MAXIMUM ENTROPY BASED GENERIC FILTER FOR LANGUAGE MODEL ADAPTATION

Dong Yu, Milind Mahajan, Peter Mau, Alex Acero

Speech Research Group, Microsoft Research, Redmond WA, USA
{dongyu, milindm, petermau, alexac}@microsoft.com

ABSTRACT

Language Model (LM) Adaptation has been shown to be very important to reduce the Word Error Rate (WER) in task specific speech recognition systems. Adaptation data collected in the real world, however, usually contain large amount of non-dictated text such as email headers, long URL, code fragments, included reply, signature, etc. that the user will never dictate. Adapting with these data may corrupt the LM. In this paper, we propose a Maximum Entropy (MaxEnt) based filter to remove a variety of non-dictated words from the adaptation data and improve the effectiveness of the LM adaptation. We argue that this generic filter is language independent and efficient. We describe the design of the filter, and show that the usage of the filter can give us 10% relative WER reduction over LM adaptation without the filtering, and 22% relative WER reduction over the un-adapted LM in English email dictation task.

1. INTRODUCTION

Typical Automatic Speech Recognition (ASR) systems recognize speech based on the following criteria:

$$\hat{w} = \arg\max_{w} P(A \mid w) P(w), \tag{1}$$

where w is a word sequence candidate, $P(A \mid w)$ is the Acoustic Model (AM) probability, and $P(w)$ is the prior probability or Language Model (LM) probability. Since the decision is made on both the AM and LM scores, good estimation of LM probability is crucial to the performance of the ASR system.

Training LM, however, requires large amount of relevant data which is usually unavailable for task specific speech recognition systems. An alternative way is to use small amount of domain and/or user specific data to adapt the LM trained with huge amount of task independent data (e.g., Wall Street Journal) that is much easier to obtain. For example, we may harvest emails authored by a specific user to adapt the LM and improve the email dictation accuracy.

LM adaptation consists of four steps. First step consists of collection of task specific adaptation data which we term "harvesting". In the second or the normalization step, adaptation data are transformed into standard form. Normalization is especially important for abbreviations, date and time, and punctuations. In the third step, the adaptation data are analyzed and a task specific LM is generated. In the last step, the task specific LM is interpolated with the task independent LM. The most frequently used interpolation scheme is linear interpolation:

$$P_a(w \mid h) = \gamma P_t(w \mid h) + (1-\gamma) P_i(w \mid h), \tag{2}$$

where w is the word, h is the history, $P_a(w \mid h)$ is the adapted LM probability, $P_t(w \mid h)$ is the task specific LM probability, $P_i(w \mid h)$ is the task independent LM probability, and γ is the interpolation weight.

Much of earlier work on LM adaptation focuses on comparing adaptation algorithms and/or finding relevant data automatically ([1][2][3]). In this paper, we point out that filtering out non-dictated words from adaptation data is also very important. For example, all of the harvested email data for the user may not be useful for adapting the LM. There are parts which the user will never dictate such as email headers, long URL, code fragments, included reply, signature, foreign language text, etc. Adapting on all of the harvested data may cause significant degradation in the LM. An example of the non-dictated text is illustrated in the Table 1.

TABLE 1

AN EXAMPLE EMAIL WITH NON-DICTATED PART

Label	Email Text
Dictated	The following header is automatically generated by the email client application. Adapting LM with them may corrupt the LM.
Non-Dictated	______________________________ >From: Milind Mahajan >Sent: Wednesday, September 01, 2004 5:38 PM >To: Dong Yu >Subject: LM Adaptation

Filtering out these *non-dictated* texts is not an easy job in general. One common way of doing this is to use hand-crafted rules (e.g. regular expressions). This approach has three limitations. First, it does not generalize well to situations which we have not encountered. For example, you may have a rule to filter out Microsoft Outlook's email header, but that rule may not work with Yahoo email headers. Second, rules are usually language dependent. Porting rules from one language to another almost equals to rewriting the rules. Third, developing and testing rules are very costly.

In this paper, we propose a filtering (or sampling) approach based on Maximum Entropy (MaxEnt) classifier. This approach is language independent and efficient. We describe the design of the filter, and show that the use of the filter during LM adaptation can give us 10% relative WER reduction over the LM adaptation without filtering, and 22% relative WER reduction over the un-adapted LM in English email dictation task.

The rest of the paper is organized as follows. In the section 2, we introduce the MaxEnt based filtering model. Specifically, we describe our view of the problem, the assumptions we made, and factors we considered. In the section 3, we discuss the features which we may use for the filter, and various ways of using them in the filter. We evaluate the effectiveness of the filter with personalized email dictation task in the section 4, and conclude the paper in the section 5.

2. MAXIMUM ENTROPY BASED FILTER

While other approaches (such as [8]) might be used, we consider the filtering task as a labeling problem to segment the adaptation data into two categories:

- Category D (Dictated text): Text which should be Used for LM adaptation;
- Category N (Non-dictated text): Text which should not be used for LM adaptation.

Text is divided into a sequence of text units (such as lines). The task of filtering is thus to associate each text unit with a label D for Dictated text or N for Non-dictated text with the following criteria:

$$\left(l_1 \ldots l_n\right) = \underset{l_1 \ldots l_n}{\arg\max}\, P\left(l_1 \ldots l_n \mid t_1 \ldots t_n\right), \qquad (3)$$

where t_i is the text unit i, and l_i is the label associated with t_i.

We assume that we have a small amount of task specific labeled training data for training the filter parameters. The training data can be labeled using a rule based filter or through a manual annotation tool. Since data cleaning is just a small step in the LM adaptation process, the filter has to be very efficient, easy to develop, and easy to port to other languages. To design such a filter, we need to consider several factors such as: text unit to use, label dependency modeling across units, general form of the classification model, and the feature set used for classification. We discuss the first three factors in this section and leave the discussion of features for the next section.

2.1. Text Unit

Text is divided into a sequence of text units. Natural text units to consider are lines of text (text separated by linefeed), windows with fixed number of words per window, and sentences using a sentence breaker. The unit used in our system is a line of text.

Using line as a unit has three advantages. First, it cleanly breaks out headers, signatures, tables, ASCII art etc. Second, since the line has a visual manifestation, it is likely that people use it naturally to separate out logically related pieces. Third, using line as the unit would also make it easier to label the training data if manual annotation tool is used.

2.2. Label Dependency

In our system, we assume that the labels of text units are independent with each other given the complete sequence of text units, i.e.,

$$P\left(l_1 \ldots l_n \mid t_1 \ldots t_n\right) \approx \prod_{i=1}^{n} P\left(l_i \mid t_1 \ldots t_n\right). \qquad (4)$$

In other words, our model is state-less (in contrast to [4]). Run-time complexity of state-less models is very low since we can independently compute label probability for each text unit. We further assume that the label for a given unit depends only upon units in a surrounding window of units:

$$P\left(l_i \mid t_1 \ldots t_n\right) \approx P\left(l_i \mid t_{i-k} \ldots t_{i+k}\right), \qquad (5)$$

where k is the window size. In the limit when $k = 0$, the label depends only upon the unit itself:

$$P\left(l_i \mid t_1 \ldots t_n\right) \approx P\left(l_i \mid t_i\right). \qquad (6)$$

This is what we have used in our current system. The reason we choose state-less model with zero size window is its efficiency. We will show in the section 4 that this works well in practice.

2.3. Classification Model

In our system, we use Maximum Entropy (MaxEnt) based classification model ([4][5][6]). The principle of Maximum Entropy states that we should select the unique probability distribution which satisfies all known constraints but does not make any additional assumptions, i.e., maximizes the entropy subject to all known constraints. A MaxEnt model has the form:

$$P(l_i \mid t_i; \bar{\lambda}) \approx \frac{\exp(\bar{\lambda} \bullet \bar{f}(l_i, t_i))}{Z(t_i)}, \qquad (7)$$

where $\bar{\lambda} = \left\{\lambda_1 \ldots \lambda_m\right\}$ is the vector of model parameters, $\bar{f}(l_i, t_i)$ is the vector of features on the text unit t_i for label l_i, and $Z\left(t_i\right)$ is the normalization factor so that the probabilities sum to one.

$$Z\left(t_i\right) = \sum_{l_i} \exp(\bar{\lambda} \bullet \bar{f}(l_i, t_i)). \qquad (8)$$

Given the feature values and labels on the training set, we can obtain the values for $\bar{\lambda}$ using MaxEnt training algorithm such as the Generalized Iterative Scaling (GIS) algorithm ([7]).

Once we have the model parameters determined, we can use the model to label (or filter) the adaptation text. First, we segment the adaptation text into a sequence of text unit. We then extract features for each text unit and obtain $P(l_i \mid t_i; \bar{\lambda})$. After that, we use a simple threshold to decide whether to label the text unit as D (Dictated) or N (Non-Dictated). In other words, whether the text unit is accepted for LM adaptation is determined by:

$$l_i = \begin{cases} D, & \text{if } P(D \mid t_i; \bar{\lambda}) > P_{thresh} \\ \text{N}, & \text{otherwise} \end{cases}, \qquad (9)$$

where P_{thresh} is the threshold. We use 0.5 as the threshold in our system as we have observed that the performance does not change much if the threshold is in the range of [0.4, 0.6].

3. FEATURE SET

Feature set is one of the key factors that affect the filter performance. For our purpose, we want to find out the feature set that is inexpensive to compute, language independent, and effective enough to classify text units.

3.1. Features

We start with an exhaustive list of all easily computable features which might be useful. Since text normalization is an integral part of the LM adaptation process, we have used features which depend on normalization. The minimum set of features considering the cost-benefit trade-off is determined with experiments.

- *UnitLen*: the number of tokens in the unit. Token is determined by the word breaker. In English, this is simply determined with the space delimiter.
- *TokLen*: the average number of characters per token in the unit.
- *Norm*: the percentage of the tokens in the raw text which require normalization.
- *RawCompact*: the ratio of the number of tokens in the raw text of the unit to the number of words in the normalized text.
- *EOS*: the percentage of words in the normalized text which are end-of-sentence words.
- *OOV*: the percentage of words in the normalized text which are Out Of Vocabulary.
- *Perp*: the perplexity of the normalized text using task independent LM.
- *TgHit*: the percentage of trigrams in the normalized text which are present in the task independent LM.
- *BgHit*: the percentage of bigrams in the normalized text which are present in the task independent LM.

3.2. Space Splitting

In general, feature induction to create new features is useful to compensate for the limitations of the linear classifier. We experimented with a specific form of feature induction which is equivalent to splitting or partitioning the space of text units into sub-spaces and creating a separate classifier for each sub-space with all of the underlying features.

More specifically, in *some* of our experiments, we expand each feature of the form $f(l,t)$ into multiple features by using the length of the text unit for partitioning:

$$\begin{aligned} f_{*-4}(l,t) &= f(l,t) \bullet \delta(0 \le TokLen(t) < 4) \\ f_{4-8}(l,t) &= f(l,t) \bullet \delta(4 \le TokLen(t) < 8) \\ f_{8-16}(l,t) &= f(l,t) \bullet \delta(8 \le TokLen(t) < 16) \\ f_{16-*}(l,t) &= f(l,t) \bullet \delta(16 \le TokLen(t)) \end{aligned} \qquad (10)$$

where

$$\delta(x) = \begin{cases} 1, & if \;\; x = true \\ 0, & otherwise \end{cases} \qquad (11)$$

As a heuristic rationale for this we note that since all of the feature values are normalized by a measure of the unit length, it is possible that the feature values of smaller units are likely to be noisier than those for the longer units. Introducing features in this form, provides the model some flexibility to make allowances for this.

3.3. Feature Value: Binary or Continuous

We experimented with using the continuous feature value directly in the model or bucketing it to create a binary value. As an example, if a feature such as a percentage takes values in the range 0-100, the bucketed feature for each of the buckets such as 0-10, 10-20, takes a value 0 or 1 depending on whether the value fell in the bucket. In other words, a feature $f(l,t)$ bucketed by range [x, y) results in bucket feature:

$$f^{b}{}_{x-y}(l,t) = \delta(x \le f(l,t) < y) \qquad (12)$$

Using the continuous value results in less number of features but introduces some a-priori assumptions into the model. Use of bucketing gives us more control over the assumptions which go into the treatment of feature values.

In our experiments, we selected bucket end-point values heuristically. So, for all percentage valued features, the bucket end points are: 1, 5, 10, 20, 40, 60, 80, 90, 95, 99. This creates the buckets: 0-1, 1-5, 5-10 etc. Finer bucketing at the extremes of the range is used based upon the guess that finer granularity there is likely to be more important than say in the range between 40-60 for example.

There is a trade-off involved in bucketing. Finer granularity has the potential of giving more control. However, that needs to be balanced against noisier estimation of the parameter value given data sparseness. An alternative way for doing this automatically is to select buckets based on percentile based (such as decile) threshold values.

4. EVALUATION

Classifiers are usually evaluated using Accuracy or Receiver Operating Characteristic (ROC) curve. However, in our task, the most important criterion is the WER after adapting the LM while using our filter to classify the adaptation text.

We evaluated the effectiveness of our approach with personalized email dictation task. We used email text data from four users in our experiments. The reason we only tested with four data sets is that emails contain privacy and/or sensitive information and it's hard to collect emails from more users.

The email data from each user was first separated into the adaptation set and the test set. The test set was chosen from the most recent emails and did not overlap with any of the adaptation set. The adaptation set was semi-manually annotated with D (Dictated) and N (Non-dictated) using a rule based annotation tool. We built a MaxEnt filter for each user using the labeled adaptation data from the other three i.e. using leave-one-out strategy. Note that the filter never sees any of the data for the user being tested.

For each user, the adaptation set was first processed by the MaxEnt filter. The filtered adaptation data was then used for LM adaptation. After that, we performed ASR experiments with the adapted LM on the acoustic test data for that user which consists of read emails from the test set of that user.

Table 2 displays the information about each data set. It shows the base line WER, the number of words in each test set, as well as the number of words in each user's adaptation data. The size of the adaptation data varies from 83K words to over 3M words.

We evaluated our filter with a variety of filter configurations (different set of features and different ways of using features, as indicated in the section 3.). We found out that all the configurations we have tested gave us roughly the same results. In the following we present the detailed results for the most efficient configuration which is as follows:

- Uses only features RawCompact, EOS, and OOV (see 3.1);
- Values are bucketed to binary (see 3.3);
- No space splitting (see 3.2).

TABLE 2

INFORMATION ABOUT EACH DATA SET

User #	Base Line WER	Test Set (Wrds)	Adaptation Set (Wrds)	Adaptation Set After Filtering (Wrds)
U1	17.3 %	1974	83K	42K
U2	18.1 %	2106	451K	155K
U3	16.7 %	4836	963K	559K
U4	20.7 %	2008	2976K	1680K

Fig. 1 illustrates the recognition results of base line, LM adaptation without filtering, and LM adaptation using MaxEnt based filter. We can observe from the figure that except for U1 (which has the smallest adaptation set), using our filter during LM adaptation reduces WER against that of LM adaptation without filter. We can also see that the WER after adaptation increases over the un-adapted LM baseline for U2 if our filter is not used. On average, we get 10% relative WER reduction by using our filter during adaptation over no filtering, and 22% relative WER reduction the un-adapted LM baseline.

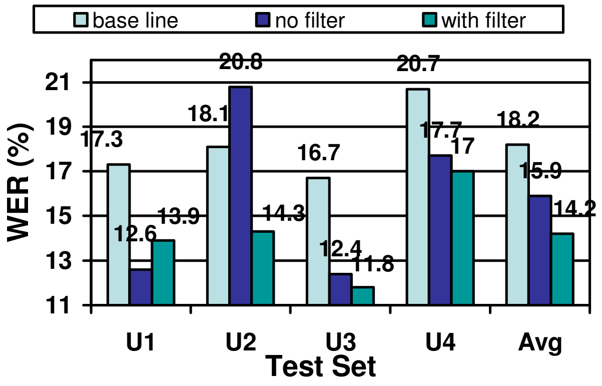

Fig. 1. Comparison of WER results between base line, LM adaptation without filter, and LM adaptation with filter. Using filter provides relative 10% and 22% WER reduction against the base line and the LM adaptation without filter condition respectively.

We have also tested the generalization ability of our filter. There is no foreign language text in the filter training data. We retrieved 27K words of Chinese text, 26K words of French text, and 26K words of Spain text from the Internet and mixed them with the adaptation data. Our results indicates that the MaxEnt based filter can successfully remove those foreign language texts without retraining. On average, the WER is 14.2% if our filter is used during adaptation, and 17.9% if it's not used.

5. CONCLUSION AND DISCUSSION

We have demonstrated that removing non-dictated text from the adaptation text is important to improving the effectiveness of LM adaptation. We described the design and trade-offs of our MaxEnt based generic filter, and showed that processing the adaptation data with our filter can effectively improve the ASR recognition.

The effects of the filtering are two folds. On the one hand, filtering out the non-dictated text would prevent the corruption of the LM. This would improve the effectiveness of the LM adaptation. On the other hand, removing some of the text from the adaptation data will decrease the size of the adaptation data. This usually causes data sparseness problem and so make the LM probability estimation less accurate and would decrease the effectiveness of the LM adaptation. The final outcome of the filtering is the combined effect of these two factors.

The above analysis is confirmed by our experiments. We have noticed that the filtering is especially important and effective for the adaptation data with high percentage of non-dictated text (e.g., U2 case) and the adaptation data with large size.

To speed up the training data preparation, we have developed an easy to use manual labeling tool. Our future work is to use the tool to generate labeled training data for other languages, and to train and evaluate the filter for these additional languages.

6. ACKNOWLEDGMENT

We want to thank our colleagues in Speech Research Group, Microsoft Research for valuable discussion and comments.

7. REFERENCES

[1] R. DeMori, & M. Federico, "*Language model adaptation*", In K. Ponting, editor, Computational Models of Speech Pattern Processing, pages 280--303. Springer Verlag, Berlin, New York, 1999.

[2] S. F. Chen, K. Seymore, & R. Rosenfeld, "*Topic Adaptation for Language Modeling Using Unnormalized Exponential Models*", in Proc. 23rd International Conf. on Acoustics, Speech, and Signal Processing, Seattle, Washington, USA, 1998.

[3] X. Fang, J. Gao, J. Li, H. Sheng, "*Training Data Optimization for Language Model Adaptation*", EUROSPEECH 2003 Geneva, Switzerland September 1-4, pp1485-1488, 2003.

[4] A. McCallum, D. Freitag, & F. Pereira, "*Maximum Entropy Markov Models for Information Extraction and Segmentation*", in Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, pp591-598, 2000.

[5] A. Berger, S. Della Pietra, & V. Della Pietra, "*A Maximum Entropy Approach to Natural Language Processing*", Computational Linguistics, 22(1):39–71, 1996.

[6] R. Rosenfeld, "*A Maximum Entropy Approach to Adaptive Statistical Language Modeling*", Computer, Speech, and Language, 10, 1996.

[7] J. N. Darroch, & D. Ratcliff, "*Generalized iterative scaling for log-linear models*", The Annals of Mathematical Statistics, 43(5), 1470–1480, 1972.

[8] D. Hakkani-T¨ur, G. Riccardi, and A. Gorin, "Active learning for automatic speech recognition," in Proceedings of the ICASSP, 2002.