Some Issues on Applying SA-class Bigram Language Models

Chao-Huang Chang (張照煌) and Cheng-Der Chen (陳正德)
E000/CCL, Building 11, Industrial Technology Research Institute
Chutung, Hsinchu 31015, TAIWAN, R.O.C.
changch@e0sun3.ccl.itri.org.tw

Abstract

This paper investigates some issues on application of class-based Chinese language models, especially the SA-class bigram model in which the word classes are automatically clustered by simulated annealing. The studied issues include (1) using test-set perplexity as a quality measure for evaluating performance of language models across domains, subdomains, and character codings; (2) using the SA-class bigram model to different applications – OCR postprocessing, syllable-to-character conversion, and linguistic decoding for speech recognition; (3) comparing the model with other language models – least-word, word-frequency, inter-word character bigram, and word bigram; and (4) deciding appropriate number of classes based on corpus size. The experimental results show that the test-set perplexity is indeed a good measure for performance evaluation of language models, and the SA-class bigram language model is not only theoretically plausible but also practically feasible – high performance with less resource requirement.

1 Introduction

Statistical language models have become mainstream in computational linguistic research because rule-based models lack robustness and expandability. Figure 1 shows our proposed taxonomy of statistical language models. On the left-hand side are the models without classification, including character n-gram and word n-gram. These models are simple and useful. However, available text corpora are not large enough to estimate the huge number of parameters in these models. For example, in a 50,000-word system, the number of parameters for word bigram is $NV^2 - 1 \approx 2.5 \cdot 10^9$ (NV: number of words) and heuristically needs $2.5 \cdot 10^{10}$ to estimate appropriately. Typical size of available text corpora is about 10^6 to 10^7. There is still a difference of order 1000. Thus, class-based language models [1] have been proposed to deal

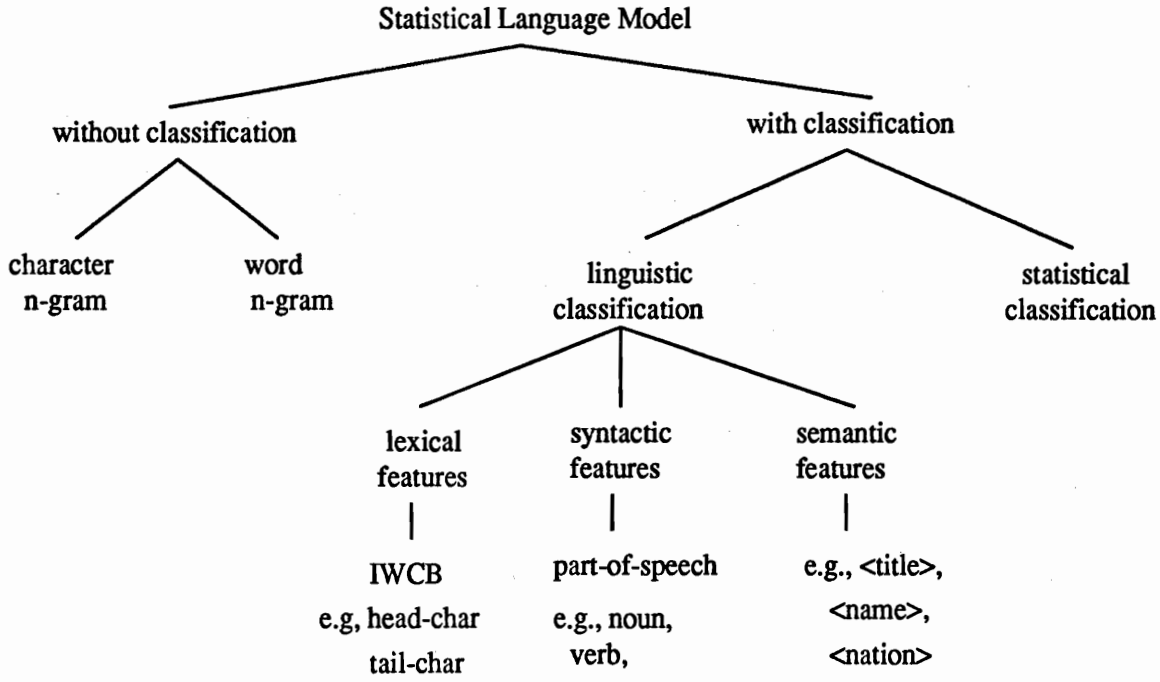

Figure 1: Taxonomy of Statistical Language Models

with the data sparseness problem. In other words, the words in the vocabulary are classified into word groups. The number of parameters for class bigram is reduced to $NC^2 - NC + NV - 1$ (NC: number of classes) [1]. If $NC = 500$, it would be about $3 \cdot 10^5$ and needs $3 \cdot 10^6$ words to estimate. The classification can be either *supervised* (linguistic) or *unsupervised* (statistical). Supervised classification is usually based on linguistic categories, such as morphological features [15], grammatical parts-of-speech [6,10,11], and semantic categories [19]. Unsupervised classification is corpus-based according to statistical characteristics, such as perplexity [5,13] or mutual information [1]. In Chang and Chen [5], we proposed an unsupervised classification based on perplexity by simulated annealing clustering and suggested applications of the results to Chinese language models. We call the proposed model SA-class n-gram model. The practical value of the model needs to be proved by real-world applications. In this paper, we will show the experimental results of applying the model to various tasks and investigate the following issues:

1. test-set perplexity as quality measure for evaluating performance of language models across domains, subdomains, and character codings (Big-5 or GB);

2. using the SA-class bigram model to different applications – OCR postprocessing [4], syllable-to-character conversion, and linguistic decoding for speech recognition;

3. comparing the model with other language models – least-word, word-frequency, inter-word character bigram, and word bigram; and

4. deciding appropriate number of classes based on corpus size.

2 SA-Class Bigram Language Models

In this section, we give a brief review of automatic word clustering by simulated annealing and the SA-class bigram language model. For more detail, see Chang and Chen [5].

2.1 Automatic word clustering by simulated annealing

Given a text corpus $S = w_1, w_2, ..., w_L$ with NV word types and L word tokens, the word clustering problem is, thus, to seek the best partitioning of the NV word types into a predefined number NC of word classes maximizing the probability of the word sequence according to the language model. In other words, it is a partitional clustering problem [1] with (1) NV words as objects to be clustered, (2) NC as the desired number of clusters, and (3) the estimated probability $\hat{p}(S)$ of the given word sequence as the clustering criterion. Typical values for NV and NC are 50,000 and 200. A partitioning can be considered as a mapping ϕ of NV elements in which each element is the class label the corresponding word belongs to.

For a class bigram model, find a partitioning ϕ to maximize $\hat{p}(S)$

It is straightforward to apply simulated annealing algorithms [14] to the word clustering problem: (1) a feasible partitioning or class assignment ϕ specifies the **configuration**, A, (2) to rearrange the elements in a configuration, simply randomly choose a word and assign it to a randomly chosen cluster, (3) use a probabilistic measure, such as perplexity, as the objective function (described below), and (4) follow the Metropolis algorithm [17] to specify the annealing schedule.

An **annealing schedule** specifies time and duration to decrease the control parameter (or temperature) T. Initially, T is set to T_0, and multiplied by an annealing factor α after a fixed number of trials i_{max}. The algorithm stops when the temperature T is lower than the final temperature T_f. The original Monte Carlo optimization accepts a new configuration only if the objective function value improves, suffers from the local minimum problem. Metropolis *et al.* [17] proposed in 1953 that a worse configuration can be accepted according to the control parameter T. The new configuration is accepted if $exp(\Delta PP/T)$ is greater than a random number between 0 and 1, where ΔPP is the difference of perplexities for two consecutive steps.

The simulated annealing word clustering algorithm is summarized as follows:

Definition of variables:

A:	current class assignment	A_{new}:	perturbed class assignment
T:	temperature (control parameter)	T_0:	initial temperature
T_f:	final temperature	α:	annealing factor
i:	number of trials in T	i_{max}:	max. i in T
r:	number of rejected trials	r_{max}:	convergence threshold
P:	perplexity of A	P_{new}:	perplexity of A_{new}
ΔP:	difference of P and P_{new}		

begin

1. initialize the class assignment A to be A_0;
   $T := T_0$; i := 0; r := 0; $P :=$ perplexity-of (A_0);
2. **repeat** {
   2.1 $i := i + 1$;
   2.2 randomly reassign a word to form A_{new};
   2.3 $P_{new} :=$ perplexity-of (A_{new});
   2.4 $\Delta P := P_{new}$ - P;
   2.5 **if** $\Delta P < 0$ **or** $e^{-\Delta P/T} >$ a random number in [0..1] **then** /* accept */
   $A := A_{new}$; $P := P_{new}$; r := 0;
   else /* reject */
   $r := r + 1$;
   endif
   2.6 **if** $i \geq i_{max}$ **then**
   $T := T * \alpha$; i := 0;
   endif
   } **until** $r \geq r_{max}$ **or** $T \leq T_f$;

end.

clustering by simulated annealing and For more detail, see Chang and Chen [5].

2.2 The SA-class bigram language model

The classification results of words (or characters) can be used in language models for speech recognition or OCR postprocessing. The procedure for building an SA-class bigram language

model includes the following steps:

1. For a given text corpus, use a word-segmentation program to divide the character stream into a word stream;
2. Build a word list and word bigrams based on the word stream;
3. Decide the number of classes you want;
4. Run the simulated annealing clustering until converging or terminating conditions are met;
5. Store the SA-class-ids for the words in the system dictionary;
6. Map words in a new input sentence automatically to the SA-classes through dictionary look-up;
7. Complete the SA-class bigram model using a smoothing scheme for unseen word problem.

3 Applications of SA-class Bigram Model

We describe here experimental results of applying the SA-class bigram model to three tasks: OCR postprocessing, syllable-to-character conversion, and linguistic decoding for speech recognition.

3.1 The model and three competing models

The language models for these problems can be usually summarized as seeking the optimal path in a word-lattice formed by candidate characters. The path probability of a word-lattice path is the product of lexical probabilities and contextual SA-class bigram probabilities. For a path of F words $H = W_1, W_2, ..., W_F$, the path-probability estimated by the language model is

$$P_{LM}(H) = (\sum_{i=1}^{F} P(W_i|\phi(W_i))) * (\sum_{i=2}^{F} P(\phi(W_i)|\phi(W_{i-1})))$$

We will compare our model with three other models: Least-word model (LW), Word-frequency model (WF), and Inter-word character bigram model (IWCB).

Least-word model (LW)

A simple language model is based on a dictionary (actually a wordlist). The cost function of the model is the number of words in the word-lattice path. The best path is simply one with the least number of words, $P_{LM}(H) = (-1) * F$. This is similar to the principle of Maximum Matching for Chinese word segmentation.

Word-frequency model (WF)

Another simple model is based on the word frequencies of the words in the word-lattice path. This can be considered as a word unigram language model. The path probability is the product of word probabilities of the words in the path.

Inter-word character bigram model (IWCB)

This model is a variation of the *word-lattice-based Chinese character bigram*proposed by Lee *et al.* [15]. The path probability is computed as the product of word probabilities and inter-word character bigram probabilities of the words in the path. For path H: $W_1 = W_{i_1 j_1}$, ..., $W_F = W_{i_F j_F}$,

$$P_{LM}(H) = (\sum_{k=1}^{F} P(W_k))) * (\sum_{k=2}^{F} P(C_{i_k}|C_{j_{k-1}})$$

where C_{i_k} and C_{j_k} are the first and last characters of the k-th word, respectively.

This model is one of the best among the existing Chinese language models, and has been successfully applied to Chinese homophone disambiguation and linguistic decoding [3,15].

3.2 The corpora and word bigrams

Three large corpora are involved in the experiments: the 1991 UD newspaper corpus (`1991ud`), the seventh-day subcorpus (`d7`), and an electronic politic news message corpus (`poli2`). These corpora have been preprocessed to clear up irrelevant materials, such as sentences containing Arabics, English alphabets, and typesetting commands. Simple statistics for the corpora are summarized in Table 1.

The 1991 UD newspaper corpus (`1991ud`) of more than seven million characters has been used for collecting word frequencies in the WF model, the character bigrams in the IWCB

Corpus	#sentences	#characters	#word-tokens	#word-types
1991ud	579,123	7,312,979	4,761,120	60,585
d7	42,273	540,454	353,876	25,346
poli2	6,930	92,710	62,433	7,622

Table 1: Statistics for the Three Corpora

model, and the word bigrams used in simulated annealing word clustering. The d7 subcorpus, a part of 1991ud, was used for studying the effect of training data size and number of word classes.

An independent set of electronic news messages, poli2, were collected for evaluating the performance of language models. poli2 is different from the other two corpora in both publisher and time period. poli2 contains 6,930 sentences or 92,710 Chinese characters.

The two corpora for word clustering, 1991ud and d7, are first segmented automatically into sentences, then into words by our Viterbi-based word identification program VSG [7]. The same lexicon and word hypothesizer are used in the language models.

3.3 Task 1: OCR postprocessing

Recognition of Chinese texts is usually performed in two steps: (1) recognition of printed or handwritten Chinese characters; and (2) contextual postprocessing of the multiple-candidate recognition results. The latter uses contextual linguistic constraints to improve the recognition accuracy of the former and is the focus of this section.

The problem of contextual postprocessing can be described as follows: the character recognizer produces top K candidates $C_{i1}, ..., C_{iK}$ (with similarity score) for each character I_i in the input text of N characters $I = I_1, I_2, ...I_N$; the postprocessor then decides which of the K candidates is correct based on the context and a language model. Thus, the recognizer produce the candidate matrix $M_{NK} = \{C_{ij}, i = 1, ..., N, \text{and} j = 1, ..., K\}$ for the input text I. The postprocessor is to find the combination with highest probability according to the language model: $O = O_1, O_2, ...O_N = argmax\ P(O|M)$ among the K^N feasible combinations.

The overall probability can be divided into two parts: pattern recognition probability and linguistic probability, $P(O|M) = P_{PR}(O|M) * P_{LM}(O|M)$. The former is produced by the recognizer, while the latter is defined by thr language model.

3.3.1 Handwriting recognition

We have used a state-of-the-art Chinese handwriting recognizer [16] developed by ATC, CCL, ITRI, Taiwan as the basis of our experiments. The CCL/HCCR1 handwritten character database (5401 character categories, 200 samples each category) [18] was first automatically sorted according to character quality [9], then was divided into two parts: the odd-rank samples for training the recognizer, the even-rank samples as held-out test data.

We have used for our experiments twenty sets of even-rank character samples, which are the samples with quality ranks 10, 20, ..., and 200. These samples are classified into four sets: (A-1) best-quality samples ranked 10, 20, ..., 60, with ATC recognizer's accuracy around 95%; (A-2) good-quality ones ranked 70 – 110, with accuracy around 90%; (B-1) fair-quality ones ranked 120 – 160, with accuracy around 85%; and (B-2) bad-quality ones ranked 170 – 200, with accuracy below 80%. This classification is helpful for performance analysis of contextual postprocessors for different quality of recognizers and input texts.

Experimental results show that the recognition accuracies, in terms of character categories, would be 95.53%, 90.79%, 84.66%, and 73.77% for the A-1, A-2, B-1, B-2 sets, respectively.

The `poli2` corpus of 92,710 Chinese characters was used for evaluating the performance of contextual postprocessing. The recognition results for the twenty sets of character samples were used as the basis of evaluation. (The corpus contains 52 uncommon characters which do not belong to any of the 5401 character categories.) Therefore, we recompute the recognition accuracies by the ATC recognizer for the four quality classes of samples. Without postprocessing, the accuracies are 96.11%, 90.45%, 84.57%, and 71.77%, for A-1, A-2, B-1, B-2 classes, respectively. To ease comparison, we have set the number of candidates K to 6 in all the experiments. Thus, the characters ranked after 6 and the 52 uncommon characters are impossible to recover using the postprocessor. The upper bounds for performance of language models are thus with accuracies 99.69%, 98.78%, 96.39%, and 90.87%, for A-1, A-2, B-1, B-2 classes, respectively.

3.3.2 Postprocessing with SA-class bigrams

Table 2 summarizes the experimental results of postprocessing for the four classes of character samples. The NoGram (No Grammar) column lists the accuracies without postprocessing; the UpBound column shows the upper bounds with $K = 6$; and the rest four columns list the accuracies after postprocessing with the LW (Least-Word), WF (Word-Frequency), IWCB

Quality	NoGram	UpBound	LW	WF	IWCB	300/ud
A-1 (best, 6 sets)	96.11	99.69	96.03	97.19	98.88	**99.31**
A-2 (good, 5 sets)	90.45	98.78	94.03	95.08	97.62	**98.00**
B-1 (fair, 5 sets)	84.57	96.39	90.01	90.86	94.55	**94.87**
B-2 (bad, 4 sets)	71.77	90.87	81.71	82.35	**87.66**	86.95
Ave. (20 sets)	86.94	96.87	91.16	92.11	95.24	**95.40**

Table 2: Comparison of four models: accuracies (%)

(Inter-word Character Bigram), and 300/ud (SA-class bigram, $NC = 300$, trained with 1991ud) models, respectively. We can observe that the SA-class bigram model out-performed the other three models in general. The order of performance is: $300/ud > IWCB > WF > LW$. The average error rates are – Recognizer: 13.06%, LW:8.84%, WF:7.89%, IWCB:4.76%, and ud/300: 4.60%. However, in case of class B-2, IWCB is better than SA-class. When the recognizer's accuracy becomes too low, words are not easy to compose. The word lattice is, thus, mostly composed of single-character words. Word cooccurrences become character cooccurrences. That is why the IWCB model (using character bigram) performs better than our model (using word class bigram) for the bad-quality samples. Besides, the storage requirement of our model is much less than that of IWCB model.

3.3.3 Analysis and comparison of correction results

change types	LW	WF	IWCB	300/ud
XO	6.91	8.13	9.27	8.76
OX	2.69	2.96	0.97	0.30
XX	0.97	2.12	2.42	1.79
Gain	4.22	5.17	8.30	8.46

Table 3: Analysis of correction results (%)

The changes the postprocessor makes can be classified into three types: wrong-to-correct (XO), correct-to-wrong (OX), and wrong-to-wrong (XX). In the XO type, a wrong character (i.e., a recognition error) is corrected; in the OX type, a correct character is changed to a wrong one; and in the XX type, a wrong character is changed to another different wrong one. The performance of the postprocessor can be evaluated as the net gain, $\#XOs - \#OXs$. Table 3 summarizes the experimental results by types of change. The rows XO, OX, XX, and Gain list the percentages of characters in types XO, OX, XX, and XO-OX, respectively. The percentages are based on the whole samples, i.e., $92,710 \cdot 20$ characters. For example, the XO, OX, XX

types of change for our 300/ud model are 8.76%, 0.30%, and 1.79% of the whole samples, respectively. The IWCB model usually corrects more errors than ours, while it also commits much more OX mistakes.

3.4 Task 2: Syllable-to-character conversion (intelligent phonetic input)

We use the concepts of bidirectional conversion and automatic evaluation [2] to conduct the syllable-to-character conversion experiments. The same corpus `poli2` and the SA-class model ud/300 are used. The conversion accuracy is 95.07%. This can be compared with our previous results – POS bigram: < 89%, WF model: 91%, IWCB model: 93.46%, and IWCB with adaptation: 94.8%.

3.5 Task 3: Linguistic decoding for speech recognition

Our speech recognition systems is still under development. Using intermediate results from versions of a speaker-dependent, isolated-syllable prototype, we have the following results (Table 4). For example, Sample A has accumulated syllable accuracies– top-1: 88%, top-2: 96%, top-3: 98%, and the character accuracy from language model is 91.25%.

Sample	top-1	top-2	top-3	accuracy
A	88%	96%	98%	91.25%
B	63%	78%	84%	80.04%

Table 4: SA-class bigram for linguistic decoding

4 Perplexity and Performance

In this section, we will study the relationship between test-set perplexity and performance of language models. To ease comparison and discussion, the annealing schedule $(T_0, T_f, \alpha, i_{max}, r_{max})$ is empirically set to $(0.1, 10^{-20}, 0.9, 20000, 5000)$ in all the experiments.

4.1 Different number of classes

In the context of OCR postprocessing, Table 5 shows the perplexities and accuracies after correction for our models with different numbers of class (NC) and different training corpora.

model	testPP	A-1	A-2	B-1	B-2	Average
50/d7	1,690	99.20	97.69	94.41	86.27	95.04
100/d7	1,553	99.21	97.76	94.48	86.42	95.11
150/d7	1,524	99.19	97.75	94.49	86.51	95.12
200/d7	1,522	99.20	97.77	94.53	86.44	95.12
250/d7	1,543	99.18	97.71	94.50	86.44	95.10
300/d7	1,560	99.17	97.73	94.47	86.49	95.10
500/d7	1,689	99.13	97.63	94.37	86.26	94.99
50/ud	1,464	99.26	97.88	94.64	86.71	95.25
100/ud	1,301	99.28	97.95	94.79	86.82	95.33
150/ud	1,219	99.29	97.93	94.77	86.86	95.34
200/ud	1,188	99.29	97.96	94.78	86.90	95.35
250/ud	1,152	99.30	97.97	94.88	86.89	95.38
300/ud	1,112	99.32	98.01	94.88	87.03	95.42
500/ud	1,036	99.33	97.99	94.91	87.09	95.44
600/ud	1,013	99.32	98.00	94.97	87.09	95.46
900/ud	991	99.34	98.04	94.98	87.10	95.47
1000/ud	988	99.33	98.06	94.96	87.11	95.48

Table 5: Comparison of our models with different NC/corpus

We can see that there is a clear relationship between test-set perplexity and correction performance: the lower the perplexity, the higher the correction performance. In other words, perplexity is indeed a good measure for evaluating performance of language models, for speech recognition or for text recognition.

4.2 Size of training corpus

The size of training corpus is important: all models trained with `1991ud` (4.76 million words) performed much better than those models trained with `d7` (0.35 million words). The perplexities for the `d7` models are between 1,522 and 1,690, while those for the `1991ud` models are between 988 and 1,464.

The optimal NC value for the `d7` class bigram models is between 100 and 200: 100 for A-1, 200 for A-2 and B-1, and 150 for B-2. This is consistent to the common rule of thumb: the size of training data should be at least ten times the number of parameters, which suggests an NC value of approximately 100 for the size of the `d7` corpus. The $NC = 500$ models are apparently overtrained, which is consistent to the evaluation of test set perplexities we discussed in Chang and Chen [5]

4.3 Test-set perplexity for a Kung-Fu novel

We tried a rather different domain of texts for comparison. Part of a Kung-Fu novel is collected from an electronic newsgroup: 28,069 word-tokens, 4,274 word-types, 237 unseen, 920 low-freq. words. The test-set perplexity is computed for both `poli2` and the novel. As we can see from Table 6, the difference is quite significant: the perplexity is doubled. However, the perplexity is almost consistent across domains with an exception (ud/250).

1991ud	*Model*	poli2	Kung-Fu
1,247	ud/100	1,301	2,524
1,177	ud/150	1,219	2,489
1,148	ud/200	1,188	2,484
1,108	ud/250	1,152	2,528
1,069	ud/300	1,115	2,441
962	ud/500	1,036	2,402
917	ud/600	1,013	2,375

Table 6: Test-set Perplexity for a Kung-Fu novel

4.4 Test-set perplexity across subdomains

Table 7 shows the test-set perplexity across subdomains: subnewsgroups. The electronic newsgroups are close to the newspaper domain of our training corpus. Thus, we can see the range of test-set perplexities for these subdomains is close: between 816 and 1074 for ud/600.

1991ud	*NC*	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1,247	100	1,253	1,231	1,293	1,026	1,164	1,252	1,052	1,339
1,177	150	1,181	1,185	1,212	987	1,086	1,158	983	1,272
1,148	200	1,142	1,152	1,186	939	1,048	1,138	942	1,235
1,108	250	1,117	1,128	1,147	922	1,025	1,134	922	1,204
1,069	300	1,076	1,107	1,113	893	988	1,068	903	1,181
962	500	1,012	1,043	1,029	829	917	1,015	827	1,081
917	600	990	993	1,008	819	889	967	816	1,074
	word-tokens	94,626	28,069	81,987	58,303	72,605	17,978	15,754	25,518
	word-types	8,294	4,274	8,778	5,541	6,754	3,116	2,735	4,322
	unseen	126	237	136	48	80	30	15	47
	low-freq.	644	920	624	288	342	109	87	210

Table 7: Test-set Perplexity across Subdomains

4.5 Test-set perplexity across domains

To study the test-set perplexity across domains, we use the NTHU's multi-domain corpus [8]. The preliminary results are shown in Table 8. Domains A and B are close to our training domain and thus have lower perplexities. Other domains have perplexities ranging from 1,079 (domain F) to 2,497 (domain N). Domain N is Kung-Fu novels and its perplexity is close to that for another collection of Kung-Fu novel. Therefore, test-set perplexity seems not only a good measure for performance, but also a candidate indicator for text categorization.

domain	wtokens	wtypes	unseen	low-freq	ud/300	ud/500	ud/600
1991ud	4,761,121	60,585	0	28,638	1,069	962	917
B: editorial	17,358	4,170	64	315	1,096	1,041	1,011
A: news	11,068	3,077	30	143	1,172	1,117	1,073
F: com.sense	2,735	834	19	61	1,162	1,131	1,079
P: romance	22,334	4,664	255	845	1,315	1,270	1,251
L: sci-fi	16,856	3,736	80	436	1,323	1,287	1,266
K: novel	43,662	7,751	468	1,695	1,658	1,614	1,604
C: review	2,097	970	8	72	1,736	1,725	1,633
G: literary	72,178	11,309	847	3,010	1,704	1,670	1,659
J: sci.tech	43,378	6,358	385	1,136	1,731	1,724	1,684
R: humor	22,284	5,338	349	1,075	1,839	1,797	1,790
D: religion	37,355	6,745	374	1,309	1,831	1,804	1,801
H: gov. doc.	13,945	2,834	44	174	2,038	1,928	1,912
E: hobby	19,080	3,770	119	478	1,964	1,953	1,927
N: kung-fu	25,420	5,272	321	1,165	2,547	2,489	2,497

Table 8: Test-set Perplexity for a Multi-domain Corpus

4.6 Test-set perplexity for word bigram

Table 9 compares word bigram models with SA-class bigram models. Four subcorpora, `day5`, `day7`, `day8`, `day9` of `1991ud` are used in the experiments. `day7` is used for training and the other three for testing. The word bigram and SA-class bigram models are trained and smoothed under the same conditions. We can observe that the difference of perplexity between the two models is approximately 1000. This proves that SA-class bigram models are indeed better trained with the same corpus.

In fact, the test-set perplexity is dependent on the size of training corpus. Table 10 shows the perplexity of `poli2` trained with the whole `1991ud` corpus. We can observe that the perplexity for word bigram is not necessarily higher than those for SA-class bigrams. However,

Model	NC(special)	day5	day8	day9
No Model		24,706	18,948	19,111
word bigram	day7/7,899(1)	2,583	2,604	2,493
word bigram	day7/7,074(6)	2,400	2,435	2,331
SA-class bigram	day7/100(6)	1,477	1,483	1,430
SA-class bigram	day7/150(6)	1,455	1,463	1,415
SA-class bigram	day7/200(6)	1,456	1,453	1,408

Table 9: Test-set Perplexity for Word Bigram

the perplexiy of SA-class bigram with appropriate number of classes is still much lower than that for word bigram even when the size of training data is as high as 4.76 million words.

Model	NC(special)	poli2
word bigram	1991ud/31,948(1)	1,119
word bigram	1991ud/30,706(6)	1,130
SA-class bigram	1991ud/200(6)	1,188
SA-class bigram	1991ud/300(6)	1,112
SA-class bigram	1991ud/500(6)	1,036
SA-class bigram	1991ud/600(6)	1,013
SA-class bigram	1991ud/900(6)	991
SA-class bigram	1991ud/1,000(6)	988

Table 10: Test-set Perplexity for Word Bigram: Larger Corpus

4.7 Test-set perplexity across character codings

We have been working on automatic word clustering of the PH corpus [12] provided by ISS, National University of Singapore. The character coding in the PH corpus is GB-code for simplified Chinese characters used in the Chinese mainland. The result is yet to be seen.

5 Concluding Remarks

We have described our experience on applying SA-class bigram to various tasks and studied the relationship between test-set perplexity and performance of language models. The experimental results show that the SA-class bigram language model is not only theoretically plausible but also practically feasible – high performance with less resource requirement. Besides, the test-set perplexity is indeed a good measure for performance evaluation of language models.

Future works include

- studying other practical issues such as alternative smoothing schemes of model parameters, stopping criteria and efficiency of SA clustering.
- investigating further the performance of word clustering across different domains and/or different applications.

Acknowledgements

Thanks are due to the Chinese Handwriting Recognition group, ATC/CCL/ITRI for the character recognizer, especially S.-S. Yu and Y.-C. Lai for preparing the recognition results. We also like to thank Prof. J.S. Chang for the multi-domain corpus. This paper is a partial result of the project no. 37H2100 conducted by the ITRI under sponsorship of the Minister of Economic Affairs, R.O.C.

References

[1] P.F. Brown, V.J. Della Pietra, P.V. de Souza, J.C. Lai, and R.L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[2] C.-H. Chang. Bidirectional conversion between Mandarin syllables and Chinese characters. In *Proc. of 1992 International Conference on Computer Processing of Chinese and Oriental Languages*, Florida, USA, 1992.

[3] C.-H. Chang. Corpus-based adaptation mechanisms for Chinese homophone disambiguation. In *Proc. of the ACL Workshop on Very Large Corpora (WVLC1)*, pages 94–101, Columbus, Ohio, USA, June 1993.

[4] C.-H. Chang. Word class discovery for contextual postprocessing of Chinese handwriting recognition. To appear in Proc. of COLING-94, August 1994.

[5] C.-H. Chang and C.-D. Chen. Automatic clustering of Chinese characters and words. In *Proc. of ROCLING VI*, pages 57–78, Chitou, Nantou, Taiwan, September 1993.

[6] C.-H. Chang and C.-D. Chen. HMM-based part-of-speech tagging for Chinese corpora. In *Proc. of the ACL Workshop on Very Large Corpora (WVLC1)*, pages 40–47, Columbus, Ohio, USA, June 1993.

[7] C.-H. Chang and C.-D. Chen. SEG-TAG: A Chinese word segmentation and part-of-speech tagging system. In *Proc. of Natural Language Processing Pacific Rim Symposium (NLPRS '93)*, pages 319–327, Fukuoka, Japan, December 1993.

[8] J.-S. Chang, S.-D. Chen, and C.-D. Chen. Conversion of phonetic-input to Chinese text through constriant satisfaction. In *Proc. 1991 ICCPCOL*, pages 30–36, Taipei, 1991.

[9] S.-L. Chou and S.-S. Yu. Sorting qualities of handwritten Chinese characters for setting up a research database. In *Proc. of ICDAR-93*, pages 474–477, Tsukuba, Japan, October 1993.

[10] K. Church. A stochastic parts program and noun phrase parser for unresticted text. In *Proc. of ICASSP-89*, pages 695–698, Glasgow, Scotland, 1989.

[11] A. Derouault and B. Merialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Trans. PAMI*, 8(5):742–749, 1986.

[12] Jin Guo. PH – a Chinese corpus. *Communications of COLIPS*, 3:45–48, 1993.

[13] M. Jardino and G. Adda. Automatic word classification using simulated annealing. In *Proc. of ICASSP-93*, pages II:41–44, Minneapolis, Minnesota, USA, 1993.

[14] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[15] L.-S. Lee et al. Golden Mandarin (II) - an improved single-chip real-time Mandarin dictation machine for Chinese language with very large vocabulary. In *Proc. of ICASSP-93*, pages II:503–506, April 1993.

[16] T.-F. Li, S.-S. Yu, H.-F. Sun, and S.-L. Chou. Handwritten Chinese character recognition using Bayes rule. In *Proc. of ICCPCOL-92*, pages 406–411, Florida, USA, December 1992.

[17] N. Metropolis et al. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[18] L.-T. Tu et al. Recognition of handprinted characters by feature matching. In *Proc. of 1991 First National Workshop on Character Recognition*, pages 166–175, Hsinchu, Taiwan, 1991.

[19] C.-H. Tung. *A Study of Handwritten Chinese Text Recognition.* PhD thesis, National Chiao-Tung University, Hsichu, Taiwan, 1994.