Using An N-Gram-Based Document Representation With A Vector Processing Retrieval Model

William B. Cavnar
379 Brookside
Ann Arbor MI 48105
wcavnar@mail.msen.com

1.0 Abstract

N-gram-based representations for documents have several distinct advantages for various document processing tasks. First, they provide a more robust representation in the face of grammatical and typographical errors in the documents. Secondly, N-gram representations require no linguistic preparations such as word-stemming or stopword removal. Thus they are ideal in situations requiring multi-language operations.

Vector processing retrieval models also have some unique advantages for information retrieval tasks. In particular, they provide a simple, uniform representation for documents and queries, and an intuitively appealing document similarity measure. Also, modern vector space models have good retrieval performance characteristics.

In this work, we combine these two ideas by using a vector processing model for documents and queries, but using N-gram frequencies as the basis for the vector element values instead of more traditional term frequencies. The resulting system provides good retrieval performance on the TREC-1 and TREC-2 tests without the need for any kind of word-stemming or stopword removal. We also have begun testing the system on Spanish language documents.

2.0 Background

2.1 N-Gram Representations

An N-gram is an N-character slice of a longer string. For example, we could represent the word TEXT with the following N-grams:

- bi-grams: _T, TE, EX, XT, T_
- tri-grams: _TE, TEX, EXT, XT_
- quad-grams: _TEX, TEXT, EXT_

where the underscore character represents a leading or trailing space. N-grams provide a distributed representation, in which each word is represented by a set of N-grams. Words that are similar, say by virtue of having a different suffix or a spelling variation, will nonetheless share a large number of N-grams. For example, consider the words RETRIEVE, RETRIEVAL, and RETRIEVING, which share the bi-grams _R, RE, ET, TR, RI, IE, and EV. Likewise, RETRIEVE and its frequent misspelling RETREIVE share most of their N-grams.

We built several systems using this notion of N-gram-based representations to perform various retrieval tasks. The first was a simple text retrieval tool called *zview* which served as an informal corporate database at ERIM. *zview* provided very easy-to-use access to a set of ASCII documents since it tolerated spelling and word order variations in queries, and required no Boolean search expressions.

Another example of N-gram-based representations was in some of our work for the U.S. Postal Service [Cavnar93a]. In conjunction with USPS-sponsored research on address interpretation, we built an N-gram-based system for matching city, state and ZIP information off of an address in order to determine a set of relevant ZIP codes. This was quite important, given the realities of postal patrons' addressing errors, peculiar local addressing conventions, errors in the Postal databases, and errors in OCR systems attempting to read low-quality address images. Thus, one could not count on any one piece of information in an address as being correct. It was only the coordinated redundancy inherent in an N-gram-based representation that allowed this system to maintain a high retrieval performance in the face of multiple errors in the city, state and ZIP information.

2.2 TREC

The TREC competitions [Harman94] are a series of conferences sponsored by ARPA and NIST specifically to stimulate the advancement of the art of information retrieval by

- Providing large standardized datasets, query sets, and relevance assessments for measuring retrieval performance, and

- Providing a conference where both commercial vendors and academic researchers may meaningfully exchange approaches and results. Furthermore, these conferences are high in content and reasonably free of advertising hype.

For TREC-2, we fielded a retrieval system that used an N-gram-based representation, but with an adhoc retrieval model [Cavnar93b]. This system had several notable deficiencies:

- It was designed as a filter rather than as a true retrieval system. Thus, every run of the system required a complete pass of all of the data. This process took hours for a single query.

- The system built queries from the topic statements in a very simplistic way, using only the <concept> section of the topics. This was unfortunate, since many very relevant terms appeared only in the narrative portion of the topics. Furthermore, we knew that for TREC-3, at least some of the topic sets would not have <concept> sections at all.

- The system used a completely adhoc method of term weighting based on the order of the terms in the <concept> section of each topic and somewhat on the frequency of occurrence of the terms in each document. Although we tried several different variations, none proved at all adequate.

On the other hand, we did demonstrate that the system could tolerate some level of typographical errors in the queries. This level of errors would have rendered most, if not all, of the other TREC-2 systems completely helpless.

During the same time, we also did some research on using N-gram occurrence frequencies to categorize documents by language and subject [Cavnar94]. Basically, the idea was to compute an N-gram frequency profile for both a set of category samples and for a document. Using a simple distance measure, we then computed distances between the document and each of the category samples. For language categorization, this approach was remarkably successful. Using only the top 300 N-grams by frequency (where N = 1 to 5) we could correctly identify the language of a Usenet posting in the soc.culture hierarchy 99.8% of the time. The system was somewhat less successful in the subject categorization task we tested, but the system performed well enough on certain parts of the task to suggest that the approach merited further inquiry.

2.3 Vector Processing Retrieval Models

The most common type of text retrieval system uses complicated Boolean expressions to specify relevant documents by selecting on the presence or absence of terms and combinations of terms. Although researchers and vendors alike have been working on these kinds of retrieval systems for decades, there continue to be difficulties with using them. First, it is difficult for novice users to get satisfactory results because of the complexity, and in some cases, the opacity of Boolean term expressions. Moreover, even for experts, some kinds of information needs are difficult or tedious to specify in Boolean expressions.

Among the alternatives to Boolean retrieval systems is the vector processing retrieval model introduced by Salton over 30 years ago, and continuously researched by him and his group since then [Salton94]. The key concept in vector processing models is representing both documents and queries as vectors in very high dimensional spaces. Typically, each dimension in a vector represents the frequency of occurrence of a term in the document or query. Given all of the possible terms in a document collection, a vector may well have many thousands of dimensions, most of whose specific values are zero for a given document. Viewing documents and queries as vectors suggests one very natural measure of similarity: the angle between two vectors. That is, the smaller the angle between two vectors, the more similar we would consider their respective documents or queries to be. In practice, most vector processing systems actually use the cosine of the angle between the vectors as the actual measure since that is easy to calculate and yields a number between 1.0 and -1.0.

Viewing documents and queries as vectors has many attractions. For example, one could use a document itself as a query, which would enable the system to find documents similar to a given document. One could likewise form a vector to represent a whole set of documents by simply averaging the values of each of their dimensions or performing some other similarly plausible geometric operation on their vectors.

3.0 System Description

Given the success of using N-gram frequency information for text categorization that we mentioned earlier, we felt inspired to find a different approach to the TREC task that made better use of this information. The similarity in spirit between our N-gram frequency profiles and the vectors used in vector processing retrieval models suggested to us a new possibility. We decided to try using a conventional vector processing model, but with N-gram frequencies instead of term frequencies for the dimensions. We also decided to limit the system to just quad-grams for this version.

Although shorter N-grams work very well for name matching, our experience with the text categorization task showed that the longer N-grams are the ones that seem to carry more content or meaning. We used quad-grams composed of digits and letters, and also the leading and trailing space for each word.

This idea is very similar to one independently developed by Thomas and described in [Thomas]. Although that system also used N-grams and a vector processing model there were several notable differences:

- Thomas's system used only tri-grams of letters.
- It ignored all word boundaries due to punctuation, white space, or digits, and treated each line as a very long single word.
- It did not make use of the notion of inverse document frequency.

Also Thomas did not attempt the TREC task, but successfully performed a smaller relevance measurement task retrieving medical records. Our motivation for developing an N-gram-based vector processing system is very similar to that which inspired Thomas, and produces largely the same benefits.

Figure 1 gives a dataflow diagram for the system we designed. We will describe each component of these in some detail.

3.1 *ngram_stats* And The Quad-Gram Dictionary

The first bubble, labeled *ngram_stats,* represents a program which reads the documents and builds a dictionary of quad-grams and their associated information. Among the attributes that the dictionary associates with each quad-gram is its inverse document frequency (IDF). The IDF measures how focused a word is in the collection: a word with a high IDF would occur in few documents, whereas one with a low IDF would occur in most documents. See section 4.0 for a detailed description of the IDF calculation.

The quad-gram dictionary provides the mechanism which allows the system to keep track of each quad-gram in order to apply the appropriate term weighting. After building the dictionary, *ngram_stats* removes all quad-grams that occur only once. Data disks 1 and 2 together had 171514 entries left in the dictionary, while disk 3 had only 136414.

3.2 *ngram_vecs* And Document Vectors

The process represented by the bubble titled *ngram_vecs* reads the document files and produces for each document a vector giving the appropriate weights for each quad-gram with respect to that document. This program dealt with parsing the SGML structure of the document files, splitting the body of the text into quad-grams, counting the quad-grams, and building the actual document vectors according to the resulting quad-gram frequencies.

Even though most of the quad-grams had zero weights for any particular document, prompting us to use a sparse vector representation, we still had a problem with the space required to store the non-zero weights for a document. In general we found that the more quad-grams with non-zero weights that we kept in the vector, the better the retrieval performance. Unfortunately, that also meant that the vector file required more disk space. Happily, the IDF corresponds very strongly with the importance of the quad-gram, and thus makes it easy to threshold on the IDF as a way of limiting quad-grams in the vectors to the most important. We compromised on a minimum IDF threshold of 2.75, which maximized retrieval performance on the several test sets we had available while staying within disk availability. At this threshold, the vector for documents in disks 1 and 2 had an average length of 360 quad-grams out of an average of 1460 possible quad-grams. We did a few informal experiments on disk 3 reducing the minimum IDF threshold all the way down to 2.0. The best performance in this short set of experiments was at an IDF of 2.25. However, even at this level, the improvement was a few percent in precision, and that was at the expense of increasing the storage by over 40%.

The overall effect of this IDF thresholding was very similar to performing word-stemming and stopword removal. Quad-grams with low IDFs correspond precisely to the lexical components that word-stemming and stopword removal would eliminate. The great advantage of the N-gram-based approach is that this elimination is dictated entirely by the statistics of the language. There is no need for a step requiring detailed linguistic knowledge, and thus the system is immediately usable without change on any language that can be represented in ASCII. To illustrate this, we have provided some frequency statistics tables in Appendix I. Table 1 in the Appendix lists the top 20 quad-grams by frequency from disks 1 and 2. Notice that nearly all of these quad-grams are components of words or suffixes that would have to have been eliminated by the more traditional linguistic preprocessing. The one difference is the presence of very frequently occurring prefixes, such as 'PRO', 'COM', and 'CON'. Given their extremely high frequency, these prefixes actually contribute very little semantic content. Likewise, Table 2 shows similar statistics for the Spanish data, and again, the very frequent quad-grams represent those

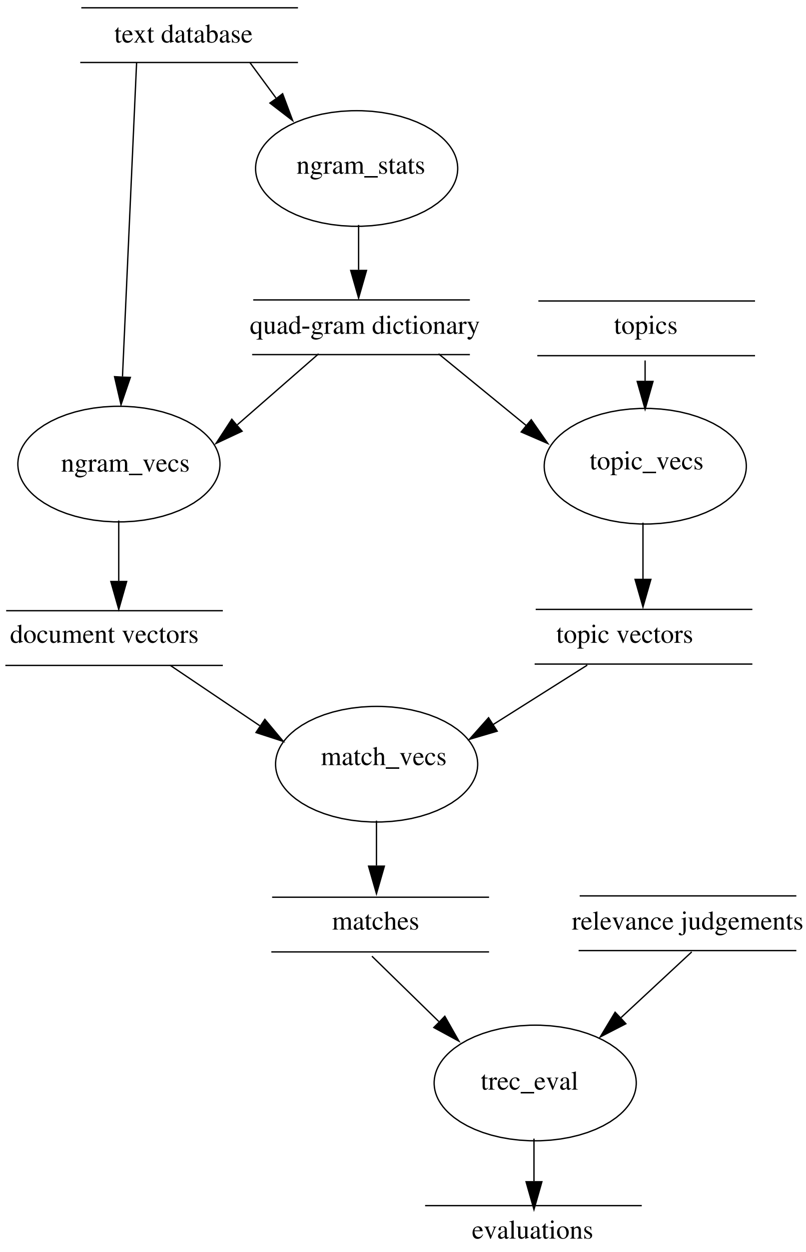

Figure 1: System Dataflow Diagram

words or word parts that word-stemming and stopword elimination would have removed.

3.3 *topic_vecs* And The Topic Vectors

The bubble labeled *topic_vecs* represents a process that is very similar to the *ngram_vecs* process. It reads the topic files and produces for each topic a vector giving the appropriate weights for each quad-gram with respect to that topic. The only significant difference between *topic_vecs* and *ngram_vecs* is that they have to parse two different kinds of SGML structures, and there are some differences in the details of term weight calculations. The format of the actual vectors produced is the same. Figure 2 gives a sample topic and Figure 3 a portion of the corresponding vector for the topic.

3.4 *match_vecs* And Document Retrieval

Once we have a set of document vectors available for every document and topic, we can use them to retrieve documents that are relevant to a particular topic. Following the vector processing model, the process represented by the *match_vecs* bubble performs this retrieval. This process

<top>

<head> Tipster Topic Description

<num> Number: 001

<dom> Domain: International Economics

<title> Topic: Antitrust Cases Pending

<desc> Description:

Document discusses a pending antitrust case.

<narr> Narrative:

To be relevant, a document will discuss a pending antitrust case and will identify the alleged violation as well as the government entity investigating the case. Identification of the industry and the companies involved is optional. The antitrust investigation must be a result of a complaint, NOT as part of a routine review.

...

<\top>

Figure 2: Topic 001 From TREC-1

```
=001
` ANT' 14.851851
`NTIT' 19.151485
`TITR' 32.506756
`ITRU' 31.479759
`TRUS' 20.348030
`RUST' 18.863096
`CASE' 9.288112
`ASES' 3.004144
` PEN' 9.087247
`ENDI' 7.693622
`ISCU' 6.482330
`SCUS' 6.481742
`CUSS' 6.467620
`USSE' 3.795147
`USS ` 5.027708
` IDE' 8.047592
`NTIF' 6.976310
`TIFY' 4.401021
`IFY ` 3.791184
`LLEG' 7.611647
`LEGE' 4.174561
...
```

Figure 3: Portion Of Vector For Topic 001

consists of computing the similarity between the topic's vector and all of the document vectors, and then taking the top 1000 documents by this similarity measure.

For the similarity measure, we are again taking the lead from the classic vector processing model described by [Salton94], which uses the cosine of the angle between the vectors. Since we also normalized all vectors to unit length, this similarity measure thus amounts to computing the dot product between the topic vector and each document vector. This is simple and reasonably efficient given the sparse vector representation. For the full dataset consisting of disks 1 and 2, *match_vecs* typically takes around 20 minutes on a Sun SPARCstation 2 to pass all of the 741856 document vectors and compute the top 1000 scores against a single topic vector. While this is fine for our research task, it is too slow for any realistic use as an interactive tool. However, there are several possible ways to significantly improve search times which we hope to investigate in future versions.

3.5 *trec_eval* And Performance Evaluation

The final bubble labeled *trec_eval* is a program supplied by Cornell University as part of their SMART system, but specifically modified by them to serve as the standard evaluation tool for the TREC task. This program produces some standard reports showing precision and recall values, and is the basis for the comparative reports and graphs in all of the TREC proceedings.

4.0 Term Weighting

In her closing talk at TREC-2, Donna Harman observed that by making simple changes in the term weighting one could drive the performance curve of a good retrieval system down to completely unacceptable levels. Also Salton has explored numerous term weighting schemes, and found that they make very large differences in performance [Salton88]. As we observed earlier, our TREC-2 system used a completely adhoc term weighting scheme that doubtless was a major contributor to its weak performance. For our initial work in TREC-3, we decided to use a simplified version of one of Salton's weighting methods. For documents, our system uses term weights of the form:

$$w_j = \frac{(\log 2\,(tf_j) + 1)}{\sqrt{\sum_i w_i}}$$

where tf_j is the term frequency of the jth quad-gram in the document, and the denominator is a normalization for the document vector for unit length. (log2 refers to the log function in base 2.) For queries, our system uses similar term weights, but which include the IDF (inverse document frequency):

$$w_j = \frac{(\log 2\,(tf_j) + 1) \cdot IDF_j}{\sqrt{\sum_i w_i}}$$

The precise formula we used for the IDF of a quad-gram was:

$$IDF_j = \log 2\left(\frac{N}{n_j}\right)$$

where N is the number of the documents in the collection, and n_j is the number of documents in the collection containing at least one occurrence of the jth quad-gram. This is about the simplest IDF formulation possible, and we have not had a chance to pursue the effects of using one of the more sophisticated versions. See [Harman92] for a more detailed discussion of the IDF, and alternative formulations other researchers have used.

Following a suggestion by Salton, we use the IDF factor only for the query vector so that the IDF factor appears only once for each term in the final dot product between the document vector and the query vector.

5.0 Assessing Performance

The beauty of the TREC task is that there already is a defined performance criterion in the relevance assessments provided by NIST and the standard *trec_eval* tool for actually calculating the performance results. According to *trec_eval* measures, our system appears to be functioning reasonably well. We ran a number of tests using the earlier TREC data sets and query sets. Results for tests on the dataset from Disks 1 & 2 appear in Table 1. 'Average Precision', 'Precision @ 100 docs', and 'R-precision' are some of the most popular and best understood of the performance measures used by other TREC participants. In the Description field of the table, NG93 and NG94 refer to last year's and this year's N-gram-based retrieval systems, respectively. Q1, Q2, Q3, and Q4 refer to the topic query sets 1, 2, 3, and 4. The shaded rows are the results from this year's (NG94) system. The unshaded rows provide some basis for comparing the performance of NG94 on query set 3 against last year's system and against the median performance of all system in last year's TREC evaluation. NG93 was at the top of the bottom quartile of last year's TREC competitors, but has now moved very close to the median performance of those systems. This is very encouraging, especially in view of the fact that we have not had a chance to further tune the system with regards to term weighting or query enhancements. The row labelled NG94/Q4 gives our results for this year's adhoc test.

TABLE 1. Test Results For Disks 1&2

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/Q1	0.1997	0.3934	0.2874
NG94/Q2	0.2172	0.3988	0.3002
NG94/Q3	0.2614	0.4068	0.3236
NG93/Q3	0.1885	0.3426	0.2494
Median93/Q3	0.2804	0.4354	0.3388
NG94/Q4	0.2061	0.2884	0.2668

Table 2 shows similar results for the dataset on Disk 3.

TABLE 2. Test Results For Disk 3

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/Q2	0.1877	0.2938	0.2502
NG93/Q2	0.1415	0.2524	0.2031
NG94/Q3	0.2528	0.3360	0.3101

Again, comparing the 'NG94/Q2' and 'NG93/Q2' rows shows a significant performance improvement from last year's system to this one.

The row labelled 'NG94/Q3' gives our results for this year's routing test.

Table 3 gives our results from the Spanish test.

TABLE 3. Test Results For Spanish Test

Description	Average Precision	Precision @ 100 docs	R-precision
NG94/SP1	0.4275	0.5258	0.4522

6.0 Conclusions And Future Directions

We have successfully implemented an N-gram-based information retrieval system using the vector processing model. This system uses the frequency of N-gram occurrence in queries, documents, and the entire document collection as a whole to drive its processing. This approach has many advantages, including:

- It provides a robust retrieval system that can tolerate spelling errors in both documents and queries.

- It requires no linguistic pre-processing of documents or queries to perform word-stemming or stopword removal. Thus it is also inherently language independent, as indicated by our early results in processing Spanish documents.

- It allows the system to accrue all of the benefits of the vector processing model, including being able to manipulate documents and queries in a uniform way. For example, it easy to use a retrieved document as a query for a more refined search, such as is necessary for relevance feedback systems.

Using the various TREC datasets, query sets, and relevance assessments, we determined that our new system performed substantially better than last year's adhoc N-gram-based system. Indeed, even in its current simple form, it is competitive with over half of the systems fielded last year.

Besides performing better, the new system, unlike its predecessor, has more obvious directions for improvements. For example, we used only the simplest of term weighting schemes that Salton suggested. It would be straightforward to try some of the more elaborate weighting schemes mentioned in [Harman92] and [Salton88]. Also, all of the high-performing systems from last year included some sort of auxiliary processing using thesauri or semantic networks to enhance or enlarge the queries, and improve their retrieval performance. We have not yet done any of those things, but it seems clear that they would help. Furthermore, as we noted in Section 3.2, one of the real limitations we found was that we had to trade off disk space and performance. It would be very worthwhile to investigate other low-level representation techniques that would allow us to keep more quad-grams with lower IDFs to allow a bit more selectivity.

Finally, although this system is far faster than last year's entry, it still is not usable for true interactive use. We will pursue various approaches for significantly improving the system's retrieval times, including the use of intermediate indexes to speed the selection of relevant vectors to evaluate.

Acknowledgments

We would like to thank the Environmental Research Institute of Michigan for its support during this research, and especially for the use of its computing resources.

References

[Cavnar93a] Cavnar, William B., and Vayda, Alan J., "N-Gram-Based Matching For Multi-Field Database Access In Postal Applications," Proceedings of the Second Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 287-297.

[Cavnar93b] Cavnar, William B., "N-Gram-Based Text Filtering for TREC-2," Proceedings of The Second Text REtrieval Conference (TREC-2), D. K. Harman, Editor, National Institute of Standards and Technology, March 1994, pp. 171-179.

[Cavnar94] Cavnar, William B., "N-Gram-Based Text Categorization," Proceedings of the Third Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 161-176.

[Harman92] Harman, Donna, "Ranking Algorithms," in Information Retrieval: Data Structures & Algorithms, edited by William B. Frakes and Ricardo Baeza-Yates, Prentice Hall, 1992, pp. 366-376.

[Harman94] Harman, Donna, "Overview of the Second Text REtrieval Conference (TREC-2)," Proceedings of The Second Text REtrieval Conference (TREC-2), D. K. Harman, Editor, National Institute of Standards and Technology, March 1994, pp. 1-20.

[Salton88] Salton, Gerard, and Buckley, Chris, "Parallel Text Search Methods," CACM, Feb. 1988, pp. 202-215.

[Salton94] Salton, Gerard, and Allan, James, "Text Retrieval Using the Vector Processing Model," Proceedings of the Third Symposium on Document Analysis and Information Retrieval, University of Nevada Las Vegas, 1994, pp. 9-22.

[Thomas] Thomas, Timothy, "Document Retrieval From a Large Dataset of Free Text Descriptions of Physician-Patient Encounters via N-Gram Analysis," Los Alamos National Laboratory.

Appendix I: Quad-Gram Frequency Tables

TABLE 1. Top 20 Quad-Grams By Frequency In Data Disks 1 & 2 (English)

Quad-Gram	Total Occurrences	Number of Documents Containing Quad-Gram	Inverse Document Frequency
_THE	20563024	731778	0.019733
THE_	18302404	730260	0.022729
ING_	8240660	690620	0.103247
TION	7751182	687725	0.109307
AND_	7631370	706559	0.070329
_AND	7179946	703706	0.076166
ION_	7125066	686460	0.111964
ATIO	4532871	619104	0.260957
_FOR	4177190	624514	0.248405
ENT_	3985281	592830	0.323521
_PRO	3534605	551068	0.428909
FOR_	3344583	592207	0.325038
MENT	3307425	530937	0.482598
_THA	3293673	510667	0.538756
_COM	3093855	547052	0.439461
HAT_	2981242	486217	0.609539
TED_	2976304	587264	0.337130
_CON	2963004	566401	0.389315
THAT	2775365	480230	0.627414
ONS_	2239260	495977	0.580866

Appendix I Continued

TABLE 2. Top Quad-Grams By Frequency In Spanish Data

Quad-gram	Total Occurrences	Number of Documents Containing Quad-Gram	Inverse Document Frequency
QUE_	1136311	57071	0.011732
_QUE	1084538	57043	0.012440
_CON	651580	56787	0.018929
LOS_	650327	56509	0.026009
_LOS	567903	56147	0.035281
_EST	471028	55709	0.046580
_PAR	428472	55453	0.053224
_DEL	424458	55696	0.046916
NTE_	409750	55510	0.051742
DEL_	406817	55496	0.052106
_POR	387101	55370	0.055385
ENTE	382908	55007	0.064875
LAS_	370691	53887	0.094553
_COM	353891	53944	0.093027
ADO_	346527	54791	0.070551
_LAS	326221	53274	0.111058
POR_	324827	54548	0.076964
DOS_	319534	53112	0.115452
ARA_	310465	53803	0.096803
PARA	308931	53718	0.099084