/-IR/-YR detached), and check whether the form occurs in a multi-word sequence (e.g. a 2- or 3-grams table) where (1SG) present tense verbs would likely occur. Although the database actually performs these searches in sequence, the intermediate result of each sub-query is stored in a temporary table and is automatically processed by the database program.

In the standard approach, it would likely be much more complicated. After each traversal of the entire corpus (to look for high frequency, morphologically-related forms, or word sequences), the program would have to store the temporary results and then re-use these as part of subsequent traversals of the corpus. Depending on the efficiency of the script used to process the data, this could quickly become too complicated or prohibitive, in terms of speed and performance.

Third, the n-grams approach naturally lends itself to advanced collocational analysis of the data. For example, in examining the top one hundred forms of suspected adjectives, it might be useful to see the fifteen or twenty most frequent 3-grams for each word form, in which the word form occupies the middle slot. With the relational database approach, the sorted results could be produced (using sub-queries) in two or three seconds. While it would certainly be possible to produce the same listing using the standard approach, it is likely that it would be somewhat more complicated than the five or six lines of code in the relational database SQL command.

Fourth, the relational database approach allows one to easily select and deselect word forms that are potential members of a syntactic category or particular lemma. For example, the SQL command can INSERT 100 or 1000 probable forms into a "temporary" table, along with the highest frequency preceding and following words, if so desired. These forms can then be quickly reviewed and the value for an "action" column can be quickly set to a particular value for those word forms that belong to the grammatical category or lemma. Another UPDATE command can then go back and assign the particular POS or lemma to all of the matching forms in the actual n-grams databases, for those items whose "action column" has been set to a certain value. In the standard approach, it would likely not be as intuitive or natural to review a list of suspected forms, select a subset of these, and then annotate the corpus itself based on which items have been selected.

The fifth and final advantage of the relational database / n-grams approach is also perhaps the most important one. In this approach, the n-grams/frequency tables are simply one part of the overall database, albeit the most important part. But because they are in a relational database, they can very easily be joined to other tables within the same database, and there is no limit to the amount of annotation that can be applied to the corpus. For example, in the Corpus del Español, there is a table containing the synonyms for 30,000 words, and users can access this information as part of their search. An example of this is the following query, which searches for all forms of all lemma that are synonyms of *mandar* "to order, command", followed by the subjunctive marker *que* "that", followed by a past subjunctive (*mandé que fueran* "I made them leave"; *hicieron que dijera* "they made her say"):

```
!mandar.* que *.v_subj_ra
```

Likewise, users of the Corpus del Español can create customized lists of words that are morphologically, syntactically, or semantically related, such as adjectives describing emotions, words ending in [-azo] that denote a blow or strike, or a list of temporal adverbs. These 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”). The user can then use this list as part of the following syntactic construction to return phrases like *me gusta que haya* “it please me that there is”, *le sorprende que tengan* “it depresses him/her that they have”. The following is the query as it is entered into the search form, along with a description of the queryL

me/nos/te/os/le/les [Jones:emotions].* que *.v_subj_pres

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

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

```
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 ('gustar', 'sorprender', 'agradar', 'alegrar')) and
w3 in ('que') and
w4 in (select w1 from x_c where x1 in ('v_subj_pres'))
```

The important point is that there can be an unlimited number of levels of annotation on the corpus – 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 based within the textual corpus itself.

9. The design and architecture of annotation in the relational database

As explained previously, there are two approaches to the placement of annotation in the relational databases. One approach would be to include it as additional columns in the n-grams/frequency databases themselves. For example, in the following case there is annotation (POS and lemma) within the table for each of the three words in the 3-gram *son las cosas* “are the things”

Table 3: N-grams/frequency table with integrated POS and lemma annotation

W1	L1	C1	W2	L2	C2	W3	L3	C3	x12	x13	x14	x15	x16	x17	x18	x19	19-Lit	19-Oral	19-Misc
son	ser	vp	las	lo	adef	cosas	cosa	N	38	16	77	67	16	19	33	68	24	40	14

In this scenario, for example, a user who is searching for cases of a form of *poder* + *estar* + a present participle (e.g. *puede estar pensando* “3SG-can be thinking”) would enter the following in the web-based form:

poder.* estar *.v_ndo

This is then translated into the following SQL command, which queries only the 3-grams table (x3), since it has all of the POS and lemma information included within that one table:

```
select top 300 * from x3 where
L1 = ‘querer’ and
W2 = ‘estar’ and
C3 = ‘v_ndo’
```

The alternative is to have the n-grams/frequency information in one table (x3), and the POS and lemma information in two additional tables (x_c and x_L, respectively):

Table 4: 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
son	las	cosas	38	16	77	67	16	19	33	68	24	40	14

x3

w1	x1
es	v_pres
será	v_fut
sido	v_pp

pos (x_c)

w1	x1
es	ser
cosa	cosa
cosas	cosa

lemma (x_L)

In this scenario, there would be simple SQL JOIN queries 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):

```
select top 300 * from x3 where
w1 in (select w1 from x_L where x1 in ('poder')) and
w2 in ('estar') and
w3 in (select w1 from x_c where x1 in ('v_ndo'))
```

So which of the two approaches produces the best results? The advantage of having the annotation within the n-grams / frequency table itself is that the database creator can use contextual information to resolve ambiguity. For example, in the 3-grams table two of the 40+ million records are for the strings *el poder de* "the power of" and *para poder* "in order to be able":

Table 5. Contextual disambiguation, based on n-grams

W1	L1	C1	W2	L2	C2	W3	L3	C3	x12	x13	x14	x15	x16	x17	x18	x19	19-Lit	19-Oral	19-Misc
el	el	a_d	poder			de	de	prep	192	57	54	200	135	93	360	255	67	36	152
para	para	prep	poder			ser	ser	v_inf	0	0	2	11	5	3	13	37	6	17	14

In Spanish, *poder* can either be a noun (*el poder* "the power") or an infinitive (*poder saber* "to be able to know"). Therefore, it is not immediately clear what the POS or the lemma should be for the [*poder*] in [w2] slots.

Using simple SQL updates, however, we can look at the contextual information to fill in or alter the POS and lemma values, based on the values in the other word slots. For example, the following UPDATE query will assign the value of [noun] to [*poder*] when it occurs before a preposition (as in the first row) and it will assign a value of [v_inf] when it is followed by another infinitive (the second row). The second case is shown in the following example:

```
update x3
set c2 = 'v_inf'
where w2 = 'poder' and c3 = 'v_inf'
```

Judging from languages like English – which have such a high degree of polysemy – it may seem unwise to have to run a large amount of SQL updates – such as the one just shown – to improve the annotation. Yet it turns out that because Spanish is such a morphologically strong language, there are relatively few forms (as with *poder*) that have a high frequency as two different parts of speech or two different lemma.

Because of this very low level of polysemy, we might take a more radical step and completely remove the annotation from the n-grams/frequency tables, as shown in Table 4 above. In this case there would be two entries for *poder* in both the POS and lemma tables, and in most cases, the queried phrase will naturally disambiguate itself. For example, if a user searches for [de *.v_inf *.v_inf], then [*de poder saber*], [*de poder estar*] and others will be retrieved, because *poder* is listed as a [v_inf] in the POS table. And it turns out that all of these examples of the potentially polysemous *poder* really are in its use as an infinitive. Likewise, if a user searches for [el *.n de], then [*el poder de*] "the power of" will be retrieved, and again virtually all of these cases will be with *poder* as a noun (because of the following preposition). There will be problems only in those relatively few cases in which 1) a word is polysemous and 2) both meanings are highly frequent and 3) the context is not sufficiently rich to disambiguate the multiple meanings.

Assuming that it is the case that ambiguity in POS and lemma assignment is much less of a problem in a morphologically complex language like Spanish (as opposed to a morphologically weak language like English), then we can take the further step of simply removing the annotation from the n-grams/frequency tables altogether (where they might be useful for contextual disambiguation) and place them in separate POS and lemma tables. In fact this is precisely what we have done in the Corpus del Espanol, and more than a year's worth of complex searching on the corpus suggests that it very rarely leads to any problems.

One of the major advantages of placing the annotation in tables that are separate from their context 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 also 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 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. 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. This is why queries of the 100 million word Corpus del Español typically take just one or two seconds – for even more the most complex queries, involving POS, lemma, word patterns, synonyms, and user-defined lists of words.

References

Cucerzan S, Yarowsky D 2002 Bootstrapping a multilingual part-of-speech tagger in one person-day. *CoNLL-2002 Sixth Conference on Natural Language Learning*. Taipei, Taiwan, pp. 132-38.

Ghani R, Jones R 2002 A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems. In *Proceedings of LREC 2002 Workshop on "Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*". Spain: Las Palmas, pp. 61-72.

Moreno, A., López S, Sánchez F 2003 Developing a syntactic annotation scheme and tools for a Spanish treebank. In Abeillé A (ed), *Building and using syntactically annotated corpora*. Dordrecht: Kluwer, pp. 149-65.

Rocio V, Pereira G 1999 Análise sintáctica parcial em cascata". In: Marrafa P, Mota M (eds), *Linguística Computacional: Investigação Fundamental e Aplicações*. Lisboa: Edições Colibrí, pp. 235-251.

Simov K, Kouylekov M, Simov A 2002 Incremental specialization of an HPSG-based annotation scheme. In *Proceedings of LREC 2002 Workshop on "Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*". Spain: Las Palmas, pp. 16-23.

van Eynde F, Zavrel J, Daelemans W 2000 Lemmatisation and morphosyntactic annotation for the spoken Dutch corpus. In M. Gavrilidou et al. (eds), *Proceedings of the Second International Conference on Language Resources and Evaluation*. Paris: ELRA, pp. 1427-1433.