

Training Data Optimization for Language Model Adaptation

Xiaoshan Fang¹, Jianfeng Gao², Jianfeng Li³ and Huanye Sheng¹

¹Computer Science & Engineering Department, Shanghai Jiao Tong University, Shanghai 200030, China

²Microsoft Research Asia, Beijing 100080, China

³University of Science and Technology of China, Hefei 230027 China

fang-xs@sjtu.edu.cn

Abstract

Language model (LM) adaptation is a necessary step when the LM is applied to speech recognition. The task of LM adaptation is to use out-domain data to improve in-domain model's performance since the available in-domain (task-specific) data set is usually not large enough for LM training. LM adaptation faces two problems. One is the poor quality of the out-domain training data. The other is the mismatch between the n-gram distribution in out-domain data set and that in in-domain data set. This paper presents two methods, filtering and distribution adaptation, to solve them respectively. First, a bootstrapping method is presented to filter suitable portion from two large variable quality out-domain data sets for our task. Then a new algorithm is proposed to adjust the n-gram distribution of the two data sets to that of a task-specific but small data set. We consider preventing over-fitting problem in adaptation. All resulting models are evaluated on the realistic application of email dictation. Experiments show that each method achieves better performance, and the combined method achieves a perplexity reduction of 24% to 80%.

1 Introduction

Language model (LM) adaptation is a necessary step when the LM is applied to speech recognition applications such as email dictation (Jelinek, 1990). For example, consider that a speech engine is used for writing emails by Microsoft employee, while the language model is trained on Wall Street Journal (WSJ). Mismatch between LM and application document occurs since WSJ is too general to capture any personal writing style. Usually in-domain data (i.e. personal emails in our case) is rare. Clearly, the ideal solution would be a general language model whose parameters could be automatically adapted according to the style of text it is attempting to model.

The state-of-the-art methods of LM adaptation can be grouped roughly into three categories (Iyer, 1997; Gao, 2002; Clarkson, 1997; Seymore 1997; Gao 2000): combining data, combining counts, and combining models. Combining data is simply pumping all the corpora together, also referred to as Brute-Force (BF) approach below. Combining counts combines corpora by n-gram counts with different weights. Combining models linearly interpolates the language models trained on different data sets. In all these approaches, language model adaptation faces two problems. One is the poor quality of the out-domain training data. The other is the mismatch between the n-gram distribution in out-domain data set and that in in-domain data set.

This paper presents two methods to deal with the above two problems. They are filtering and distribution adaptation. Both methods are based on an assumption that there is a small and high quality task-specific data, call the seed set, and a large variety of data, called the training set. This assumption is reasonable. Take the email dictation task as an example. It is feasible to attain small quantity of internal employee's email and large quantity of general domain data such as the data from public email fold or MS websites. We shall show that each method brings improvement over traditional approaches, and the combined method achieves the best result in our experiment.

The remainder of this paper is organized as following: Section 2 describes the evaluation methodology. Sections 3 and 4 present the basic idea, method and results of filtering method and adaptation method separately. Finally we conclude in Section 5.

2 Evaluation methodology

2.1 Application characteristic

Email dictation is the task of writing emails by speech. It is very popular nowadays due to the increasingly emergence of handhold devices, such as Mobile phone, pocket PC. Email dictation task is individuation, by that we mean free formatting, and colloquial style. The email text has different characteristics of newspaper collection, which is the LM training corpus used in many commercial speech systems. Thus, LM adaptation is crucial for this task.

2.2 Evaluation metric and data sets

We use perplexity as our experiment metric. Perplexity is the most common metric for evaluating a language model. It can be roughly interpreted as the expected branching factor of the test document when presented to a language model. It is widely used due to its simplicity and efficiency. Although the ultimate quality of a language model should be measured by its effect on accuracy to speech recognition, we assume in this paper that lower perplexities result in lower error rates. We report the number of words which is out of vocabulary (OOV) as well for reference. It is also an important metric but is not our focus.

For our experiments, we tested our language models on a test set containing 154KB emails provided by an employee of Microsoft Research. We also collect 504KB of the same person's email as our seed set, and 104KB as held-out data. So the seed set is of high quality.

¹ This work was done while the author was visiting Microsoft Research Asia.

According to the style and domain of the task, we collect two data sets training corpora. One is Public Email (PE) which has the same style as personal email. It is collected from Microsoft Research Asia (MSRA)’s public folders of exchange server. We write a set of rules to filter junk text. For example, we delete line begins with To, CC, BCC, etc and delete all paragraphs with length lower than 100 characters. The other training corpus is from the websites www.microsoft.com (MSW). It is expected to contain documents of the same domain – Microsoft affairs. The website data is preprocessed by deleting the blocks except the content block and removing all the tags.

The PE corpus is amount to 46.2 MB. The MSW corpus is amount to 47.1 MB.

3 Training data filtering

3.1 Basic idea

In applying an SLM, it is usually the case that more training data will improve a language model. The problem is for specific domains there is always not enough high quality data to build a robust model. A large number of mixed quality data is possible to attain but blindly adding them to seed set can probably hurt the LM’s performance. It is desirable to select portions that are suitable for LM from large amount of mixed-quality out-domain data. The filtered data set is then of higher quality than the original training data and larger than the seed set.

3.2 Methodology

Our approach here is to take a small set of high-quality corpora (e.g. available application documents), called the seed set, and a large but mixed-quality corpus (e.g. data collected from the web), called the training set, and train a language model which not only satisfies the memory constraint but also has the best performance. To filter large amounts of data (e.g., data with errors) and select portions that are suitable for language modeling, we propose a new method to jointly optimize performance subject to memory requirements. The basic method has four steps: (1) segmenting training data; (2) ranking training units; (3) selecting and combining training data; and (4) pruning language models. Steps (3) and (4) are repeated until the improvement in the perplexity of the language model is less than a preset threshold.

3.2.1 Segmenting training data

The first step is to take the large training set and divide it up into units, so that we can decide whether to keep each unit and how much to trust each unit. Expanding the idea of TextFiling (Hearst 1997), we propose an algorithm to automatically segment the training data into N units, satisfying a size-range constraint while maximizing similarity within units and maximizing differences between units. It involves the following steps:

Search for available sentence boundaries and empirically cluster approximately every 30,000 words into a training chunk. We refer to the points between training chunks as gaps.

Compute the cohesion score at each gap. The cohesion score is the measure of the similarity between training blocks

(a sequence of training chunks) on both sides of the gap. Due to the limited data within each unit, our score is based on smoothed within-block term frequency (TF). Formally, the score between two training blocks, b1 and b2, is the number of terms in common in both blocks.

$$\text{Score}(b_1, b_2) = \sum_w I(w_i = w_j, w_i \in b_1, w_j \in b_2) \quad (1)$$

Where I is an indicator function such that IA=1 if A is true, and 0 otherwise, W is the vocabulary. Select the N-1 gaps with lowest cohesion scores. Each gap separates two units, and each unit has one or more chunks. We also add a size-range constraint to avoid training units that are too small or too large.

3.2.2 Ranking Training Data

The second step is to assign a score to each unit. Following our unified approach, we use perplexity as our metric (Lin, 1997). We train a language model from our seed set and measure each training data unit’s test-set perplexity against this language model. Here we use a bi-gram model, since our seed set is not large enough to train a reliable trigram. We then iteratively increase the seed model by adding blind feedback (Rocchio, 1991), which is widely used for query expansion in information retrieval. Similar to the case of information retrieval, the basic idea is that if we trust the performance of the test-set perplexity measurement, the top-ranked training units may be considered as a similar training unit set to the seed set, and can be used as a seed set as well. In practice, we augment the initial seed set with training units in the top 5-8% of N training units and then retrain the seed language model. This process is iterated until the resulting seed set is sufficient to train a robust language model. Figure 1 shows the procedure of the segmentation.

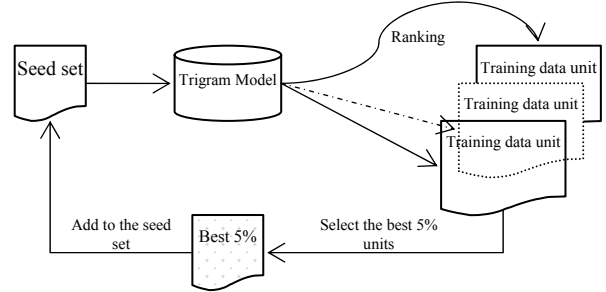


Figure 1: The Procedure of Training Data Segmenting and Ranking

3.2.3 Combining Training Data

We combine training data by interpolating the language model. Our language model interpolation algorithm involves: (1) clustering training units into N clusters; (2) training an n-gram back-off language model per cluster; and (3) interpolating all such language models into one by simple interpolation of the following form

$$P(w) = \sum_{i=1}^N \alpha_i P_i(w) \quad (2)$$

Where α_i is the interpolation weight of the ith model, and

$$\sum_{i=1}^N \alpha_i = 1$$

The interpolation weights are estimated by using the EM algorithm.

3.2.4 Pruning the Language Model

The widely used count cut-offs prune the language model by discarding n-gram counts below a certain cut-off threshold. It is, unfortunately, impossible to prune a language model to a specific size. Furthermore, in case of a combined language model, as described above, it is not known which of the original background probabilities will be useful in the combined model, so we cannot use count cut-offs.

Given a memory constraint, our system can produce a language model. We apply a relative entropy-based cut-off method (Stolcke, 1998). The basic idea is to remove as many “useless” probabilities as possible without increasing perplexity. This is achieved by examining the weighted relative entropy or Kullback-Leibler distance between each probability P and its value from the back-off distribution.

3.3 Experiments and results

3.3.1 Seed set and over-fitting

Seed set selection is a common issue in training data optimization. The seed set is defined as a set of high-quality corpora, but in practice such a corpus is not always large enough to train a reliable LM. This would lead to the over-fitting problem.

In order to solve this problem we iteratively increased the seed set by adding filtered training data until the resulting seed set is sufficient to train a robust language model. Accordingly our baseline language models were built on the combination of seed set and a portion of training data randomly selected from the large training set.

3.3.2 Results

Figure 2 displays the result of filtering MSW. We each time add the best 5% training set which is selected by the approach described in Section 3.2 to seed set for language model training. A baseline is constructed by adding randomly selected 5% training set each time to seed set. Then, we train language models with these two series of data sets, using the vocabulary DM. Finally all the language models are combined with DM and pruned to 5MB.

Figure 2 shows four curves. They are the perplexity curves from top to bottom: the pruned baseline models, the baseline models, the pruned models trained by filtered training data, and the models trained by filtered training data. The results indicate that: (1) the filtering perplexity is consistently lower than that of the baseline. (2) After filtering the perplexity increment are lower than that of the baseline. That implies that filtering can reduce the performance dropping due to language model pruning.

The results for filtering MS web data are similar except for that the best language model is attained when adding about 65% of training data to seed set.

We also have result on MS web data using a new vocabulary which combines DM’s vocabulary and a vocabulary extracted from MS web data. There is no big difference in terms of perplexity.

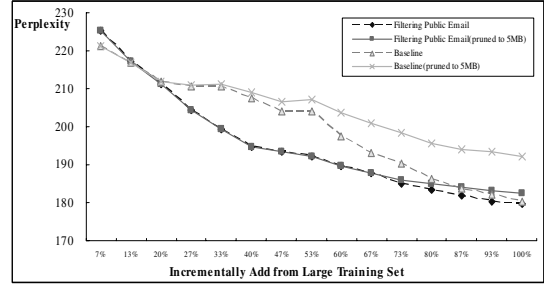


Figure2: Result on Filtering Public Email

4 Training data adapting

4.1 Basic idea

Even if we filter good data, we may want to balance it among all the training data. Previous approaches to combining training data can be grouped into two categories: combining counts, and combining models. In the former, n-gram counts from different corpora are combined with different weights. Although the method is simple and efficient, the weights are determined in an ad hoc fashion. In the latter, several models are first generated on different corpora, and then combined by linear interpolation. Interpolation weights are tuned by using EM algorithm. (Zhou, 2002)

We find that n-gram distribution characterizes domain-specific training data. In this article we propose an approach based on adapting n-gram distribution for language model training, where we adapt the language model to the domain by adjusting the n-gram distribution in the training set to that in the seed set.

4.2 Methodology

Instead of combining trigram models built on the training set and seed set, respectively, we directly combine trigram counts $C(xyz)$ of the form

$$C(xyz) = \alpha_1 \times (N_{SS} / N_{TS}) C_{TS}(xyz) + \alpha_2 C_{SS}(xyz) \quad (5)$$

$C_{SS}(xyz)$ and $C_{TS}(xyz)$ are the trigram counts of the seed set and training set respectively. N_{SS} and N_{TS} are total counts of the trigrams in the seed set and training set. We discount the n-gram counts of the training set by N_{SS}/N_{TS} for two reasons. The first reason is that we want to compress the training set to the same size with the seed set. The training set is usually much larger than the seed set and then directly combining their counts would make the final model dominated by training set. The second one is that we would like to filter the trigram counts which do not occur frequently in the training data. We do so by setting a threshold, and remove all trigrams with frequency smaller than the threshold. α_1 is the adaptation weight of the training set whereas α_2 is the weight of the seed set. α_2 is larger than α_1 since we always give more emphasis on the seed set. Extending (Gao, 2002), α_1 is estimated by

$$\alpha_1 = \begin{cases} \log \left[\frac{C_{SS}(xyz) * N_{TS} + C_{TS}(XYZ) * N_{SS}}{C_{TS}(XYZ) * N_{SS}} \right]^{\alpha_3}, & \text{while } C_{SS} \neq 0 \\ \log \left[\frac{C_{SS}(xyz) * N_{TS} + C_{TS}(XYZ) * N_{SS}}{C_{TS}(XYZ) * N_{SS}} \right]^{\alpha_4}, & \text{while } C_{SS} = 0 \end{cases} \quad (6)$$

Differently with source algorithm in (Gao, 2002), our adaptation weight α_1 have two different adaptation coefficients, α_3 and α_4 , for the condition $C_{SS}(xyz)$ is 0 and $C_{SS}(xyz)$ is not 0 respectively while the source algorithm only uses one coefficient. We find using different coefficients attains better results experimentally.

4.3 Experiments and results

4.3.1 Seed set selection and over-fitting

Except for seed set selection and over-fitting, the key point of combining count is the selection of adaptation coefficient. We use the following cross validation approach accordingly.

- Segment randomly the seed set into three units, each of the same size. Then combine two of the three units as a new seed set and the other one as a test set.
- Construct trigram models with the seed set using different group values of α_2 , α_3 and α_4 . We use the 104KB held-out data as test set and PE as training set.
- Create the other two pairs of seed set and test set, repeat the last step. Each pair of seed set and test has a group value of α_2 , α_3 and α_4 which results in the best language model. We chose the maximum of each coefficient. The coefficients' group value is 4, 17.4 and 11.2.

4.3.2 Results

We compare the overall performance on our method with the traditional BF and SI approaches in table 1.

Table 1: Result on Different Language Models

No	Training set	Technique	Pplx	Pplx reduction%	OOV%
1	YM	Baseline1	224.9		1.86
2	DM	Baseline2	174.5		0.67
3	MSW/PE	Baseline3	653.2/503.5		0.67
4	DM+MSW/PE	BF	321.2/326.5	51/35	0.67
5	YM+DM+MSW/PE	SI	171.0/171.9	74/66	0.67
6	YM+DM+MSW/PE	FABO	137.4/138.0	79/73	0.67

In table 1 we present three baselines which are produced to compare with our methods. We give some explanation here. YM is a LM which trained on 500 GB Wall Street Journal corpus. While tested on the 504 KB test set its perplexity is 224.9 and the OOV rate is 1.86%. DM is a LM trained on seed set, using a vocabulary which contains all unique words in seed set plus the vocabulary extracted from YM LM. Its result is 174.5 in perplexity and 0.67% in OOV. The third row of Table 1 gives the results of the two training set. Their perplexities are 653.2 and 503.5. Both OOV rates are 0.67%. Row four indicates the result on the combination of DM and each of MSW and PE using brute-force (BF) approach. The resulting perplexity is 321.2 and 326.5. Row five shows the result on simply interpolates (SI) Language model YM and each of the two models attained in BF approach. The perplexities are 171.0 and 171.9. The last row are the results of our approach named filtering and adaptation based training data optimization (FABO). We have 137.4 and 138.0 on perplexities. All the OOV results are 0.67% since we use a same vocabulary in all the experiments. From table two we conclude that using FABO approach attains higher performance than the BF method and SI method.

From the last three rows we see the perplexity reduction comparing with baseline3. The FABO method has the largest perplexity reduction which is 79% / 73% while using the training data YM+DM+MSW and YM+DM+PE respectively. While the perplexity reduction after using BF approach and SI approach is 74%/66% and 51%/35%, separately. We have not reported the perplexity reductions using different approaches while comparing with baseline 1 and baseline 2 since FABO approach results the lowest perplexity and the largest perplexity reduction simultaneously and then the same conclusion can be drawn that FABO method attains largest perplexity. Therefore it is more efficient than traditional BF and SI method.

We gain our best result 132.2 on perplexity after combining the two models whose perplexities are 171.0 and 171.9 respectively. Comparing with previous baselines in table 1, we conclude that after using our combine approach the performance improved from 24% to 80%.

5 Conclusions

In this paper we present two methods, training data filtering and training data distribution adaptation, to solve the poor quality problem of the out-domain training data and the mismatch problem between the n-gram distribution in out-domain data set and that in in-domain data set respectively. A bootstrapping method is presented to filter suitable portion from two large variable quality out-domain data sets and then a new algorithm is proposed to adjust the n-gram distribution of the two data sets to that of a task-specific but small data set.

Experiment indicates that our FABO method is better than the BF approach and SI approach. We attain 24% to 80% improvement on perplexity comparing with baselines.

References

1. Andreas Stolcke. 1998. Entropy-based pruning of back off language models. In Proceedings of the DARPA News Transcription and Understanding Workshop (Lansdowne, VA.), 270-274.
2. Gao, Jianfeng, Joshua Goodman, Mingjing Li and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for Chinese. ACM TALIP, 1(1): 3-33.
3. Gao, Jianfeng, Mingjing Li and Kai-Fu Lee. 2000. N-gram distribution based language model adaptation. In ICSLP 2000
4. Marti A. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. Computer Linguist. 23, 33-64.
5. Frederick Jelinek. 1990. Self-organized language modeling for speech recognition. In Readings in Speech Recognition.
6. Joseph John Rocchio 1971. Relevance feedback in information retrieval. In The SMART Retrieval System.
7. Kristie Seymore, Ronald Rosenfeld. 1997. Using story topics for language model adaptation. In EUROSPEECH-97.
8. Philip Clarkson and A.J. Robinson. 1997. Language model adaptation using mixtures and exponentially decaying cache. In ICASSP 1997.
9. Rukmini Iyer, Mari Ostendorf, and Gish, H. 1997. Using out-of-domain data to improve in-domain language models. IEEE Signal Process Letter. 4:8, Aug., 1997
10. Sung-Chien Lin, Chi-Lung Tsai, Lee-Feng Chien, Keh-Jiann Chen and Lin-shan Lee. 1997. Chinese language model adaptation based on document classification and multiple domain-specific language models. In EUROSPEECH 97
11. Zhou, Zhengyu, Jianfeng Gao, and Eric Chang. 2002. Improving language modeling by combining heterogeneous corpora. In ISCSLP02.