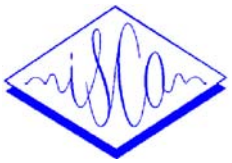

6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

ASSESSMENT OF SMOOTHING METHODS AND COMPLEX STOCHASTIC LANGUAGE MODELING

Sven Martin[1], *Christoph Hamacher, Jörg Liermann, Frank Wessel, Hermann Ney*

Lehrstuhl für Informatik VI, RWTH Aachen – University of Technology
Ahornstraße 55, 52056 Aachen, Germany

ABSTRACT

This paper studies the overall effect of language modeling on perplexity and word error rate, starting from a trigram model with a standard smoothing method up to complex state–of–the–art language models: (1) We compare different smoothing methods, namely linear vs. absolute discounting, interpolation vs. backing-off, and back-off functions based on relative frequencies vs. singleton events. (2) We show the effect of complex language model techniques by using distant-trigrams and automatically selected word classes and word phrases using a maximum likelihood criterion (i.e. minimum perplexity). (3) We show the overall gain of the combined application of the above techniques, as opposed to their separate assessment in past publications. (4) We give perplexity and word error rate results on the North American Business corpus (NAB) with a training text of about 240 million words and on the German Verbmobil corpus.

1. INTRODUCTION: LANGUAGE MODELING

In this paper, a statistical speech recognizer is based on the Bayes rule [1]:

$$w_1^N = \underset{w'^N_1}{\text{argmax}} \left\{ Pr(w'^N_1) \cdot Pr(x_1^T | w'^N_1) \right\} ,$$

which assigns the most probable N words w_1^N to an observed sequence of length T of acoustic vectors x_1^T. The language model is the approximation $p_\theta(w_1^N)$ with parameters θ of the unknown probability distribution $Pr(w_1^N)$. The parameters are estimated by Maximum Likelihood [8], i.e. by optimizing the log–likelihood function

$$F_{w_1^N}(\theta) := \log \quad p_\theta(w_1^N)$$

on a training set w_1^N. The log–likelihood function F will also be used here for the construction of the language models. Language models are either assessed by computing the perplexity [4]

$$PP := \left[p_\theta(w_1^N) \right]^{-1/N}$$

on a test set w_1^N of length N which is not part of the training set, or by performing either a recognition or a rescoring on a word lattice [12] and computing the resulting word error rate, i.e. the Levenshtein distance between the spoken and recognized words.

In this paper, the training and assessment of language models is performed on two well–known tasks with quite different properties. The first one is the large North American Business (NAB) corpus from 1994, consisting of American English newspaper texts. The second one is the small Verbmobil corpus from 1996, consisting of German spontaneous speech. Detailed information about these corpora and the respective lattices, generated by an integrated trigram search, for the rescoring experiments can be found in Tables 1 and 2. Note that the NAB test set is further subdivided into two sets named DEV and EVL. The DEV set is used for optimizing language model parameters which cannot be well estimated from the training set, such as interpolation parameters, whereas the EVL set contains unseen test data. Since the Verbmobil test set is not subdivided, a simplified form of cross–validation on the training set is used.

[1]now Philips Research Laboratories, Weißhausstraße 2, 52066 Aachen, Germany

Table 1: Words in vocabulary, test and training set.

	NAB	Verbmobil
Vocabulary	19 977 + 2	5328 + 8
Training set	240 875 674	322 588
Test set	DEV set: 7387 EVL set: 8186	6258

Table 2: Properties of the word lattices for the rescoring experiments (GER = graph error rate).

	Speakers	Sentences	GER[%]	
			del/ins	tot
NAB: DEV set	20	310	0.2/0.6	4.1
EVL set	20	316	0.1/0.9	3.8
Verbmobil	35	305	1.3/0.9	5.9

2. SMOOTHING OF TRIGRAMS

The most common language model is based on trigrams. It is constructed by applying the chain rule to the language model probability

$$p_\theta(w_1^N) = \prod_{n=1}^{N} p_\theta(w_n | w_1^{n-1}) , \qquad (1)$$

where the word history w_1^{n-1} is abbreviated as h_n, and approximating the word history h_n by the two successor words w_{n-2}, w_{n-1} of word w_n at corpus position n:

$$p_\theta(w_n | h_n) = p_\theta(w_n | w_{n-2}, w_{n-1}) .$$

Estimating the probabilities by Maximum Likelihood results into relative frequencies:

$$p_\theta(w_n | w_{n-2}, w_{n-1}) = \frac{N(w_{n-2}, w_{n-1}, w_n)}{N(w_{n-2}, w_{n-1})} , \qquad (2)$$

where $N(\cdot)$ denotes the frequency of the associated word tuple in the training set. Since each event (w_{n-2}, w_{n-1}, w_n) has its own fixed probability, the trigram model is called parameterless. For this reason, the symbol θ denoting the model parameters is useless and will be dropped for the rest of this paper.

For speech recognition tasks with large vocabularies of, say, 20 000 words, the straightforward trigram model results into

$8 \cdot 10^{12}$ trigrams, each of them with its own probability. This large number cannot be estimated from today's training sets of up to one billion words. Thus, most probabilities will be estimated by Eq. (2) as zero, which is not desired and, by the chain rule Eq. (1), estimates the whole test set as zero if such a trigram appears in it. This is known as the sparse data or zero frequency problem.

To counteract this problem, smoothing methods are applied. With smoothing, probability mass is discounted from the trigrams observed in the training set and redistributed over the remaining (unseen) trigrams according to a more robust probability distribution $\beta(w|\bar{h})$ based on a simplified word history $\bar{h}$. In case of the trigram model, the bigram probability distribution is used as $\beta(w|\bar{h})$.

A simple smoothing method is linear discounting [4] with the two variants interpolation

$$p(w|h) = (1-\lambda) \cdot \frac{N(h,w)}{N(h)} + \lambda \cdot \beta(w|\bar{h})$$

or backing–off:

$$p(w|h) = \begin{cases} (1-\lambda) \cdot \dfrac{N(h,w)}{N(h)}, & N(h,w) > 0, \\ \lambda \cdot \dfrac{\beta(w|\bar{h})}{\sum_{w':N(h,w')=0} \beta(w'|\bar{h})}, & \text{otherwise} \end{cases}$$

Since the most reliably estimated (i.e. the most frequent) events (h,w) are distracted most by using linear discounting, an alternative called absolute discounting was proposed in [10] and tested in [11]. Using interpolation, this approach results in:

$$p(w|h) = \max\left\{0, \frac{N(h,w) - d}{N(h)}\right\} + d \cdot \frac{n_+(h)}{N(h)} \cdot \beta(w|\overline{h})$$

with $n_+(h)$ as the number distinct events (h,w) observed in the training set. Absolute discounting can also be used with backing–off.

Both the interpolation parameter λ and the discounting value d would be estimated as zero on the training set because, by definition, the are no unobserved events in the training set. A method called leaving–one–out, a special case of cross–validation, is used for that task [10]: each position in the training set is visited, the event at that position dropped and the probability for that event estimated on the remaining $N-1$ events. Thus, events observed once in the training set (singletons) become unobserved. Closed–form solutions exist for both λ and d. It is even possible to derive a bigram statistics based on singleton events for the probability distribution $\beta(w|\bar{h})$ [7], called singleton back–off (BO) function.

Tables 3 and 4 show the effect of the presented smoothing methods on the NAB and Verbmobil corpora, respectively. Interpolation is superior to backing–off in most cases, absolute to linear discounting in all cases. Using the singleton back–off function further improves the results. Compared to linear discounting with interpolation as the basic method of smoothing, a reduction in perplexity of up to 21% and in word error rate of up to 9% (relative) is achieved by using absolute discounting in connection with singleton back–off. This smoothing method will be used throughout the rest of this paper.

3. WORD PHRASES

Some word pairs appear to be closely connected to each other so that they should be treated as one word. E. g. the city name "New York" consists of two words, though it is one notion. Thus, the trigram ("New", "York", w) is, in effect, just a bigram. Using word phrases, we try to find such word pairs and add them to the vocabulary. Word phrases for language modeling were first proposed by [4] and used quite often thereafter. The most recent investigation was performed by [5].

Table 3: Test set perplexities and word error rates for smoothing methods, NAB corpus.

	DEV set			EVL set		
	PP	word errs. [%]		PP	word errs. [%]	
		del/ins	WER		del/ins	WER
lin. disc.:						
backing–off	144.7	1.7/2.6	13.9	150.4	1.8/3.0	14.8
interpolation	148.8	1.5/3.1	14.1	156.5	1.5/3.4	14.8
abs. disc.:						
backing–off	125.8	1.7/2.4	13.3	127.6	1.9/2.5	13.5
interpolation	132.0	1.5/2.7	13.4	135.4	1.6/2.9	13.8
abs. disc., singleton–BO:						
backing–off	122.9	1.8/2.0	13.1	123.7	2.1/2.3	13.6
interpolation	121.8	1.8/2.0	12.8	123.4	2.0/2.3	13.6

Table 4: Test set perplexities and word error rates for smoothing methods, Verbmobil corpus.

	PP	word errs. [%]	
		del/ins	WER
linear discounting:			
backing–off	52.4	3.9/3.4	19.6
interpolation	48.2	3.3/4.1	18.9
absolute discounting:			
backing–off	43.8	3.5/3.2	18.8
interpolation	41.9	3.3/3.2	17.9
absolute discounting, singleton–BO:			
backing–off	43.9	3.6/3.0	18.3
interpolation	40.6	3.3/3.1	17.7

The word pairs are selected by optimizing a unigram–based log–likelihood criterion:

$$F = \sum_w N(w) \cdot \log\left[\frac{N(w)}{N}\right] . \qquad (3)$$

Once a new word c is formed by joining two words (a,b), we get new counts

$$\begin{aligned} \tilde{N}(a) &:= N(a) - N(a,b) \\ \tilde{N}(b) &:= N(b) - N(a,b) \\ \tilde{N}(c) &:= N(a,b) \\ w \neq a,b,c: \quad \tilde{N}(w) &:= N(w) \\ \tilde{N} &:= N - N(a,b) \end{aligned}$$

and with these new counts a new unigram log–likelihood $\tilde{F}(a,b)$ from Eq. (3). Then the difference in unigram–log–likelihood is:

$$\begin{aligned} &\Delta F(a,b) \\ &= \tilde{F}(a,b) - F \\ &= \tilde{N}(a) \cdot \log \tilde{N}(a) + \tilde{N}(b) \cdot \log \tilde{N}(b) + \tilde{N}(c) \cdot \log \tilde{N}(c) \\ &\quad -\tilde{N} \cdot \log \tilde{N} \\ &\quad -N(a) \cdot \log N(a) - N(b) \cdot \log N(b) + N \cdot \log N . \end{aligned}$$

We also tried a bigram criterion and a unigram criterion based on leaving–one–out, without much improvement. Further, it does not make much difference if we apply the criterion once for all word pairs (flat selection) or if we include the newly formed words c into the selection, thus obtaining phrases based on word triples or even more words (hierarchical selection). Another property of phrases is that there is no unique mapping from the

Table 5: Test set perplexities and word error rates for word phrases, NAB corpus, 200 phrases (flat selection).

	DEV set			EVL set		
	PP	word errs. [%]		PP	word errs. [%]	
		del/ins	WER		del/ins	WER
word trigram	121.8	1.8/2.0	12.8	123.4	2.0/2.3	13.6
phrase trigram	119.1	1.7/2.0	12.6	118.0	1.9/2.2	13.4

Table 6: Test set perplexities and word error rates for word phrases, Verbmobil corpus, 100 phrases (hierarchical selection).

	PP	word errs. [%]	
		del/ins	WER
word trigram	40.6	3.3/3.1	17.7
phrase trigram	39.5	3.4/3.0	17.2

word to the phrase sequence, because phrases may overlap. We tried several parsing strategies, based on the sum of all phrase sequences for a word sequence, on the most probable phrase sequence or on the shortest phrase sequence and found not much difference. Thus, the simplest parsing strategy, based on the shortest phrase sequence (i.e. maximum coverage of the word sequence by phrases), is used for the results.

Results are shown in Tables 5 and 6 for the NAB and Verbmobil corpus, respectively. For NAB, the optimum number of phrases in terms of word error rate was optimized on the DEV set. Adding further phrases improves perplexity but not word error rate. Similar results hold for the Verbmobil corpus. The reduction in perplexity is up to 4% and in word error rate up to 3% (relative).

4. WORD CLASSES

Using word classes, we partition the vocabulary into a fixed number of G classes, i.e. we construct a mapping function $\mathcal{G} : w \to \mathcal{G}(w)$ mapping a word w to its word class $\mathcal{G}(w)$, or g_w for short. Then we construct a class trigram statistics using the language model

$$p(w_n|w_{n-2}, w_{n-1}) = p_0(w_n|g_{w_n}) \cdot p_1(g_{w_n}|g_{w_{n-2}}, g_{w_{n-1}})$$

with the membership probability $p_0(w_n|g_{w_n})$ and the transition probability $p_1(g_{w_n}|g_{w_{n-2}}, g_{w_{n-1}})$. The advantage of this model is a reduced number of probabilities to be estimated: for, say, $G = 100$ word classes, there exist only $1\,000\,000$ class trigrams. Thus, each probability can be estimated more reliably, at the cost of a coarser model, however.

We get the classes by optimizing the bigram log–likelihood

$$F_{bi}(\mathcal{G}) = \sum_{g_v, g_w} N(g_v, g_w) \cdot \log N(g_v, g_w) - 2 \cdot \sum_{g} N(g) \cdot \log N(g) + const(\mathcal{G}) \; .$$

We have also tried the class trigram log–likelihood and a class bigram log–likelihood using leaving–one–out, but only with moderate success. The optimization is performed by the exchange algorithm depicted in Figure 1, first proposed for word clustering in [6]. By observing that removing a word w from its class g_w only affects the counts $N(g, g_w)$ and $N(g_w, g)$, that most of these counts are zero and that the same holds for adding word w to a class k, an efficient implementation of the exchange algorithm can be achieved that clusters a large number of classes

Table 7: Test set perplexities and word error rates for word classes, NAB corpus.

	G	DEV set			EVL set		
		PP	word errs. [%]		PP	word errs. [%]	
			del/ins	WER		del/ins	WER
word trigr.	—	121.8	1.8/2.0	12.8	123.4	2.0/2.3	13.6
+ class trigr.	2000	116.7	1.8/2.1	12.4	118.7	2.0/2.2	13.3
class trigr.	5000	128.9	1.8/2.1	13.0	130.6	2.2/2.4	14.0

Table 8: Test set perplexities and word error rates for word classes, Verbmobil corpus.

	G	PP	word errs. [%]	
			del/ins	WER
word trigram	—	40.6	3.3/3.1	17.7
+ class trigram	100	37.9	3.7/2.7	17.3
class trigram	500	41.8	3.6/3.0	17.7

on a large corpus within a couple of hours on a usual workstation [9]. Alternative approaches are bottom–up clustering [2] and clustering using simulated annealing [3].

Results are shown in Tables 7 and 8. Pure class models do not achieve the performance of word models. However, the linear interpolation of both models reduces the perplexity by up to 7% and the word error rate by up to 3% (relative).

5. DISTANCE TRIGRAMS

Distance trigrams are word triples with gaps between the words, i.e. the words are not consecutive. We use two types of distance trigrams: $(w_{n-3}, \cdot, w_{n-1}, w_n)$ with a gap of one word between the first two words of the trigram and $(w_{n-3}, w_{n-2}, \cdot, w_n)$ with a gap of one word between the last two words of the trigram. Thus, the interpolation with the usual trigram

$$\begin{aligned} &p(w_n|w_{n-3}, w_{n-2}, w_{n-1}) = \\ &= \lambda_1 \cdot p(w_n|w_{n-3}, \cdot, w_{n-1}) + \lambda_2 \cdot p(w_n|w_{n-3}, w_{n-2}, \cdot) \\ &\quad +(1 - \lambda_1 - \lambda_2) \cdot p(w_n|w_{n-2}, w_{n-1}) \end{aligned}$$

approximates the fourgram. Distance trigrams have been used in [13] in the context of maximum entropy, not linear interpolation.

Results can be seen in Tables 9 and 10. Due to memory limitations, singleton trigrams were dropped from the distance models for the NAB corpus. We get reductions in perplexity by up to 11% and in word error rate by up to 5%. On the small Verbmobil corpus, even the fourgram model is outperformed.

start with some initial mapping $\boldsymbol{w \to g_w}$
  for each word $\boldsymbol{w}$ of the vocabulary do
    for each class $\boldsymbol{k}$ do
      tentatively exchange word $\boldsymbol{w}$ from class $\boldsymbol{g_w}$ to class $\boldsymbol{k}$ and update counts
      compute perplexity for this tentative exchange
    exchange word $\boldsymbol{w}$ from class $\boldsymbol{g_w}$ to class $\boldsymbol{k}$ with minimum perplexity
do until stopping criterion is met

Figure 1: Exchange algorithm for word classes.

Table 9: Test set perplexities and word error rates for distance trigrams, NAB corpus.

	DEV set			EVL set		
	PP	word errs. [%]		PP	word errs. [%]	
		del/ins	WER		del/ins	WER
word trigram	121.8	1.8/2.0	12.8	123.4	2.0/2.3	13.6
+ distance trigram	109.5	1.9/1.8	12.4	110.3	2.2/2.1	13.2

Table 10: Test set perplexities and word error rates for distance trigrams, Verbmobil corpus.

	PP	word errs. [%]	
		del/ins	WER
word trigram	40.6	3.3/3.1	17.7
+ distance trigram	37.5	3.7/2.6	16.9
word fourgram	39.3	3.6/2.7	17.3

Table 11: Test set perplexities and word error rates for smoothing methods and complex language models, NAB corpus.

	DEV set			EVL set		
	PP	word errs. [%]		PP	word errs. [%]	
		del/ins	WER		del/ins	WER
lin. disc. (int.)	148.8	1.5/3.1	14.1	156.5	1.5/3.4	14.8
abs. disc. (int.)	132.0	1.5/2.7	13.4	135.4	1.6/2.9	13.8
+ singleton BO	121.8	1.8/2.0	12.8	123.4	2.0/2.3	13.6
+ word phrases	119.1	1.7/2.0	12.6	118.0	1.9/2.2	13.4
+ word classes	114.2	1.6/1.9	12.2	112.9	2.0/2.2	13.3
+ distance trigr.	105.9	1.8/1.8	11.8	104.5	2.0/2.2	13.0

6. COMBINED MODEL AND CONCLUSIONS

Tables 11 and 13 show that the improvements of each single language modeling technique are preserved in a combination of all methods. The overall reduction in perplexity is up to 33% and in word error rate up to 16% (relative). Table 12 shows the cost of this improvement in terms of memory consumption and CPU time on an Alpha 21164 processor at 533 MHz with 2 MB cache. Using distance trigrams is especially expensive because the number of word trigrams is tripled and the rescoring has to work on a longer word history.

We conclude that of all the presented techniques smoothing has the largest effect on perplexity and word error rate. Each one of the remaining techniques result in moderate or small improvements but almost further double the reduction in perplexity and word error rate in combined application.

Table 12: Memory and time consumption (RTF = real time factor) for smoothing methods and complex language models, NAB corpus.

	Memory	CPU time		RTF	
	[MBytes]	DEV	EVL	DEV	EVL
lin. disc. (int.)	214	91.6	107.9	0.03	0.03
abs. disc. (int.)	214	233.5	291.5	0.08	0.09
+ singleton BO	237	234.5	288.5	0.08	0.09
+ word phrases	263	238.1	305.8	0.08	0.09
+ word classes	403	413.7	494.2	0.15	0.15
+ distance trigr.	832	7632.5	8620.9	2.61	2.70

Table 13: Test set perplexities and word error rates for smoothing methods and complex language models, Verbmobil corpus.

	PP	word errs. [%]	
		del/ins	WER
linear discounting (int.)	48.2	3.3/4.1	18.9
absolute discounting (int.)	41.9	3.3/3.2	17.9
+ singleton BO	40.6	3.3/3.1	17.7
+ word phrases	39.5	3.4/3.0	17.2
+ word classes	36.2	3.7/2.6	16.5
+ distance trigrams	34.9	3.5/2.7	16.1

7. REFERENCES

[1] Bahl, L. R., Jelinek, F., Mercer, R. L.: "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 5, pp. 179–190, 1983.

[2] P. Brown, V. Della Pietra, P. deSouza, J. Lai, R. Mercer: "Class–Based n–gram Models of Natural Language", Computational Linguistics, Vol. 18, No. 4, pp. 467–479, 1992.

[3] M. Jardino: "Multilingual Stochastic n-Gram Class Language Models", Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, GA, pp. 161-163, May 1996.

[4] F. Jelinek: "Self-Organized Language Modeling for Speech Recognition", in: A. Waibel and K.-F. Lee (eds.): "Readings in Speech Recognition", (Morgan Kaufmann Publishers, San Mateo, CA), pp. 450-506, 1991.

[5] D. Klakow: "Language–Model Optimization by Mapping of Corpora", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Seattle, WA, Vol. II, pp. 701–704, May 1998.

[6] R. Kneser, H. Ney: "Improved Clustering Techniques for Class–Based Statistical Language Modelling", 3rd European Conference on Speech Communication and Technology, Berlin, pp. 973–976, 1993.

[7] R. Kneser, H. Ney: "Improved Backing-Off for m–gram Language Modeling", IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. I, pp. 49-52, May 1995.

[8] E. L. Lehmann: "Theory of Point Estimation", J. Wiley, New York, 1983.

[9] S. C. Martin, J. Liermann, H. Ney: "Algorithms for Bigram and Trigram Word Clustering", Speech Communication, Vol. 24, No. 1, pp. 19–37, 1998.

[10] H. Ney, U. Essen, R. Kneser: "On Structuring Probabilistic Dependences in Stochastic Language Modelling", Computer Speech and Language, Vol. 8, pp. 1–38, 1994.

[11] H. Ney, S.C. Martin, F. Wessel: "Statistical Language Modeling Using Leaving–One–Out", pp. 174–207 in: S. Young, G. Bloothooft: "Corpus–Based Methods in Language and Speech Processing", Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[12] S. Ortmanns, H. Ney, X. Aubert: "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition", Computer, Speech, and Language, Vol. 11, No. 1, pp. 43–72, Jan. 1997.

[13] R. Rosenfeld: "Adaptive Statistical Language Modeling: A Maximum Entropy Approach", Ph.D. Thesis, Technical Report CMU-CS-94-138, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 114 pages, 1994.