A STORAGE REDUCTION METHOD FOR CORPUS-BASED LANGUAGE MODELS

Hsin-Hsi Chen and Yue-Shi Lee

Department of Computer Science and Information Engineering

National Taiwan University

Taipei, Taiwan, R.O.C.

Abstract

There are many progresses in corpus-based language models recently. However, the storage issue is still one of the major problems in practical applications. This is because the size of the training tables is in direct proportion to the parameters of the language models and the number of the parameters is in direct proportion to the power of these language models. In this paper, we will propose a storage reduction method to solve the problem that results from the large training tables. We use mathematical functions to simulate the distribution of the frequency value of the pairs in the training tables. For the good approximation, the pairs are grouping by their frequency. The experimental results show that although there is a little error rate introduced by the curve function, this scheme is still satisfactory because it performs the closed performance and no extra storage is required in pure curve-fitting model. Besides, we also propose a neural network approach to deal with the pairs classification which is a problem for all class-based approaches. The experimental results show that the neural network approach is suitable to deal with this problem in our storage reduction method.

1. Introduction

There are many progresses in corpus-based language models recently. However, the storage issue is still one of the major problems. The conventional corpus-based language

models record the statistical information extracted from the corpora in the training tables. The size of the training tables is in direct proportion to the parameters of the language models and the number of the parameters is in direct proportion to the power of these language models. Thus, the size of disk space becomes one of the major factors to limit the power of the language models. For example, assume the vocabulary size is about 10^5. If we want to reduce the error rates of applications with word bigram Markov language model, then the possible way to achieve this goal is to enlarge the window size. However, it is difficult to do that from word bigram (the number of parameters is about 10^{10}) to trigram (the number of parameters is about 10^{15}), or more high degree Markov language models in practice because of the tremendous number of parameters.

To overcome this difficulty, class-based approaches and neural network approaches are proposed in recent years. The basic concept of class-based approaches is to use classes instead of words. Because the number of classes is less than the number of words, the class-based approaches will need less disk space than the word-based approaches. In order to group or classify the words into classes, some criteria such as lexical, syntactic and semantic relationships between words are presented.

If two words appear in the same or closed context, then these words have some lexical relationship and belong to the same class. Jelinek, et al. [1] used a very large number of classes on the order of the vocabulary size, and set up an adaptive language model to incorporate unknown words to suitable classes on the basis of the lexical relationships. This method does not touch on how to determine the initial vocabularies, i.e., initial classes, and it takes much time to find the synonym classes for the unknown words. Martelli [2] and Brown, et al. [3] proposed equivalent criteria and co-occurrence relationships respectively to assign words into classes. These methods are able to extract some classes that have the flavor of either syntactically based groupings or semantically

based groupings, depending on the nature of the underlying statistics. However, their experiments are still very small and do not have satisfactory results.

Classes that correspond to the grammatical part-of-speech (or semantic tag) are called syntactic (or semantic) relationship classes. That is, if there are two words in the same class, then these words will have the same part-of-speech (or semantic tag). There are some applications [4-6] using these approaches to reduce the number of parameters. However, these approaches have some drawbacks shown below:

(1) The original performance of the word-based language models may be decreased very much. These may also limit the range of the applications.

(2) These methods need the syntactic (or the semantic) corpus tagged by human.

(3) Because the practice system using these approaches has to do automatic syntactic (or semantic) tagging, it will introduce some extra error results.

Nakamura, et al. [7] proposed a neural network approach, i.e., NET-gram, to overcome the large parameters problem. Training results show that the performance of the NET-gram is comparable to that of the statistical model. However, it still has the problems of the limited network size and the longer training time.

Generally speaking, the goal of these two approaches is to reduce the number of parameters in order to reduce the usage of storage. But these two approaches have suffered from some problems respectively. This paper will propose a storage reduction method to solve the problem that results from the large training table. This method should satisfy the following four conditions:

(1) The original performance of the word-based language models can not be decreased too much by applying this storage reduction method.

(2) This method need not be interfered by human.

(3) The range of the applications will not be limited by applying this method.

(4) The processing speed of the language models must be faster than the original word-based language models by applying this method.

2. The Storage Reduction Method

Our basic idea in the storage reduction is to use a mathematical function to simulate the distribution of the frequency value of the pairs in the training table. This idea comes from theoretical models of natural distributions [8]. If there is a function F that can simulate the distribution of the frequency value of the Markov word bigram pairs, then we can use this function F to evaluate the approximate frequency value for all bigram pairs, that is, $F(word_1, word_2)$ = frequency value. Similarly, given a function G that simulates the distribution of the frequency value of the Markov word trigram pairs, the approximate value for all trigram pairs can be computed by the function $G(word_1, word_2, word_3)$.

If such a function can be found, then this function can be used instead of training table. However, two major problems will be introduced and should be considered.

(1) The distribution of the frequency value of the pairs is usually very random. It is difficult to find a function that is very closed to the distribution of the value of the pairs.

(2) The dimension of the pairs is too high. Assume that the vocabulary size is V and Markov word bigram language model is used. The dimension of the pairs is about V^2. The computation time to find this function is very long because of the large dimension.

2.1 Grouping the Pairs

To overcome the difficulties, grouping the pairs is needed. The grouping criterion is that the pairs are grouped to the same class if the frequency values of the pairs are the same.

We use BDC segmented corpus as our training corpus and word association language models as our language model. BDC corpus includes 7010 sentences about 50000 words. With word association model, 156080 different pairs are generated from this corpus. Tables 1 and 2 show two different groupings. Basically, the grouping result of the 100 classes is extended from that of the 76 classes. Its purpose is: we try to disperse the risk caused by the mathematical function. Assume ten pairs are grouped in the same class C. If the curve (mathematical function) F returns the wrong frequency for the class C, then these ten pairs will have the wrong frequency under this curve. Thus, we extend some of the last classes in the first grouping and generate the second grouping.

Table 1. The 76-Class Grouping

Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs
1	1	137644	27	27	9	53	58	1
2	2	12206	28	28	6	54	62	1
3	3	2840	29	29	8	55	64	1
4	4	1220	30	30	5	56	67	1
5	5	603	31	31	7	57	69	1
6	6	398	32	32	4	58	71	1
7	7	236	33	33	3	59	73	1
8	8	168	34	34	6	60	75	1
9	9	136	35	35	11	61	77	2
10	10	97	36	36	2	62	79	1
11	11	70	37	37	5	63	82	1

Table 1. The 76-Class Grouping (*continued*)

Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs
12	12	56	38	39	4	64	84	1
13	13	44	39	40	5	65	87	1
14	14	37	40	41	2	66	93	1
15	15	27	41	42	3	67	94	1
16	16	30	42	43	3	68	95	2
17	17	25	43	44	2	69	97	1
18	18	22	44	45	2	70	108	2
19	19	18	45	46	1	71	127	1
20	20	13	46	47	1	72	130	1
21	21	13	47	48	1	73	135	1
22	22	14	48	49	3	74	220	1
23	23	12	49	52	2	75	237	1
24	24	15	50	53	1	76	280	1
25	25	6	51	54	1			
26	26	5	52	55	1			

Table 2. The 100-Class Grouping

Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs
1	1	137644	35	35	11	69	52	1
2	2	12206	36	36	2	70	52	1
3	3	2840	37	37	1	71	53	1
4	4	1220	38	37	1	72	54	1
5	5	603	39	37	1	73	55	1
6	6	398	40	37	1	74	58	1
7	7	236	41	37	1	75	62	1
8	8	168	42	39	1	76	64	1
9	9	136	43	39	1	77	67	1
10	10	97	44	39	1	78	69	1
11	11	70	45	39	1	79	71	1
12	12	56	46	40	1	80	73	1
13	13	44	47	40	1	81	75	1
14	14	37	48	40	1	82	77	1

Table 2. The 100-Class Grouping (*continued*)

Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs	Class	Frequency	# Of Pairs
15	15	27	49	40	1	83	77	1
16	16	30	50	40	1	84	79	1
17	17	25	51	41	1	85	82	1
18	18	22	52	41	1	86	84	1
19	19	18	53	42	1	87	87	1
20	20	13	54	42	1	88	93	1
21	21	13	55	42	1	89	94	1
22	22	14	56	43	1	90	95	1
23	23	12	57	43	1	91	95	1
24	24	15	58	43	1	92	97	1
25	25	6	59	44	1	93	108	1
26	26	5	60	44	1	94	108	1
27	27	9	61	45	1	95	127	1
28	28	6	62	45	1	96	130	1
29	29	8	63	46	1	97	135	1
30	30	5	64	47	1	98	220	1
31	31	7	65	48	1	99	237	1
32	32	4	66	49	1	100	280	1
33	33	3	67	49	1			
34	34	6	68	49	1			

2.2 Curve Fitting

At this step, a tool Cricket Graph 1.3.2 in Macintosh is used to find a suitable mathematical function. Some possible curve fitting results for the 76-class grouping are shown in Figure 1 - Figure 7.

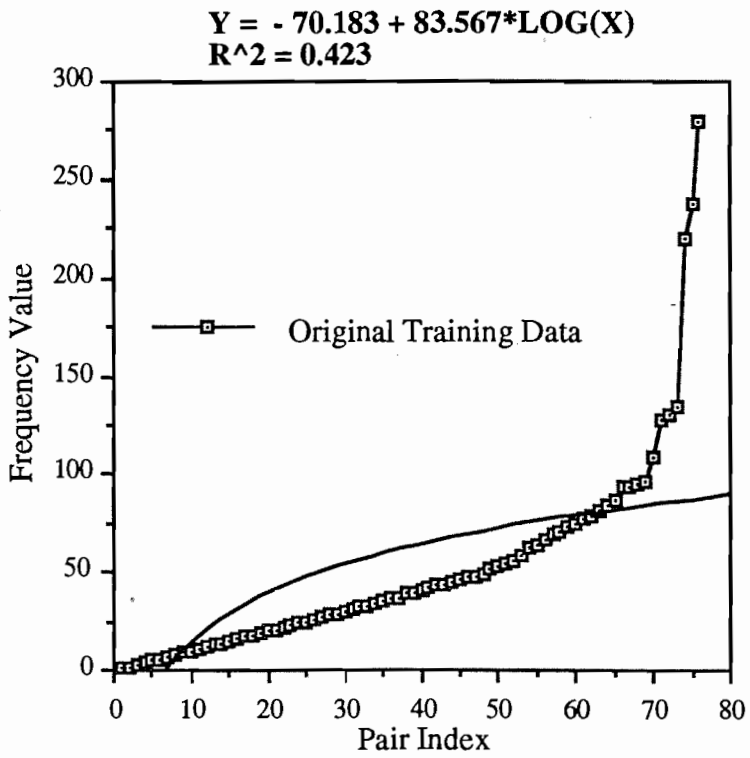

Figure 1. The 76-Class Grouping and the Log Function Are Used

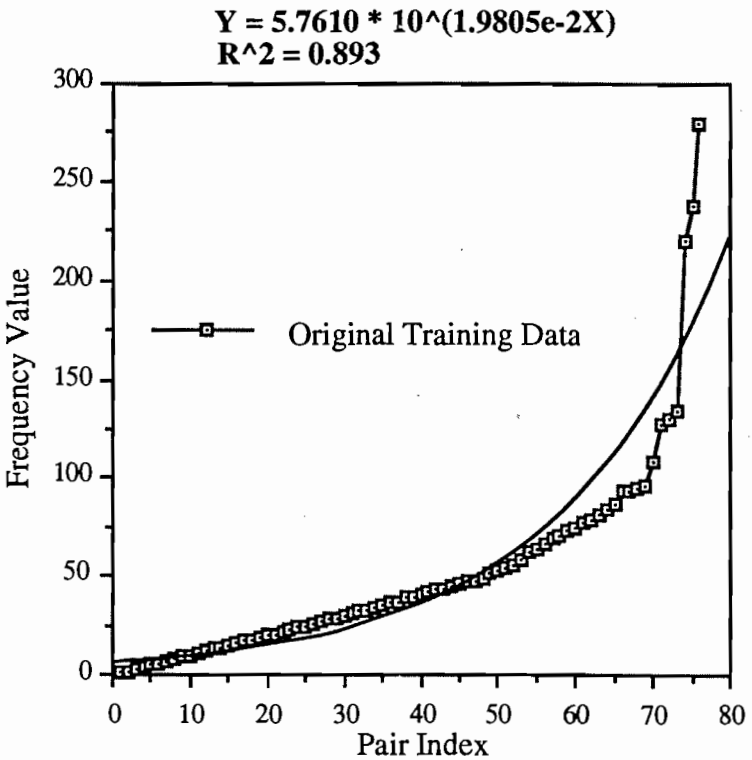

Figure 2. The 76-Class Grouping and the Exponential Function Are Used

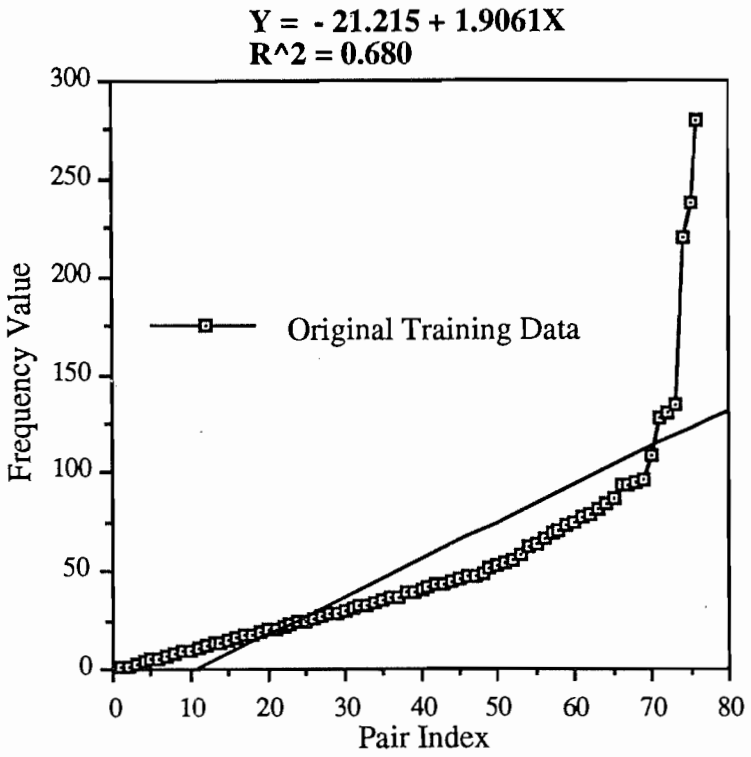

Figure 3. The 76-Class Grouping and the 1st Degree Polynomial Function Are Used

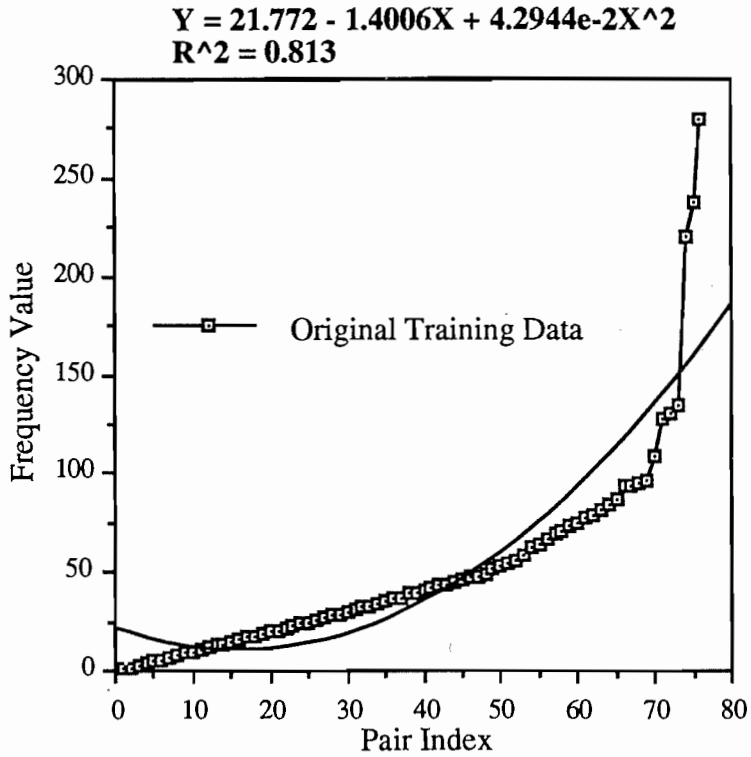

Figure 4. The 76-Class Grouping and the 2nd Degree Polynomial Function Are Used

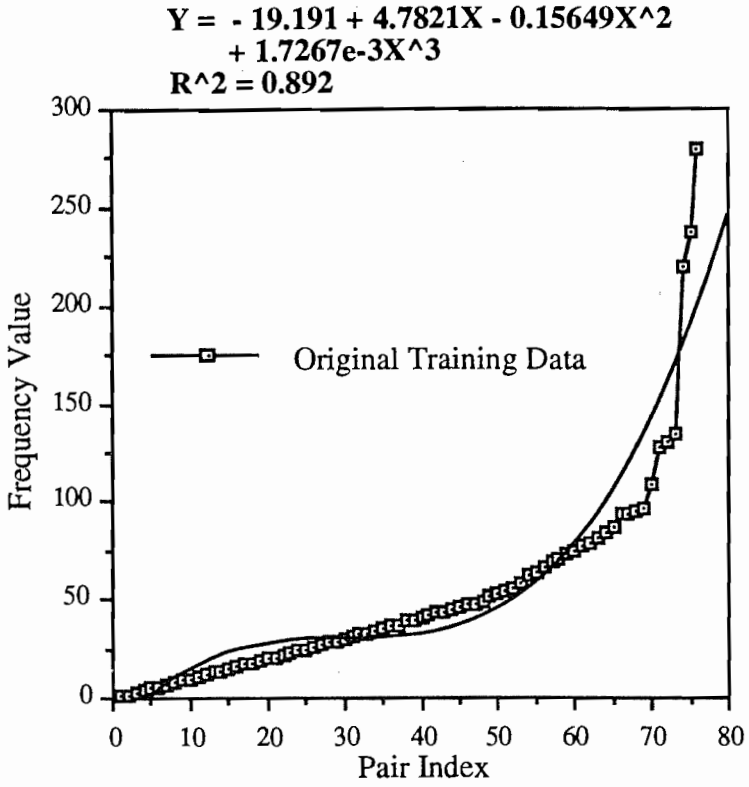

Figure 5. The 76-Class Grouping and the 3rd Degree Polynomial Function Are Used

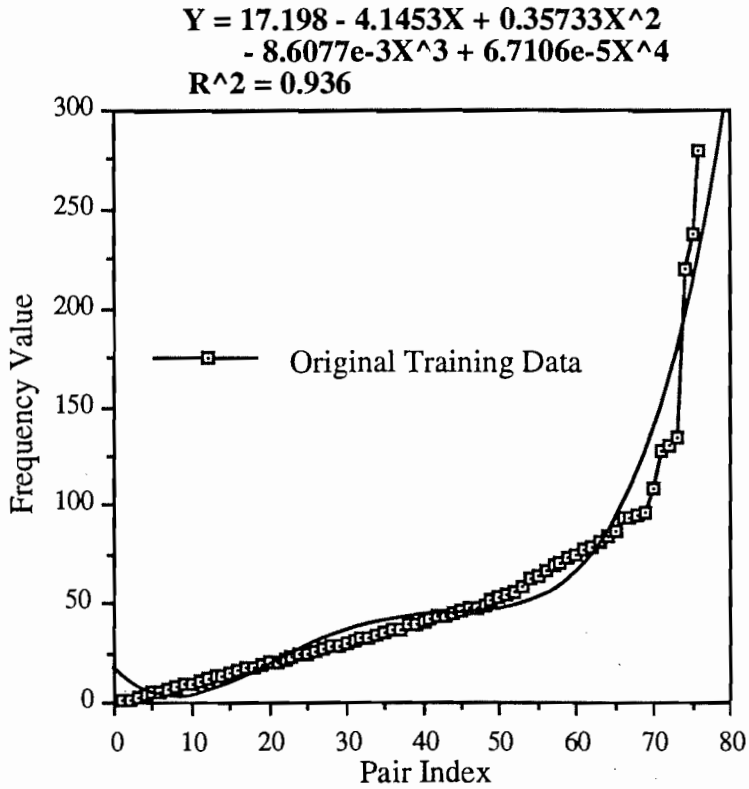

Figure 6. The 76-Class Grouping and the 4th Degree Polynomial Function Are Used

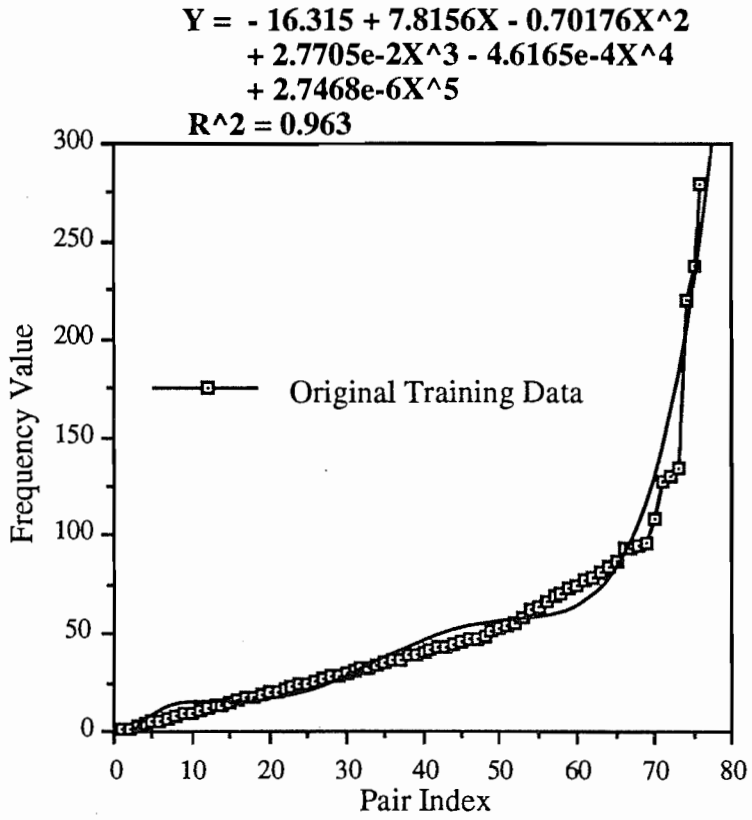

Figure 7. The 76-Class Grouping and the 5th Degree Polynomial Function Are Used

R^2 in these figures denotes the degree of the similarity between the training table and the curve function. The function

Y=-16.315+7.8156X-0.70176X^2+2.7705e-2X^3-4.6165e-4X^4+2.7468e-6X^5

has the highest R^2. Thus, it is selected as our testing function for 76-class case in the following experiment. Note that if the function returns a negative value then the value will be reset to 1. Similarly, we can find the curve function to fit the training table for 100-class grouping. The function

Y=-12.619+4.9581X-0.30176X^2+ 8.9514e-3X^3-1.1579e-4X^4+5.3913e-7X^5

is selected as our testing function for 100-class case in the following experiment (R^2: 0.941). The negative function value is treated in the same way, i.e., it will also be reset to 1.

2.3 Experimental Results

In the experiments, we use word association language model (called MM language model later) and forward training model to generate Chinese sentences described in Lee [9]. The forward training model means that the direction of word association is forward in training. For example, given a sentence $S=w_1, w_2, w_3$. Only word association pairs (w_1,w_2), (w_1,w_3) and (w_2,w_3) in forward direction will be generated. Consider a word sequence $S=w_1, w_2, ..., w_n$ as one of the arrangement of the words. The probability of the word sequence is measured as follows:

$$P(S)=P(w_1, w_2, ..., w_n) \cong \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} P_f(w_i,w_j)$$

where $P_f(w_i,w_j)$ is the probability of the word association between word w_i and w_j under forward training model.

$P_f(w_i,w_j)$ is defined as follows:

$$P_f(w_i,w_j)=\frac{F_f(w_i,w_j)}{\sum_{i=1}^{n} \sum_{j=1}^{n} F_f(w_i,w_j)}$$

where $F_f(w_i,w_j)$ is the frequency of words w_i and w_j that appear in the same sentence under forward training model.

Besides, there are two constraints used in our experiments to improve the system performance:

(1) Word/Word Linear Relation.

(2) POS/POS Linear Relation, where POS denotes part of speech.

Type (1) constraint is a set of constraint pairs (w_1,w_2). The pair (w_1,w_2) means word w_2 follows by word w_1 in the training corpus. Similarly, type (2) constraint is a set of constraint pairs (POS1,POS2). In this case, POS1 and POS2 appear in succession in the training corpus. Type (1) and type (2) constraints are enforced on the language models to eliminate the illegal combinations. Consider type (1) constraint and an arrangement of the words $S=w_1, w_2, ..., w_n$. This arrangement will be discarded if there exists any (w_i,w_{i+1}) ($1 \leq i \leq n-1$) pair in the arrangement such that it does not satisfy the constraint. In order to use POS as constraints, the BDC corpus is tagged with BDC tag set. The experimental results are shown in Table 3.

Table 3. Experiment Results

Experiment	# of Test Sentences	Sentence Length	Original Language Model	Curve Fitting Model	Decrease (Correct Rate)
1	1000	1~6	82.8%	79.5%	3.3%
2	633	7~9	72.5%	68.8%	3.7%
3	1000	1~6	99.8%	99.7%	0.1%
4	633	7~9	99.5%	99.4%	0.1%
5	1000	1~6	83.2%	80.0%	3.2%
6	1000	1~6	82.8%	77.4%	5.4%
7	633	7~9	72.5%	66.3%	6.2%
8	1000	1~6	99.8%	99.5%	0.3%
9	633	7~9	99.5%	99.2%	0.3%
10	1000	1~6	83.2%	77.9%	5.3%

The experiments adopt different grouping and different language models shown as follows:

(1) Experiments 1 and 2

The 76-class grouping and MM language model are used.

(2) Experiments 3 and 4

The 76-class grouping and MM Language Model with type (1) constraint are used.

(3) Experiment 5

The 76-class grouping and MM language model with type (2) constraint are used.

(4) Experiments 6 and 7

The 100-class grouping and MM language model are used.

(5) Experiments 8 and 9

The 100-class grouping and MM language model with type (1) constraint are used.

(6) Experiment 10

The 100-class grouping and MM language model with type (2) constraint are used.

The curve fitting model is the original word association language model, but it uses the function instead of the training table. The experimental results show that there is a little error rate introduced by the curve function. The second grouping (100 classes) has a worse performance than the first grouping (76 classes) in our experiments because the latter has higher R^2. We evaluate this storage reduction method by the four conditions mentioned in Section 1:

(1) The result seems satisfactory because the method performs the closed performance and it just uses a little disk space. The training table, the index table, and the type (1) constraint table in the original language model occupy 2.23 M, 0.16 M and 0.61 M bytes respectively. In the pure curve fitting model, i.e., no type (1) constraints, no extra disk storage is required.

(2) The grouping is done with the frequency values of the word pairs so that no human interference is required.

(3) The approach can be applied to different language models such as Markov model, word association model, hybrid model, and so on.

(4) The frequency value is computed by function application, not by table look up. In the conventional storage management approach, it may take much time in disk I/O when the frequency value is retrieved.

3. The Classification Problem

The result seems satisfactory, but a classification problem is introduced, that is, how to know what class a given word pair ($word_1$,$word_2$) belong to. It is a problem for all class-based approaches. The next subsections will propose a neural network approach to deal with this problem.

3.1 Neural Network Approach

The overall neural network architecture is shown in Figure 8.

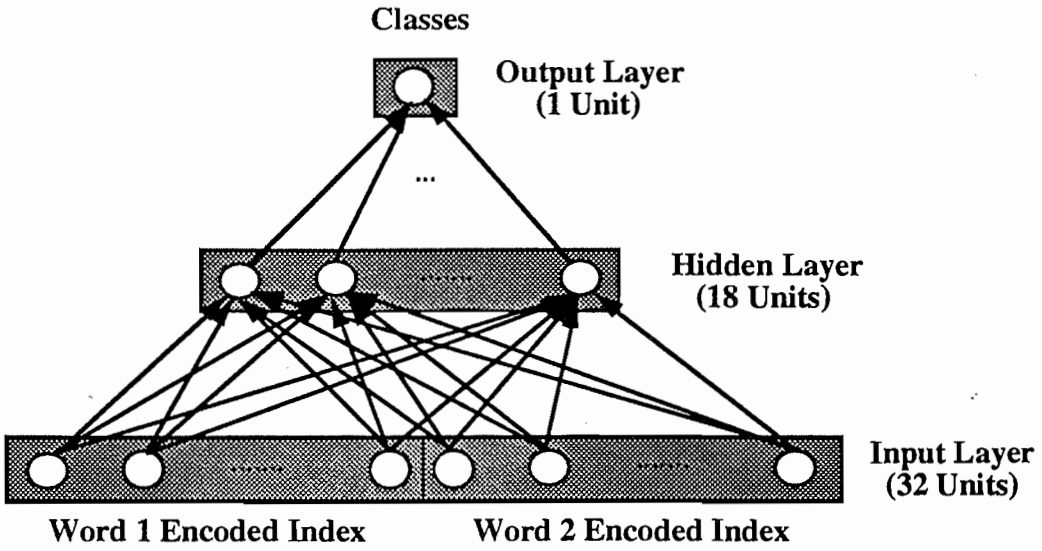

Figure 8. The Overall Neural Network Architecture

This neural network is a 3-layer feed-forward network with one hidden layer. In the input layer, word 1 and word 2 are all encoded into 16 bit vectors. For example, if the indices of words 1 and 2 are 8 and 15 respectively, then word 1 will be encoded into (0000000000001000) and word 2 will be encoded into (0000000000001111). The training adopts the back-propagation algorithm [10, 11], which uses the gradient descent to change link weights to reduce the difference between the network output and the desired output. In this algorithm, sigmoid function is used as nonlinear activation function. The sigmoid function is shown below:

$$F(y)=(1+e^{-\beta y})^{-1}$$

This function is continuous and varies monotonically from 0 to 1 as y varies from $-\infty$ to ∞. The gain of the sigmoid function, β, determines the stepness of the transition region. In our experiment, β is set to 1.0. This task is a many-to-one mapping problem. Thus, it is easy to train. In the output layer, there is only one unit. Because this unit will output a value whose range is from 0 to 1, we define the classes over this range. That is, if there are five classes, then the ranges of these classes are assigned to the five open intervals, (0.0,0.2), (0.2,0.4), (0.4,0.6), (0.6,0.8) and (0.8,1.0) respectively. After convergence, the training process confirms that the critical values such as 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0 cannot appear.

3.2 Experimental Results

Four experiments are considered. Each selects different set of test data.

Experiment 1: Group or classify the last 50 classes in 76 classes, i.e. 125 pairs.

Experiment 2: Group or classify the first 5 classes in 76 or 100 classes, but only use (1/10000) of the pairs, i.e., 17 pairs.

Experiment 3: Group or classify the first 5 classes in 76 or 100 classes, but only use (1/1000) of the pairs, i.e. 154 pairs.

Experiment 4: Group or classify the first 5 classes in 76 or 100 classes, but only use (1/100) of the pairs, i.e. 1545 pairs.

In these experiments, each pair is identified correctly. The results show that the neural network approach is suitable to deal with the classification problem, but it still has the longer training time problem.

4. Zero Frequency Problem

Last section proposes a neural network approach to deal with the pair classification problem. It can correctly identify the pairs with nonzero frequency. However, there still exists a serious problem called *zero frequency problem.* That is, we cannot tell out the word pairs that do not appear in the training table completely. In our experiments, 99.88% of the total belong to this kind of pairs, i.e., 132646496: 132802576. For the treatment of this problem, a preprocess procedure is taken. If the training data is stored in a 2-dimensional matrix (11524x11524 in our experiment), then it must be a sparse matrix. We reassign the word indices to move the zero pairs to the left-upper of the matrix. Figures 9 and 10 demonstrate the distribution of the word pairs of the training table before and after the preprocessing. Points in these figures denote the word pairs with nonzero frequency.

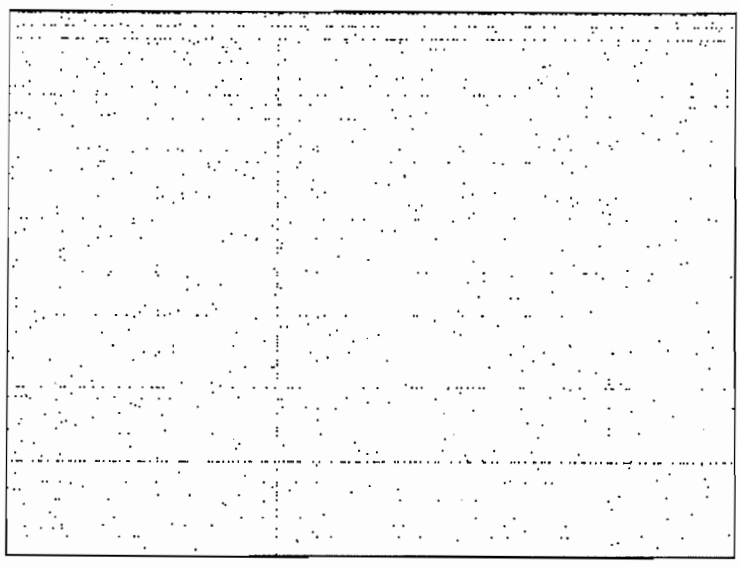

Figure 9. The Distribution of Word Pairs before Preprocessing

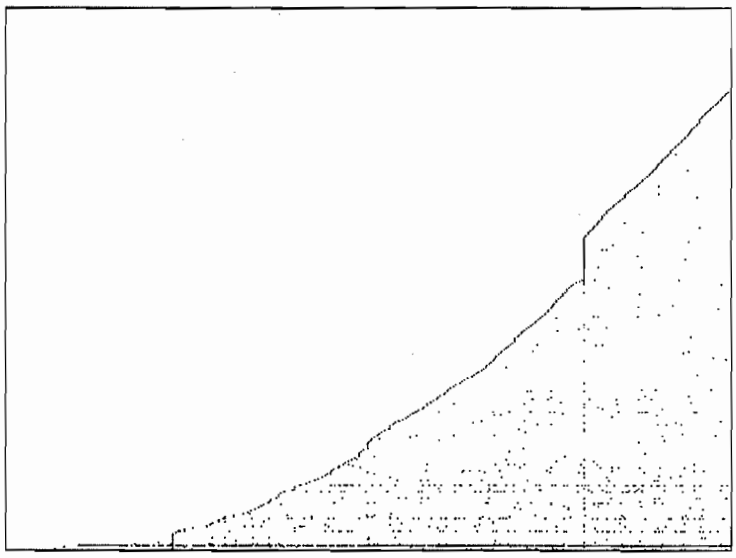

Figure 10. The Distribution of Word Pairs after Preprocessing

The experimental results show 72.5% of zero points can be rearranged to the left-upper corner. We can only record their indices (row number, column number), which occupy little space. However, the number of the remainder zero points is still very large (36477786). A sampling method is proposed to reduce the quantity in the neural network training. Given a continuous zero points P1, P2, ..., P*n*, only P1 and P*n* are taken as samples. At this step, 5862 samples (0.016%) are selected. The integration of this method to the large training data management can refer to [12].

5. Concluding Remarks

In this paper, we propose a storage reduction method to solve the problem that the training table is too large. Mathematical function is used to simulate the distribution of the frequency value of the pairs in the training table. The experimental results show that although there is a little error rate introduced by the curve function, this approach has the advantages of little space requirement, no human interference, no application limitation and faster processing speed. The neural network approach is also proposed to deal with the pairs classification problem. The experimental results show its feasibility.

References

[1] F. Jelinek, *et al.*, "Classifying Words for Improved Statistical Language Models," *Proceedings of IEEE ICASSP,* 1990, pp. 621-624.

[2] A. Martelli, "Stochastic Modeling of Language Via Sentence Space Partitioning," *Proceedings of Third Conference of the European Chapter of the ACL*, 1987, pp. 91-93.

[3] P.F. Brown, *et al.*, "Class-Based N-Gram Models of Natural Language," *Computational Linguistics*, Vol. 18, No. 4, 1992, pp. 467-479.

[4] H.C. Danon and M.E. Beze, "Three Different Probabilistic Language Models: Comparison and Combination," *Proceedings of IEEE ICASSP*, 1991, pp. 297-300.

[5] P. Dumouchel, *et al.*, "Three Probabilistic Language Models for a Large-Vocabulary Speech Recognizer," *Proceedings of IEEE ICASSP*, 1988, pp. 513-516.

[6] G. Maltese and F. Mancini, "An Automatic Technique to Include Grammatical and Morphological Information in a Trigram-Based Statistical Language Model," *Proceedings of IEEE ICASSP*, 1992, pp. 157-160.

[7] M. Nakamura, *et al.*, "Neural Network Approach to Word Category Prediction for English Texts," *Proceedings of COLING*, 1990, pp. 211-219.

[8] T.C. Bell, *et al.*, *Text Compression*, Prentice Hall Advanced Reference Series, Computer Science, 1990.

[9] Y.S. Lee, *Generating Chinese Sentences: A Corpus-Based Approach*, Master Thesis, Department of Computer Science and Information Engineering, National Taiwan University, 1993.

[10] P.D. Wasserman, *Neural Computing: Theory and Practice*, Van Nostrand Reinhold, 1989.

[11] D.R. Hush and B.G. Horne, "Progress in Supervised Neural Networks," *IEEE Signal Processing Magazine*, January, 1993, pp. 8-39.

[12] H.H. Chen and Y.S. Lee, "Very Large Training Data Management in Corpus-Based Language Modeling," Submitted to *IEEE Transaction on Signal Processing*.