An Algorithm for Predicting the Relationship between Lemmas and Corpus Size

Dan-Hee Yang[a)], Pascual Cantos Gómez, and Mansuk Song

Much research on natural language processing (NLP), computational linguistics and lexicography has relied and depended on linguistic corpora. In recent years, many organizations around the world have been constructing their own large corpora to achieve corpus representativeness and/or linguistic comprehensiveness. However, there is no reliable guideline as to how large machine readable corpus resources should be compiled to develop practical NLP software and/or complete dictionaries for humans and computational use. In order to shed some new light on this issue, we shall reveal the flaws of several previous researches aiming to predict corpus size, especially those using pure regression or curve-fitting methods. To overcome these flaws, we shall contrive a new mathematical tool: a piecewise curve-fitting algorithm, and next, suggest how to determine the tolerance error of the algorithm for good prediction, using a specific corpus. Finally, we shall illustrate experimentally that the algorithm presented is valid, accurate and very reliable. We are confident that this study can contribute to solving some inherent problems of corpus linguistics, such as corpus predictability, compiling methodology, corpus representativeness and linguistic comprehensiveness.

Manuscript received January 31, 2000; revised April 3,2000.

[a)] Electronic mail: dhyang@samchok.ac.kr

I. INTRODUCTION

Commercially available corpora run from one million tokens, e.g., the Brown Corpus to hundreds of millions or even billions of tokens, e.g., the Association for Computational Linguistics (ACL), Data Collection Initiative (DCI), the British National Corpus (BNC), etc. It is somehow assumed that 10 million words substantially enable us to describe the information on vocabulary and syntax synthetically and that the size of a corpus is not very important once it has reached a certain number of tokens [1]. We think that this is basically true whenever we intend to compile a concise medium-size paper/compact disk (CD) dictionary or build a probabilistic tagger.

However, this does not hold if we aim to build a good electronic dictionary/lexicon for national language processing (NLP) or an automatic semantic analyzer for machine learning [2], [3]. A practical semantic analyzer should be able to process virtually any sentence, containing, even, words not included in a concise medium-size paper/CD dictionary. Complete electronic dictionaries/lexicons are essential to practical NLP. The problem is compiling such a complete electronic dictionary/lexicon for an automatic semantic analyzer. Some possible approaches are the *CYC Project* or Dan-Hee Yang's corpus-based approach [3], which, in principle, seems to be more practical, efficient, and flexible.

There is a great deal of interesting research on automated acquisition of linguistic information from corpora by means of statistical or machine learning mechanisms [4]–[7]. However, all these works critically suffer from data sparseness, which is a phenomenon that a given corpus fails to provide sufficient information or data on relevant word-to-word relationships. This is an intrinsic problem and a serious drawback for most corpus-based NLP work. This phenomenon is closely related

to corpus size and there is a need to consider how large corpora should be compiled for machine learning and sampling resources in order to develop practical NLP software and/or complete dictionaries for humans and computers.

In order to answer this question, we shall discuss the flaws of previous several researches for predicting corpus size, especially using the pure regression or curve-fitting, which appears to be the best method to capture the lemma-growth pattern of a corpus. To overcome these flaws, we shall contrive a new mathematical tool (a *piecewise curve-fitting algorithm*) by means of statistically investigating the relations between the number of different base forms (lemmas) and the size of a corpus (tokens). Next, we suggest how to determine the tolerance error of the algorithm for positive predictions with specific corpora. And finally, we shall experimentally illustrate the validity and reliability of the algorithm.

II. RELATED WORK

Lauer outlined the basic requirements regarding the necessary size of linguistic corpora for general statistical NLP [8], [9]. He suggested a virtual statistical language learning system by means of establishing an upper boundary on the expected error rate of a group of language learners as a function of the size of training data. However, his approach partially lacks validity as the size of training data that can be extracted from a corpus differs significantly, depending on the type of linguistic knowledge to be learned and/or on the procedure and technique used to extract data from the corpus. De Haan agreed with this issue insisting that the suitability of data seems to depend on the specific study to be undertaken and added that there is no such thing as the best or optimal size [10].

Weischedel demonstrated that a tri-tag model using 47 possible parts-of-speech (POS) would need little more than one million words of training data [7]. However, most researchers might agree that this size seems neither suitable nor sufficient for semantic acquisition, among others. This implies that corpus size is heavily dependent on the linguistic research we want to carry out. Consequently, there is no use trying to predict the size of a corpus that can satisfy all linguistic requirements and expectations.

A more realistic and plausible research line may be found in the field of information retrieval. With reference to the increase rate of different forms, Heaps reported that the following expression is true for general English text of up to at least 20,000 words [11]: The number of different words D is related to the total number of words N by an equation relative to the way the text length increases:

$$D = kN^{\beta} \quad \text{hence,} \quad \log D = \beta \log N + \log k, \tag{1}$$

where k and β are constants that depend on the particular text. He pointed out the linear relation between $\log D$ and $\log N$ by taking common logarithms of both sides of $D = kN^{\beta}$, respectively. The purpose of his research was to create and manage index files efficiently for document retrieval. This explains why he experimented on a collection of title words of documents rather than on general English text. However, he did not give any explicit explanation on how the equation was actually derived.

Note that Heaps just insisted on the fact that (1) is true for general English text of "up to at least 20,000 words" rather than any size. This might imply that the dependent constants or, what might seem worse, the expression itself is likely to change as the corpus size grows dramatically. This suggests that even if we were to find a function that could fit some given data (corpus), there would still be no guarantee that the function would always hold. This is precisely the common major flaw of Young-Mi Jeong's and Sánchez and Cantos' researches within the field of computational linguistics [12], [13]. In Section III-3, we shall delineate experimentally this problem in more detail.

III. INDUCTION OF A PIECEWISE CURVE-FITTING ALGORITHM

We are confident that Heaps, Young-Mi Jeong and Sánchez and Cantos failed by using regression techniques [11]–[13]. This might be attributed to their preference for Zipf's laws and on their belief that these statistical models would not just hold for their moderately large experimental data but also for any data (corpus size). However, recall that most regression models developed in statistics and applied to many engineering and scientific problems in daily life are generally used with small, finite and normally distributed data sets (such as heights, ages, temperature, and the like). However, some linguistic items, such as lemmas, are neither normally distributed, in principle, nor finite.

We propose to clarify the limits of regression or curve-fitting and to induce a new algorithm that might overcome these constraints.

1. Variation among Corpora

Heaps, Young-Mi Jeong, and Sánchez and Cantos experimented with small texts and/or relatively small corpora [11]–[13]. However, as already discussed, empirical corpus-based data depend heavily on the corpora from which this information has been extracted. This means that different linguistic domains (economy, science, etc.), degrees of formality, and media (radio, newspaper, etc.) result in different token-lemma relationships. To illustrate this issue, consider variation among various corpora depicted in Table 1 and Fig. 1.

Table 1. Corpora used for the experiment.

	YSC I	YSC II	YSC III	YSC V	YSC VI
Short Name	STANDARD	DEWEY	1980s	1970s	1960s
Number of Tokens	2,022,291	1,046,833	5,177,744	7,033,594	6,882,884
Sampling Criteria	Reading pattern	Dewey decimal	'80s Texts	'70s Texts	'60s Texts

	YSC VII	YSC VIII	YSC IX	NEWS	TDMS
Short Name	1990s	TEXT	CHILD	NEWS	TDMS
Number of Tokens	7,171,653	674,521	1,153,783	10,556,514	9,060,973
Sampling Criteria	'90s Texts	Textbooks	Children's books	Newspaper	Sampling

The *Yonsei Corpus* (YSC) consists of eight subcorpora (the YSC I ~ YSC IX) compiled according to different sampling criteria. The *Center for Language and Information Development* (CLID) paid special attention to the sampling criteria in order to get the most balanced corpus (see [14]). In Fig. 1, the *x*-axes present the corpus size in words (tokens) and the *y*-axes the number of different lemmas or lexemes. The *TEXT Corpus* consists of Korean textbooks (ranging from elementary school to high school level) written by native Korean-speaking authors. The *CHILD Corpus* was compiled by means of samples taken from children's books (fairy tales, children's fiction, etc.).

The upper graph in Fig. 1 shows how the number of different lemmas in the *TEXT* and *CHILD Corpora* increases at a slower rate than the *Dewey* and *STANDARD Corpora* ones. This seems obvious as both the *Dewey* and the *STANDARD Corpora* contain different linguistic domains and, additionally, the texts refer to adult language, which, in principle, is more varied and lexically and semantically more complex.

Note that the *1960s, 1970s, 1980s and 1990s Corpora* were compiled having chronological criteria in mind: the *Dewey Corpus* by the Dewey decimal classification criteria and the *STANDARD Corpus* by the reading pattern criteria. The *TDMS* (Text and Dictionary Management System) *Corpus* was compiled similarly

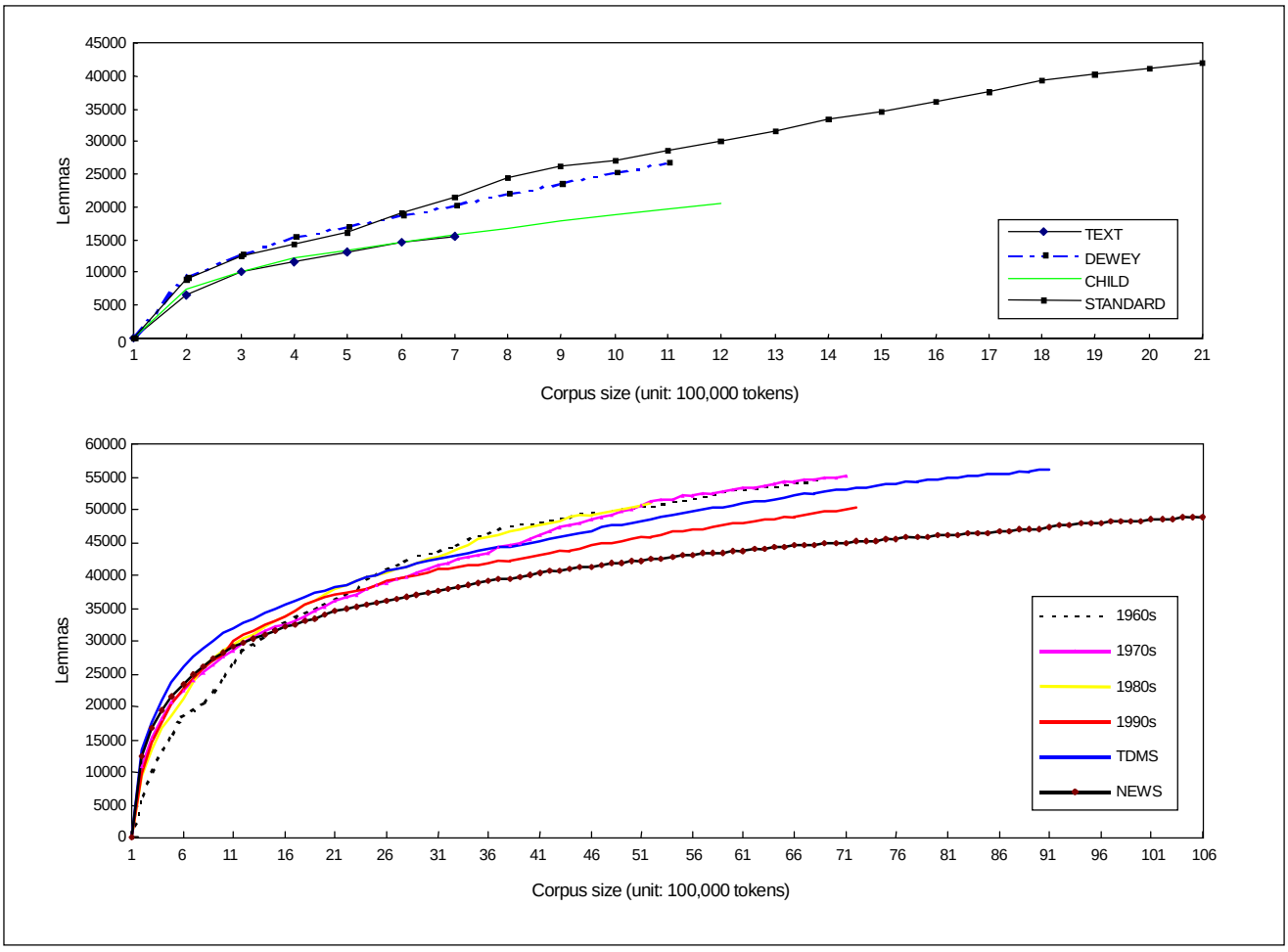

Fig. 1. Lemma-growths in various corpora.

to the *STANDARD Corpus* by the *Korean Advanced Institute of Science and Technology* (KAIST). The *NEWS Corpus* is a CD-ROM title, which consists of all the Chosun-ilbo newspapers from 1993 to 1994.

The lower graph in Fig. 1 shows that the *NEWS Corpus* has a lower lemma-growth compared with its counterparts on the same graph. Interesting is the way that the *1960s Corpus* behaves compared with the *1970s, 1980s and 1990s Corpora*. Notice that the difference between the *1960s* and the *1990s Corpus* is more marked than the one between the *NEWS* and the *1990s Corpus* though the *1960s* and the *1990s Corpus* (not the *NEWS Corpus)* were compiled following similar sampling criteria.

From these observations, it follows that sampling strategies affect lemma-token slope to some degree. However, more important is the fact that all corpora have one common inherent characteristic: *monotone, convex up,* and *increasing* curve.

For our research purpose, we merged all 10 corpora into the *TOTAL Corpus* (totaling 50,780,790 tokens).

2. General Function and Good Curve-fitting

In this section, we shall derive an equation able to describe monotone, convex up and increasing curves that matches the lemma-token growth curve. It is a well-known fact that regression is similar to curve-fitting, though curve-fitting offers more general and sophisticated techniques than regression [15]–[17]. However, regression does not even offer any clue on how to find a general form of a desired function. Consequently, we shall proceed with curve-fitting rather than regression.

Given the data points $\mathrm{p}(x_1, y_1), \mathrm{p}(x_2, y_2), \cdots, \mathrm{p}(x_n, y_n)$, estimating the corpus size would be to find x_k corresponding to $y_k \notin [y_1, y_n]\,(k > n)$, where x is the corpus size, and y is the number of different lemmas. In other words, the problem is to find a function that most closely fits the given data points in order to estimate how the number of different lemmas y_i grows or evolves regarding changes given in a corpus of size x_i. *Curve-fitting* is to find a function $y = g(x)$ that fits a given set of data.

$$g_1(x) = \alpha e^{\beta x},\ g_2(x) = \alpha x^{\beta},\ g_3(x) = \alpha + \beta \ln x,$$
$$g_4(x) = \alpha + \beta / x = g_5(x) = \alpha /(\beta + x),\ g_6(x) = \alpha x /(\beta + x). \quad (2)$$

In curve-fitting, it is vital to define correctly or as accurately as possible a general form of $g(x)$ in order to achieve the "best fitting" of the given experimental data. The functions in (2) above are six commonly used two-parameter general functions whose graph is monotone and convex. Any graph of a $g(x)$ function suitable for fitting monotone and convex data has no turning points and no inflection points for the given interval. Note that two parameters, say α and β, are generally sufficient to achieve good fitting in this case.

Table 2. Linearizing the curve.

Original Function	Linearized Form	Transformation Relations					
$y = g(x)$	$Y = L(X) = a + bX$	$X =$	$Y =$	$a =$	$b =$	$\alpha =$	$\beta =$
$y = \alpha x^{\beta}$	$\ln y = \ln\alpha + \beta \ln x$	$\ln x$	$\ln y$	$\ln\alpha$	β	e^a	b

One is likely to think that the more parameters we have in various general functions, the more the curve fitting improves, e.g., whenever we use any plausible linear combination of the functions in (2). Such a general function is called an *n-parameter linear model.* However, it is also well known that the model is suitable for data having either turning points or inflection points on the fitting interval. In addition, we might also consider polynomial functions. The *polynomial wiggle problem* states that it is impossible to find a correct solution to such monotone and convex data [17].

To evaluate the exactness of how a given guess function $g(x)$ fits given points $p_1, \cdots, p_n$, we will use the least square error criterion, which is to find $g(x)$ minimizing the sum of square errors $E(g) = \sum [g(x_k) - y_k]^2$. If the error for each y_k is normally distributed and $g(x)$ has the correct functional form, then the $g(x)$ obtained by minimizing $E(g)$ will approach actual functional dependence as the number of data points increases. Note that in what follows we shall refer to functions as *general functions* if the values of their parameters are not yet set, or else as *guess functions*. Those functions used for predicting *outside* the fitting interval will be named *predicting functions*.

In order to fit a two-parameter g(x) to monotone and convex data like the corpus slope (see Fig. 1), the linearizing formula of Table 2 can be used [15], [17]. We must now choose the function, among the general functions in (2), that best fits the different lemma-growth slope of a corpus. If we plot the given data points, as in Fig. 1, the lemma-growth gets monotone, convex up and increasing. Therefore, if the two new conditions (a) convex "up" and (b) "increasing" are added to the conditions "monotone and convex", the considered scope is restricted to the four functions given in Table 3.

We evaluated each of the functions according to the following three factors: (a) comparison of the square errors (the second column of Table 3; whenever the entire interval of the *TOTAL Corpus* is fitted only by means of a single function); (b) consideration of the number of subdivisions-I (whenever the *piecewise curve-fitting* method described in Section III-3 is used); and (c) consideration of the number of subdivisions-II (whenever the same

Table 3. Comparison of the degree of fitting the *TOTAL Corpus* with the monotone, convex and increasing general functions.

	Square error	Number of subdivisions-I	Number of subdivisions-II
$g_2(x) = \alpha x^{\beta}$ $(\beta > 0)$	0.631819	2	1
$g_3(x) = \alpha + \beta \ln x$ $(\beta > 0)$	73,375,272.724506	196	42
$g_4(x) = \alpha + \beta / x$ $(\beta < 0)$	15,972,823,083.486553	199	44
$g_6(x) = \alpha x/(\beta + x)$ $(\beta > 0)$	5,043,779,002.584718	196	49

method is used for the interval over 40 million tokens). The consideration of these three factors certainly shows that the general function $g_2(x)$ is superior to all the others.

Therefore, we decided to take the following equation as a general function for this study.

$$g_2(x) = y = \alpha x^{\beta}, \tag{3}$$

where y is the number of different lemmas, x is the corpus size in this study. As already advanced in Section II, Heaps found this general function [11], even though he presented neither any convincing mathematical method to reach it nor any mathematical explanation regarding its adequacy. By the use of a common logarithm instead of a natural one in (1) in Section II, we just infer that he might be using a linear regression method. Notice the mathematical plausibility in modeling the relationship between the different items (lemmas) and the total number of words (tokens) by $g_4(x)$, even if it seems to be the worst function.

We can now draw two new conclusions. First, the lemma-growth slope of a corpus, irrespective of its size, is always monotone, convex up and increasing. This means that $y = \alpha x^{\beta}$ can, irrespective of its size, give a good fit for the corpus growth slope (lemmas) within a reasonable rate of error. Second, for both the entire and partial interval of the slope, the general function $y = \alpha x^{\beta}$ is the best among those presented in Table 3.

3. Limitations of a Guess Function

As already mentioned, even if we were to find a function for any given data, using it for prediction as such cannot always be justified because the dependent constants, or what is worse, the equation itself might change whenever the corpus size grows. To test this, we fitted the entire interval of the *STANDARD* and *1980s Corpus* (Fig. 2 (a)) and the *TOTAL Corpus* (Fig. 2 (b)) by means of a single function. At first sight, the entire interval of the *STANDARD Corpus* appears to fit satisfactorily given that the graphs are drawn on a highly reduced scale and the data is relatively small. However, accuracy of approximation towards the last interval (or tail of the slope) is of prime importance in order to predict the size of a linguistic corpus accurately.

Observing both graphs, it becomes obvious that, though the fitting slope and the actual one flow similarly, the closer we get to the last interval, the more evident the differences become. Thus, the larger a corpus gets, the more the number of different lemmas that we can find is prone to be overestimated. The graph of the *TOTAL Corpus* Fig. 2 (b) reinforces this evidence.

Table 4 interprets the graphs of Fig. 2 numerically. Note that the square errors were calculated by taking logarithms of both x and y according to the linearizing formula of Table 2. The actual number of different lemmas in the *TOTAL Corpus* is 83,498. The question now is how large a corpus would need to be in order to get 83,498 different lemmas. We experimented with the *STANDARD Corpus* and the corresponding guess function revealed that a hypothetical 7,240,000 token corpus would be enough. Further research revealed that the 7,240,000 token corpus actually contains just 54,736 different lemmas. Experimenting on the *1980s Corpus*, we found that a 15,540,000 token corpus would be needed. Notice that even though the size of the *1980s Corpus* is 1.5 times bigger than the *STANDARD* one, the actual increase in lemmas is just 8,108. However, the predicted corpus size for the *1980s Corpus* is twice the one predicted for the *STANDARD* corpus.

Regarding the *TOTAL Corpus*, which is almost ten times larger than the *1980s Corpus*, the guess function predicted a 41,360,000 token corpus, which is 2.7 times more than the predicted size for the *1980s Corpus*. In addition, we found here a 17,428 different lemma increase. The experimental finding from the *TOTAL Corpus* is relatively accurate due to its small error rate: $(83{,}498 - 80{,}272)/83{,}498 \times 100 = 3.86\%$. Notice, however, that the fitting curve near the last interval in Fig. 2 (b) flows linearly and much higher compared to the actual data. From this experimental data, we can conclude that such a guess function is fairly reliable within the given interval $[x_1, x_n]$ of experimental data, but, as already discussed, we feel strongly confident that predicting x outside the given interval should be avoided.

Up to this point, we are now able to advance a number of preliminary conclusions: (a) the guess function determined by either curve-fitting or regression warrants satisfactory accuracy in estimating a value *within* the fitting interval. This implies that it seems reasonable to use a guess function as a predicting function in these cases; (b) these guess functions are not very reliable if we want to predict a value *outside* the fitting interval. Consequently, there are only two cases where we can use such guess functions

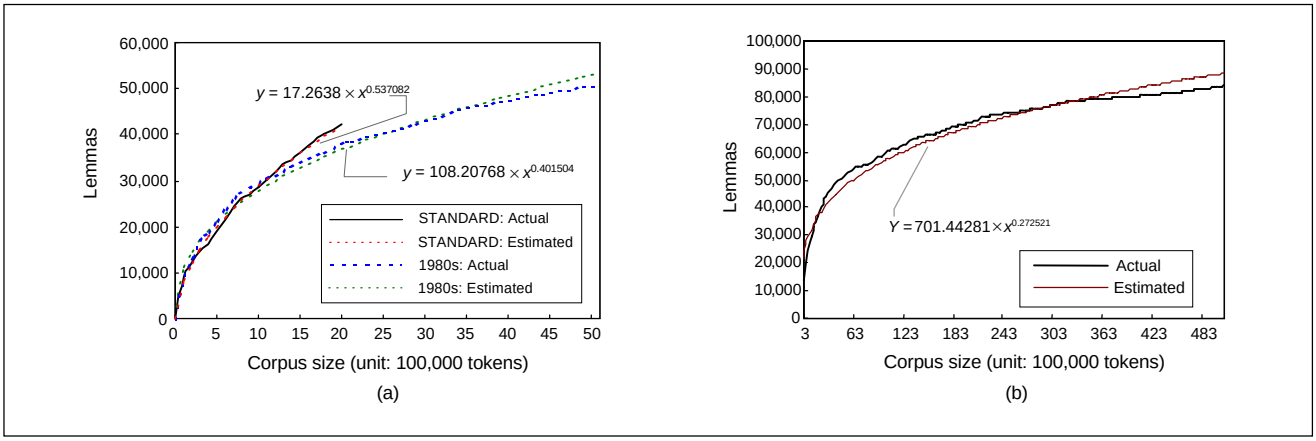

Fig. 2. Actual and Fitting curves: (a) the *STANDARD* and *1980s Corpus* (b) the *TOTAL Corpus*.

Table 4. Fitting curves and their degree of error.

	STANDARD	1980s	TOTAL
Corpus size in tokens	2,022,291	5,177,744	50,780,790
Guess Function	y= 17.26380×$x^{0.537082}$	y=108.20768×$x^{0.401504}$	y=701.44281×$x^{0.272521}$
Square Error	0.0196810	0.0826790	0.6275763
Predicted corpus size to get 83,498 different lemmas	7,240,000	15,540,000	41,360,000
Actual number of different lemmas in the predicted corpus size	54,736	62,844	80,272

as predicting functions: (a) in ideal cases, whenever the experimental data converges to infinity (∞) or contains almost all possible instances; and (b) in practical cases, whenever the experimental data is sufficient, that is, whenever we are confident that there will be no sudden change in the slope even if size increases dramatically.

In what follows, we shall use the term "estimate" strictly whenever we are dealing with the problem found *within* the given experimental interval, and "predict" whenever the problem lies *outside* it.

4. Piecewise Curve-fitting Algorithm

As already discussed, to predict the size of a corpus reliably and accurately for any NLP research, we cannot fit the entire interval by means of just a single function given in (2) in Section III-2. Instead, we have developed the *piecewise curve-fitting algorithm* (see Fig. 3) that overcomes the limitations and drawbacks of the pure curve-fitting method for a given interval $[x_1, x_n]$. The algorithm subdivides the entire interval into several pieces to satisfy certain constraints (this will be described later) and to enhance the accuracy in the curve-fitting.

To get a full understanding of this algorithm, refer to Fig. 4. The thick curve in Fig. 4 (a) shows the actual lemma-growth slope of the *TOTAL Corpus*. If we predict or project the corpus to 3.3 million words or tokens, the guess function becomes $y_0 = 21.963142 \times x^{0.510881}$ (square error: 0.008240). According to the guess function, we might expect to find roughly 190,000 different lemmas in a 50 million token corpus. However, we actually just found roughly 70,000 ones. Notice that Fig. 4 (a) visually represents the limitations and problems of the predicting approaches used by Heaps, Young-Mi Jeong and Sánchez and Cantos [11]–[13].

A solution to this problem is to predict only for the interval [3300000, 50000000], leaving aside the interval [1, 3300000]. The new guess function gives $y_1 = 1770.104205 \times x^{0.218201}$ (square error: 0.025799). As Fig. 4 (a) reveals, the new fitting curve seems to be precise within the new given interval, even though it does not fit accurately for the first interval [1, 3300000]. Thus, the algorithm fits the actual curve very closely even though size increases. As already mentioned (see Fig. 2 in Section III-3), the accuracy of approximation towards the last interval is of prime importance in predicting the size (or growth

Initialization Step

Set i = 1, j = n.

Iteration Step

Do until the square error on the given interval [i, j] is less than the given tolerance error:

(a) Find the best position *t* where the sum of tolerance errors on [i, *t*] and [*t*, j] is minimal.

(b) Subdivide the interval [i, j] into [i, *t*] and [*t*, j] by *t*.

(c) Perform the iteration step for each of the new given intervals [i, *t*] and [*t*, j].

Final Step

Return the fitting function for the last right subinterval as a predicting function.

Fig. 3. A piecewise curve-fitting algorithm.

pattern) of a corpus. Hence, the procedure in Fig. 4 (b) is worth considering. Notice that the position for subdivision, $x_t = 3.3$ million, is calculated in the iteration step (a) of piecewise curve fitting algorithm in Fig. 3.

The square error, 0.025799, of y_1 in Fig 4 (a) is still greater than the given tolerance error 0.01 (see next section for this value). Therefore, it seems wise to subdivide the interval again. The right position for subdivision is at $x_t = 33$ million tokens by the iteration step (a) as seen in Fig. 4 (b). Here, the front interval [3300000, 33000000] fits with $y_0 = 1357.905526 \times x^{0.234522}$ (square error: 0.003520), and the rear interval [33000000, 50000000] with $y_1 = 6251.224965 \times x^{0.145810}$ (square error: 0.000738). At this point, notice that the square error on each subinterval is less than the given tolerance error.

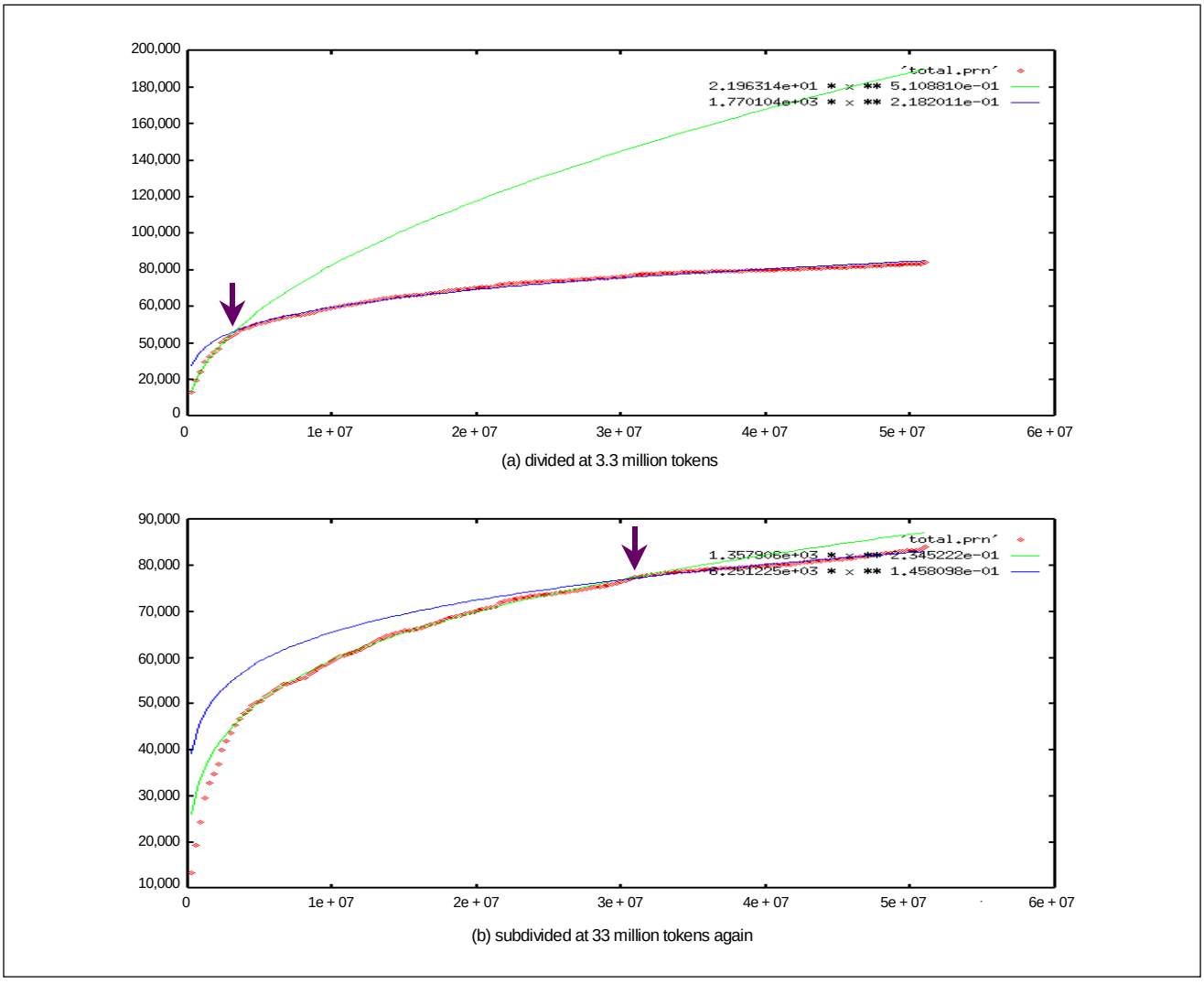

Fig. 4. Piecewise curve-fitting of the *TOTAL Corpus*.

Table 5. Corpus size by tolerance errors.

Tolerance error	0.1	0.01	0.001
Predicting function	$y = 906.14798 \times x^{0.239512}$	$y = 2141.05529 \times x^{0.190054}$	$y = 3126.23437 \times x^{0.168510}$
Number of subdivisions	2	3	7
Error of the predicting function	0.024365	0.002643	0.000834
| Predicted – Actual |	983	124	353

We shall use guess functions like y_i in Fig. 4 (b) to predict corpus sizes. The reason for this decision is based, as previously mentioned, on the assumption that there are only two cases that justify the use of an original guess function, e.g., those of Fig. 2, as a predicting function. The first one is rarely expected to be accomplished in reality, particularly with linguistic corpora. If this is so, we have no choice but to satisfy the second constraint.

In Fig. 4 (a), the *TOTAL Corpus* seems to satisfy the second constraint due to the sudden change at $x_t = 3.3$ million tokens and some change at $x_t = 33$ million tokens. At least, the quantity of the next sudden change will be less than $|y_0(x_t) - y_1(x_t)|$, where x_t is a dividing point (see Fig. 4 (b)). Consequently, we shall set up a new practical constraint: the square error on each subinterval should be less than the reasonable tolerance error. In particular, the last interval should fit closely without falling into a local solution. What we understand by a local solution is that each fitting curve fits its corresponding subinterval well, including the last interval. However, the accuracy of estimation is low (the extreme subinterval would be just 1). According to these new constraints, the last y_1 deserves to be the predicting function for the purpose of our research.

5. How to Determine the Tolerance Error

In what way can we determine the value of tolerance error for good prediction? Recall that the tolerance error aims to be smaller, the more the entire interval is subdivided into. Thus, the more we subdivide the entire interval into smaller subintervals, the better we can fit the subinterval. However, the more subintervals, the more possibilities there are of falling into a local solution. There is no mathematical algorithm to overcome such phenomena. We suggest a reasonable value derived from our experimental data.

We shall be reducing the tolerance error by 1/10 in order to find the proper number of subdivisions , i.e., acts of dividing and shall then evaluate each predicting function by means of the following factors: (a) the number of subdivisions; (b) the least square errors of the predicting functions; (c) the absolute difference between the predicted corpus size and the actual one. The most desirable predicting function will be the one where both the absolute difference and the number of subdivisions are kept small.

The procedure to find a reasonable tolerance error rate consists in finding predicting functions for the 45 million token curve for nouns (see Fig. 5 (a)) by means of our piecewise curve-fitting algorithm (see Fig. 3). Next, we predict the number of different nouns by means of the resulting predicting function. Finally, we compare the predicted number with the actual number of different nouns occurring in the *TOTAL Corpus*. (Note that the *TOTAL Corpus* contains actually 50.7 million tokens and 62,497 different nouns). Table 5 shows that the difference is minimal at a tolerance error of 0.01. Furthermore, since the entire interval is subdivided into just four parts, the possibility that it falls into a local solution is much lower compared to the tolerance error of 0.001, which has been divided into eight parts. Therefore, we can conclude that a tolerance error of 0.01 is reasonable for the *TOTAL Corpus*.

6. Experimental Result

We are now in the position to predict the size of linguistic corpora by our piecewise curve-fitting algorithm (see Fig. 3) with a tolerance error of 0.01. This prediction is performed on lemmas of the four major POS: nouns, verbs, adjectives and adverbs. The reason for this restriction is justified on the evidence that these four POS are lexical items, in contrast to functional or grammatical items (e.g. prepositions, conjunctions, etc.) and because they account for around 98% of all lemmas found in standard Korean dictionaries. Clearly, lexical items are the main goal for automatic lexical acquisition in NLP and lexicography, in general.

In addition, there are also other motivations why finding predicting functions by POS is important and necessary. These have something to do with (a) automatic lexical acquisition—trying to acquire a certain number of items (say, 20,000) regardless of its POS might be misleading as most or many items might belong to a particular POS (i.e., nouns)—and (b) specific research interests—we might be just interested in specific POS only, e.g. acquiring just 20,000 nouns, 3,000 verbs, 1,000 adjectives, 500 adverbs, etc..

Table 6. Guess functions of POS by subintervals (where *I* is a subinterval (unit: 100,000 tokens), *G* is the guess function, and shaded fields refer to the predicting functions).

N	I	[0,36]	[36,216]	[216,510]			
	G	$y = 14.31002 \times x^{0.514739}$	$y = 691.38416 \times x^{0.25613}$	$y = 2141.0553 \times x^{0.190054}$			
V	I	[0,15]	[15,30]	[30,282]	[282,510]		
	G	$y = 2.39063 \times x^{0.546146}$	$y = 44.26743 \times x^{0.342274}$	$y = 374.1718 \times x^{0.200222}$	$y = 3206.4913 \times x^{0.074925}$		
ADJ	I	[0,9]	[9,12]	[12,32]	[32,93]	[93,318]	[318,510]
	G	$y = 1.12434 \times x^{0.523399}$	$y = 0.000297 \times x^{1.126044}$	$y = 7.5463 \times x^{0.191276}$	$y = 167.90725 \times x^{0.191276}$	$y = 266.29583 \times x^{0.164006}$	$y = 1232.1248 \times x^{0.075192}$
ADV	I	[0,6]	[6,15]	[15,21]	[21,87]	[87,288]	[288,510]
	G	$y = 5.55857 \times x^{0.362271}$	$Y = 0.00125 \times x^{0.356916}$	$y = 10.44507 \times x^{0.356916}$	$y = 127.85208 \times x^{0.184552}$	$y = 238.40436 \times x^{0.14757}$	$y = 1554.3040 \times x^{0.039315}$

Table 6 gives the guess functions for the lemmas by POS according to the *TOTAL Corpus*. So, for example, if we want to guess the number of different noun lemmas for a corpus ranging from 0 to 3.6 million in size, we can calculate it using guess function $y = 14.31002 \times x^{0.514739}$, where y results into the number of different nouns and x stands for the corpus tokens. Similarly, guess function $y = 44.26743 \times x^{0.342274}$ is effective to guess the number of different verbs for a corpus ranging from 1.5 million to 3.0 million in size. Reversely, it follows that we can guess the required corpus size to get a specific number of different lemmas from the guess functions.

Young-Chae Kim manually counted a corpus consisting of 69 elementary textbooks (totaling 385,291 tokens) [18]. The number of different noun lemmas in his corpus is 12,029. If we apply the corresponding guess function $y = 14.31002 \times x^{0.514739}$ for the subinterval [0, 3600000] (since the total size of the elementary textbooks is within the subinterval [0, 3600000]), we get just 10,736 different nouns. This might be misleading for the reader. However, we need to consider that the guess function used has been derived from the *TOTAL Corpus* and applied to a different corpus [18].

However, we can try to adjust the calculations by considering homographs. Dan-Hee Yang proposed a formula to calculate the maximum number of different nouns by

$$M = N \times (\mathrm{HD} / \mathrm{HI}) \tag{4}$$

where M is the maximum number of different lemmas (nouns), N the observed number of different lemmas (nouns), *HD* the number of lemmas (nouns) in a dictionary, and *HI* the total number of homograph-independent lemmas (nouns) in the same dictionary [19]. The maximum number of different nouns is $10{,}736 \times (305{,}030 / 249{,}748) = 13{,}113$. Note that the distributions $\mathrm{HD} = 305{,}030$ and $\mathrm{HI} = 249{,}748$ are obtained from *Ulimal Keun Dictionary* 'Korean Grand Dictionary'. This reveals that the real amount of different nouns 12,029 is between the guessed value 10,736 and the maximum one 13,113.

Figure 5 compares the actual curves of the different occurring lemmas and their corresponding predicted ones regarding the four major POS relative to the *TOTAL Corpus*. A closer look at Fig. 5 reveals that our predicting functions fit reasonably well, particularly the right half of the total interval, which is long enough to demonstrate the accuracy of our predicting functions.

In order to calculate the required corpus size containing a specific number of different lemmas, we need to transform the predicting functions given in Table 7, taking logarithms of both sides of each predicting function. For example, for nouns we have

$$\begin{aligned} & y = 2141.05529 \times x^{0.190054} \\ \Leftrightarrow\ & \log y = \log 2141.05529 + 0.190054 \log x \end{aligned}$$

where the predicted corpus size x results from

$$\begin{aligned} & 0.190054 \log x = (\log y - \log 2141.05529) \\ \Leftrightarrow\ & \log x = (\log y - \log 2141.05529) / 0.190054, \end{aligned}$$

giving the following transformed function

$$x = 10^{(\log y - \log 2141.05529) / 0.190054}.$$

Other transformed functions of Table 7 are induced in the same way. We can use these transformed functions to get the corresponding corpus size x regarding a required number of lemmas y. From Table 7, for example, to obtain 100,000 different nouns, its transformed function $x = 10^{(\log y - \log 2141.05529) / 0.190054}$ reveals that a corpus of 608 million tokens would be required (by solving $x = 10^{(\log 100{,}000 - \log 2141.05529) / 0.190054}$), and a 23 billion one to get 200,000 different nouns. To get 15,000 different verbs, $x = 10^{(\log y - \log 3206.49129) / 0.074925}$ reveals that roughly 877 million tokens would be needed, and about 41 billion tokens for 20,000 different verbs.

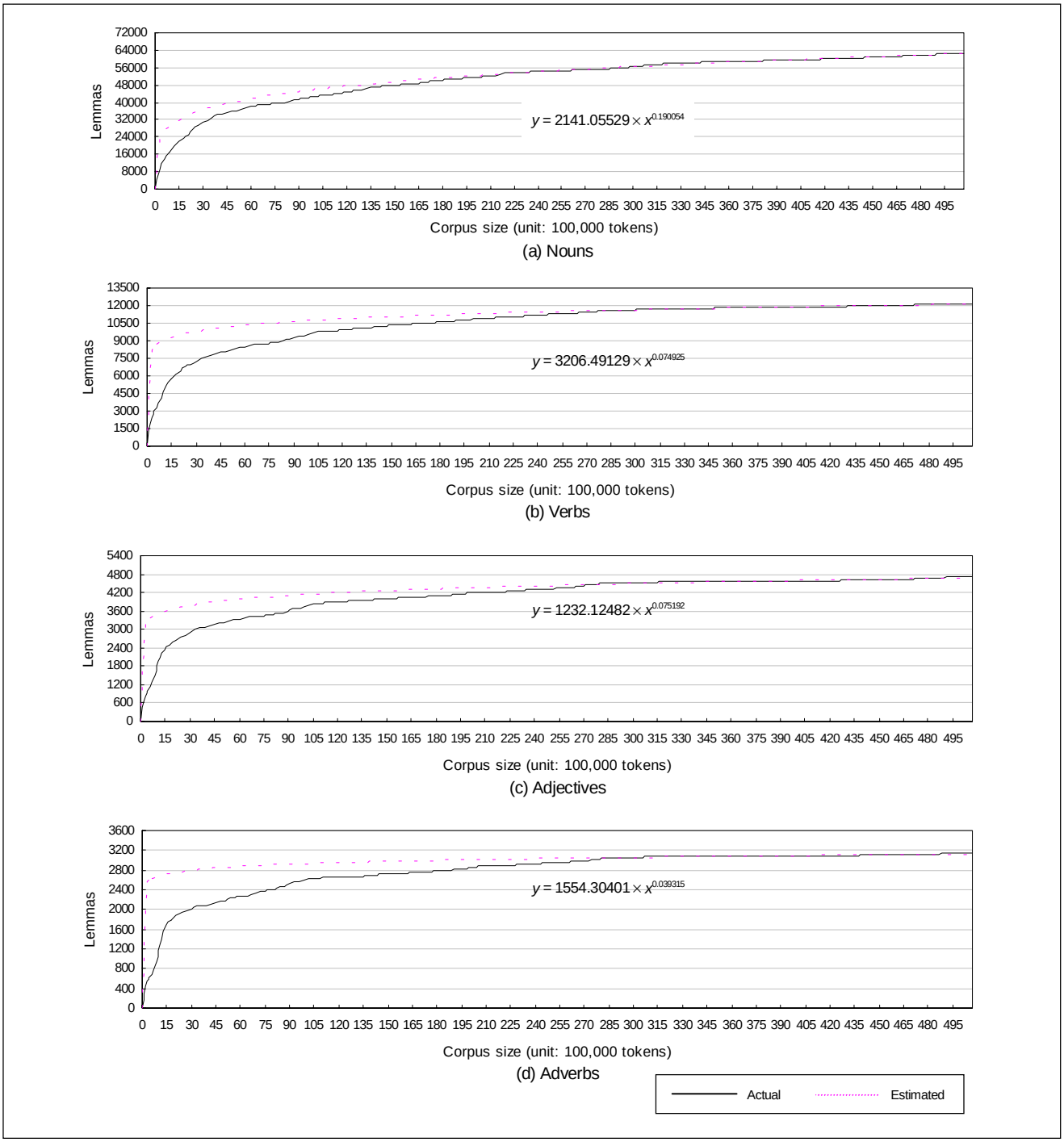

Fig. 5. Actual lemma-growth curves and fitting ones for nouns, verbs, adjectives and adverbs.

The accuracy of the predicting functions is very high. In the case of nouns, for instance, the difference between the actual number of different lemmas A and the predicted ones P is just 124, thus virtually 0; the error rate is 0.002643. Note that the actual number of different lemmas by POS is 62,497 for nouns, 12,188 for verbs, 4,720 for adjectives, and 3,138 for adverbs in the *TOTAL Corpus*. We infer from these experiments and evaluations that our predicting functions are valid, accurate and very reliable.

IV. CONCLUSION

In this paper, we have contrived a new mathematical tool for predicting the relationship between linguistic items (lemmas) and corpus size (tokens), overcoming the major flaws of several pre-

Table 7. Predicting functions and their transformed functions by POS (where A is the actual number of different lemmas and P the predicted number of different lemmas).

	Predicting functions	Transformed functions	Error rate	$\lvert A-P\rvert$
Nouns	$y = 2141.05529 \times x^{0.190054}$	$x = 10^{(\log y - \log 2141.05529)\,/\,0.190054}$	0.002643	124
Verbs	$y = 3206.49129 \times x^{0.074925}$	$x = 10^{(\log y - \log 3206.49129)\,/\,0.074925}$	0.000574	72
Adjectives	$y = 1232.12482 \times x^{0.075192}$	$x = 10^{(\log y - \log 1232.12482)\,/\,0.075192}$	0.001228	42
Adverbs	$y = 1554.30401 \times x^{0.039315}$	$x = 10^{(\log y - \log 1554.30401)\,/\,0.039315}$	0.000431	15

vious researches on this issue.

It seems trivial that the larger the experimental corpus used for predicting the required size of a corpus, the more accurate the predicting result. The importance of predicting corpus sizes becomes particularly relevant whenever we want to know accurately the required size of a hypothetical corpus needed for any specific purpose. In addition, this also allows us to estimate the compilation cost: materials needed, the time required, people involved, etc..

The piecewise curve-fitting algorithm described here in this paper has various practical uses: (a) to determine precisely the size of a corpus needed, depending on the required linguistic items (lemmas, types, tokens) and the type of research to be undertaken; (b) to choose the best compiling strategies. For example, if we find that an impractically large corpus (e.g. trillion tokens) might be required, we might have to change the current method of compiling that corpus. On the other hand, if a smaller corpus (e.g. billion tokens) is needed, we might put more emphasis on balance and coverage; (c) to induce the relationship between lemmas and tokens, types and tokens, and lemmas and types. The usefulness of these dependencies becomes evident in many ways, in particular whenever we want to contrast regularity patterns regarding specific linguistic areas (e.g., written versus spoken, Spanish versus Korean, newspapers versus fiction, see [13], [20]; and finally (d) to represent any monotone, convex up, increasing curve that cannot be represented by single functions. That is, whenever the error term is too big to be represented by a single function.

One critical problem is, however, how to determine the value of tolerance error for positive predictions. In this paper, we have suggested a reasonable method, though it is not necessarily the best, consisting of reducing the tolerance error to 1/10 in order to find the proper number of subdivisions and evaluating each predicting function according to the three factors outlined: number of subdivisions, least square error and difference between predicted and actual corpus size. Further research is required to find a better procedure or formulation, i.e., the relation between tolerance errors and corpus sizes, in this respect.

We are confident that our predicting functions are valid, accurate and very reliable, at least comparatively speaking. In addition, we do also hope that this study will shed new light on issues such as corpus predictability, compiling methodology, corpus representativeness and linguistic comprehensiveness. Our piecewise curve-fitting algorithm is perfectly applicable and valid for any other language, though for practical reasons we have focused and experimented only on Korean.

ACKNOWLEDGMENTS

This research was funded by the Ministry of Information and Communication of Korea under contract 98-86. The authors are most grateful to Prof. Sang-Sup Lee, Prof. Aquilino Sánchez and Dr. Tony Berber-Sardinha for their valuable comments on the the earlier drafts of this paper.

REFERENCES

[1] Sang-Sup Lee, "Corpus: the Concept and Implementation," *Lexicographic Study*, Tap Press, Seoul, Vol. 5, 6, 1995, pp. 7–28.

[2] Dan-Hee Yang, and Mansuk Song, "Machine Learning and Corpus Building of the Korean Language," In *Proceedings of '98 Spring Conference of the Korea Information Science Society (KISS)*, Seoul, 1998, pp. 408–410.

[3] Dan-Hee Yang, and Mansuk Song, "Representation and Acquisition of the Word Meaning for Picking out Thematic Roles," In *International Journal of Computer Processing of Oriental Languages (CPOL),* the Oriental Languages Computer Society1999a), Vol. 12, No. 2, 1999, pp. 161–177.

[4] Kenneth W. Church and Robert L. Mercer, "Introduction to the Special Issue on Computational Linguistics Using Large Corpora," *Using Large Corpora*, edited by Susan Armstrong. The MIT Press, 1994, pp. 1–24.

[5] Philip Resnik, *Selection and Information: A Class-Based Approach to Lexical Relationships*, Ph.D. Dissertation of Department of Computer and Information Science. Pennsylvania University, 1993, pp. 6–33.

[6] Dan-Hee Yang and Mansuk Song, "How Much Training Data Is Required to Remove Data Sparseness in Statistical language Learning?," In *Proceedings of the First Workshop on Text, Speech, Dialogue (TSD'98),* Bruno, 1998, pp. 141–146.

[7] Ralph Weischedel *et al*. "Coping with Ambiguity and Unknown Words through Probabilistic Models," *Using Large Corpora*, edited by Susan Armstrong, The MIT Press, 1994, pp. 323–326.

[8] Mark Lauer, "Conserving Fuel in Statistical Language Learning: Predicting Data Requirements," *the 8th Australian Joint Conference on Artificial Intelligence*, Canberra, 1995.

[9] Mark Lauer, "How much is enough?: Data requirements for statistical NLP," cmp-lg/9509001. In *2th Conference of the Pacific Association for Computational Linguistics*. Brisbane, Australia, 1995.

[10] Pieter De Haan, "The Optimum Corpus Sample Size?," In Leitner, Gerhard (eds.): *New Directions in English Language Corpora, Methodology Results, Software Development*, Mouton de Gruyte, New York, 1992, pp. 3–19.

[11] H. S. Heaps, *Information Retrieval: Computational and Theoretical Aspects*, Academic Press, New York, 1978, pp. 206–208.

[12] Young-Mi Jeong "Statistical Characteristics of Korean Vocabulary and Its Application," *Lexicographic Study*, Tap Press, Seoul, Vol. 5, 6, 1995, pp. 134–163.

[13] Aquilino Sánchez and Pascual Cantos, "Predictability of Word Forms (Types) and Lemmas in Linguistic Corpora, A Case Study Based on the Analysis of the CUMBRE Corpus: An 8-Million-Word Corpus of Contemporary Spanish," *In International Journal of Corpus Linguistics* Vol. 2, No. 2, 1997, pp. 259–280.

[14] Chan-Sup Jeong, Sang-Sup Lee and Ki-Sim Nam, *et al*. "Selection Criteria of Sampling for Frequency Survey in Korean Words." *Lexicographic Study,* Tap Press, Seoul, Vol. 3, 1990, pp. 7–69.

[15] Richard L. Burden and J. Douglas Faires, *Numerical Analysis*, Brooks/Cole Publishing, California, 1997, pp. 473–483.

[16] William Hays, *Statistics*, Harcourt Brace College Publishers, Florida, 1994, pp. 28–30.

[17] M. J. Maron, *Numerical Analysis: A Practical Approach*, Macmillan Publishing, New York, 1987, pp. 201–248.

[18] Young-Chae Kim, "Frequency Survey of Korean Vocabulary," In *Journal of Korean Psychological Association,* Vol. 5, No. 3, 1986, pp. 217–285.

[19] Dan-Hee Yang, Su-Jong Lim and Mansuk Song, "The Estimate of the Corpus Size for Solving Data Sparseness," In *Journal of KISS,* Vol. 26, No. 4, 1999, pp. 568–583.

[20] Aquilino Sánchez and Pascual Cantos, "El ritmo incremental de palabras nuevas en los repertorios de textos. Estudio experimental y comparativo basado en dos corpus lingüísticos equivalentes de cuatro millones de palabras, de las lenguas inglesa y española y en cinco autores de ambas lenguas," *Atlantis (Revista de la Asociación Española de Estudios Anglo-Norteamericanos),* Vol. 19, No. 1, 1998, pp. 205–223.

Dan-Hee Yang received his B.S. and M.S. degrees in computer science from Yonsei University, Seoul, Korea, in 1989 and 1991, respectively. From 1991 to 1996, he worked as a chief researcher at the Center of Software R&D in Hyundai Electronics Inc. He received his Ph.D. degree in computer science from Yonsei University in 1999. During 1994 to 1999, he was a part-time lecturer and/or researcher at Yonsei University and/or Asia United Theological University. Since 1999, he has been a professor of the Department of Computer Engineering at Samchok National University. Since 2000, he has been an editor of the Editorial Committee of the Korean Society for Internet Information and an honorary researcher of the Center for Language and Information Development at Yonsei University. His research interests include natural language processing, computational linguistics, information processing on internet, artificial intelligence, computer assisted language learning, multimedia.

Pascual Cantos Gómez is a Senior Lecturer at the University of Murcia, Spain. He studied English Language and Literature at the University of Murcia as an undergraduate, where he obtained his B.A. and Ph.D. on CALL (Computer Assisted Language Learning). He spent four years lecturing and doing research at the University of Murcia before taking an MA in Computational Linguistics at Essex University, UK. His main research interests are in corpus linguistics, lexical computation and CALL. He has been involved in a number of corpus linguistics and CALL research projects, and is a member of the LACELL (Lingüística Aplicada Computacional, Enseñanza de Lenguas y Lexicografía) Research Group at the University of Murcia. He has published various articles and books on corpus linguistics, lexical computation and CALL.

Mansuk Song received his B.S. degree in Mathematics from Hannam University, Korea, in 1963, M.S. degree in Mathematics from Yonsei University, Korea, in 1968, M.S. degree in Mathematics from the University of Wisconsin, US, in 1972, and Ph.D. degree in Numerical Analysis from the University of Michigan, US, in 1978. From 1978 to 1981, he was a senior researcher in the Agency for Defense Development. Since 1981, he has been a professor in the Department of Computer Science at Yonsei University. During 1988 to 1989, he was a visiting scholar at the University of Wisconsin. His research interests include natural language processing and numerical analysis.