META-DATA CONDITIONAL LANGUAGE MODELING

Michiel Bacchiani and Brian Roark

AT&T Labs-Research, 180 Park Ave., Florham Park, NJ 07932, USA
{michiel,roark}@research.att.com

ABSTRACT

Automatic Speech Recognition (ASR) often occurs in circumstances in which knowledge external to the speech signal, or *meta-data*, is given. For example, a company receiving a call from a customer might have access to a database record of that customer. Conditioning the ASR models directly on this information to improve the transcription accuracy is hampered because, generally, the meta-data takes on many values and a training corpus will have little data for each meta-data condition. This paper presents an algorithm to construct language models conditioned on such meta-data. It uses tree-based clustering of the the training data to automatically derive meta-data projections, useful as language model conditioning contexts. The algorithm was tested on a multiple domain voicemail transcription task. We compare the performance of an adapted system aware of the domain shift to a system that only has meta-data to infer that fact. The meta-data used were the callerID strings associated with the voicemail messages. The meta-data adapted system matched the performance of the system adapted using the domain knowledge explicitly.

1. INTRODUCTION

Adaptation of Automatic Speech Recognition (ASR) models to the operating environment and context has been shown repeatedly to be of enormous benefit [1, 2]. This is particularly true of acoustic models, where many robustness issues that arise when dealing with variations in channel and speaker characteristics have been successfully addressed by use of unsupervised self-adaptation and normalization techniques. Adaptation has also been shown to be effective for language modeling (e.g. [3, 4, 5, 6, 7, 8]), although this has been the subject of much less research to date. In light of the popularity of the Maximum Entropy (ME) algorithm [3], some language model adaptation efforts have looked at using long distance history features, such as triggers or lexico-syntactic features, and external features such as topic [4, 5, 6, 7]. Most of the adaptation efforts have focused on adapting on the test data itself (unsupervised self-adaptation) or on a sample representative of the context of interest.

Some ASR systems use contextual information, associated with but external to the speech signal such as gender or topic to shape the model distributions. This external information can be either explicitly given or inferred implicitly from the speech signal. For example, a gender dependent system might use the test data likelihood as modeled by gender dependent models to infer the speaker gender. We will refer to the external information, regardless of whether it was given or inferred, as *meta-data*.

Many applications have a large amount of meta-data available, e.g. from databases associated with the speech to be recognized. For example, a company receiving a call from a customer might have access to a database record of that customer, revealing their geographical location and product preferences. The models used in ASR systems currently do not use that type of information.

If the meta-data is sparse in the sense that it can take on few values (like gender), meta-data dependent models can be trained directly. Another approach is to adapt a meta-data independent model using the data corresponding to a specific meta-data value. However, in many scenarios the meta-data space will be large, i.e. it can take on many values. As a result, there will be little data available for a specific meta-data value, making direct conditioning infeasible. However, if the highly fragmented meta-data can be projected to a small number of conditioning contexts, it may be used to condition the ASR models. For the gender example, it might not be possible to create speaker-identity dependent models but if a projection can be found that provides clusters representing gender, conditioning on the speaker identity will be beneficial.

This paper describes an algorithm to construct meta-data conditional language models suitable for highly fragmented meta-data. It uses divisive likelihood-based clustering techniques to find projections of the meta-data, which is an obvious choice in the face of such fragmentation. A meta-data conditional model is then obtained by merging the meta-data conditional counts, weighted based on the given meta-data value. Section 2 describes the algorithm. In section 3 experimental results for a voicemail transcription task are presented. Section 4 provides conclusions and discussion.

2. ALGORITHM

The language model estimation algorithm consists of two parts. The first part is a tree clustering step which is performed at training time. The clustering result produced by the first step is then used at test time to estimate the language model used in recognition. Both steps are described in detail below.

2.1. Tree clustering

For an n-gram model of order k, the tree-clustering step involves estimating models for each order $\leq k$, beginning with the unigram model. The method used for the unigram tree is different from this higher order trees, and will be presented first. The output of the unigram clustering is used for the higher-order trees, which are built in two steps to control the greediness of the algorithm, allowing different meta-data dependencies for different histories.

Counts

The tree clustering algorithm groups history and meta-data dependent n-gram count distributions. For a given vocabulary V, let $w \in V$ denote words, and $h \in V^k$ denote a history, or conditioning sequence of k words. Let X be the set of possible meta-data values. Let $\mathcal{C}(w \mid h, x)$ denote the raw count distribution across words $w \in V$ following history $h \in V^*$, in meta-data condition $x \in X$. Then $\mathcal{C}(w \mid h) = \sum_{x \in X} \mathcal{C}(w \mid h, x)$.

The clustering algorithm uses a likelihood objective. Likelihoods are computed based on smoothed probability distribution estimates to account for unobserved events. The smoothing technique uses Good-Turing discounting [9] and Katz [10] backoff estimation to provide probability estimates from counts, in the standard way. Let $d(f)$ denote the discounting fraction for frequency

f, such that $0 < d(f) \leq 1$, where $d(f)$ is calculated once using Katz backoff estimation on the sample of meta-data independent counts[1].

The meta-data dependent count distributions are clustered by building trees in increasing Markov order, starting with unigrams. The results of the unigram tree clustering are used both for subsequent unigram probability estimation as well as for building the higher order trees.

Unigram tree

Since for the unigram count distributions $h \in V^0$ (i.e. there is no word history) , the count distributions used in building this tree are $\mathcal{C}(w \mid x)$ for all $x \in X$. The tree is built by greedily splitting the leaf that provides the largest likelihood gain. Leaf splits are evaluated using the Chou algorithm described in [11], which will be described here briefly. The evaluation of the merit of a split is an iterative process. Let $A \subseteq X$ denote the set of meta-data values that are assigned to the leaf, the split gain is evaluated as follows:

1. Estimate a parent probability distribution
$$\mathcal{C}(w \mid A) = \sum_{x \in A} \mathcal{C}(w \mid x) \tag{1}$$
$$\mathcal{C}(A) = \sum_{w \in V} \mathcal{C}(w \mid A) \tag{2}$$
$$\mathrm{P}(w \mid A) = \begin{cases} \frac{d(\mathcal{C}(w|A))\mathcal{C}(w|A)}{\mathcal{C}(A)} & \text{if } \mathcal{C}(w \mid A) > 0 \\ \delta & \text{otherwise} \end{cases} \tag{3}$$
choosing δ to ensure proper normalization and estimate the parent data log likelihood
$$\mathcal{L}_p = \sum_{w \in V} \mathcal{C}(w \mid A) \log(\mathrm{P}(w \mid A)). \tag{4}$$
Set $m = 1$.
2. Randomly partition set A into two disjoint subsets L_0 and R_0, i.e. $A = L_0 \cup R_0$ and $L_0 \cap R_0 = \emptyset$.
3. For Q in $\{L_{m-1}$,$R_{m-1}\}$ compute,
$$\mathcal{C}(w \mid Q) = \sum_{x \in Q} \mathcal{C}(w \mid x) \tag{5}$$
$$\mathcal{C}(Q) = \sum_{w \in V} \mathcal{C}(w \mid Q) \tag{6}$$
$$\mathrm{P}(w \mid Q) = \begin{cases} \frac{d(\mathcal{C}(w|Q))\mathcal{C}(w|Q)}{\mathcal{C}(Q)} & \text{if } \mathcal{C}(w \mid Q) > 0 \\ \alpha \mathrm{P}(w \mid A) & \text{otherwise} \end{cases} \tag{7}$$
where α is chosen to ensure a normalized distribution.
4. Set $L_m = R_m = \emptyset$. For each member $x \in A$, evaluate
$$\mathcal{L}_l(x) = \sum_{w \in V} \mathcal{C}(w \mid x) \log(\mathrm{P}(w \mid L_{m-1})) \tag{8}$$
and
$$\mathcal{L}_r(x) = \sum_{w \in V} \mathcal{C}(w \mid x) \log(\mathrm{P}(w \mid R_{m-1})) \tag{9}$$
and assign x to L_m if $\mathcal{L}_l(x) > \mathcal{L}_r(x)$, to R_m otherwise.
5. Compute total likelihood
$$\mathcal{L}(A) = \sum_{x \in L_m} \mathcal{L}_l(x) + \sum_{x \in R_m} \mathcal{L}_r(x). \tag{10}$$
If $m > 1$ and $\mathcal{L}(A) = \mathcal{B}$ goto 7.
6. Set $m = m + 1$. Set $\mathcal{B} = \mathcal{L}(A)$. Go to 3.
7. Set $G = \mathcal{B} - \mathcal{L}_p$. Set $L = L_m$, $R = R_m$.

[1]For frequencies 6 and higher, $d(f) = 1$

After termination of this iterative process, a partition of A into two subsets L and R is defined providing a likelihood gain G associated with that partition.

The unigram tree is built by iteratively splitting the leaf with the largest likelihood gain and evaluating the likelihood gain of the new leaves created by those splits. Once the likelihood gain of the best leaf split falls below a given threshold $T_{unigram}$, the unigram tree building step terminates. The unigram tree will then define N leaf sets, grouping the observed meta-data contexts X. The sets defined in this N-way partition will be denoted as $S = \{s_1, s_2, \cdots, s_N\}$.

Higher order trees

Like the unigram tree, the higher order ($h \in V^k$, $k > 0$) trees define a partition of the history and meta-data dependent n-gram count distributions. The count distributions used in building these trees are $\mathcal{C}(w \mid h, x)$ for all $x \in X$ and $h \in V^{n-1}$. The higher order trees, like the unigram tree, are built by greedily splitting the leaf that provides the largest likelihood gain. Unlike the unigram tree, the higher order trees are built in two stages.

1. In the first stage, the count distributions are partitioned allowing only history dependent splits. In other words, only subsets of V_{n-1} are considered, no subsets of X are allowed. As a result, all meta-data dependent occurrences of a history h are forced to fall in the same leaf. This clustering stage again uses the Chou algorithm for the evaluation of the likelihood gains from splitting leaves. Once this algorithm terminates based on a given likelihood gain threshold $T_{history}$, the leaves of the tree will contain groups of histories. We will refer to these leaves or nodes as the *history nodes* of the tree.
2. In the second stage, the leaves are split further but now only via domain splits. In other words, only partitions of X are allowed at this stage. The gain from splitting leaves is evaluated similarly to the Chou algorithm, however instead of repartitioning based on likelihood (step 4 of the Chou algorithm), only partitions based on class memberships are considered. The classes that are considered are those defined by the unigram tree leaves, i.e. S. The gains for partitioning on each class in S are computed and the partition that results in the largest likelihood increase is used if that leaf is split. Again, a given likelihood threshold T_{ngram} determines when the tree growing algorithm terminates.

After termination of the tree growing algorithm, each history $h \in V^{n-1}$ is assigned to one or more leaf nodes. The number of leaf nodes the history appears in determines the number of distinct meta-data projections for that history.

Sibling distance estimation

The final step in the the tree clustering stage of the algorithm is a distance computation. Let J denote the set of nodes containing the root node of the unigram tree and the *history nodes* of the higher order trees. For each node $k \in J$, let U_k denote the set of leaves that are descendants of k. Then for each $k \in J$

1. For each node $m \in U_k$ compute $\mathrm{P}(w \mid m)$ as in step 1 of the Chou algorithm, i.e. estimate the leaf conditional distribution based on the subset of histories and meta-data values that were assigned to that leaf.
2. For each node $m \in U_k$ compute a distance $D(m, p)$ to all $p \in U_k$. The distance is defined as the Kullback-Liebler

distance

$$D(m,p) = \sum_{w \in V} \mathrm{P}(w \mid m) \log \left(\frac{\mathrm{P}(w \mid m)}{\mathrm{P}(w \mid p)} \right). \quad (11)$$

2.2. Language Model Estimation

The language model estimation step is performed at test time using the trees designed in the training phase. It assumes a meta-data condition v is given for the test data that is to be recognized.

The language model is constructed by estimating history dependent distributions for all histories h seen in the training data. Together with the meta-data value $v \in X$, each history h identifies a leaf node $m \in U_k$, for some $k \in J$. The set of nodes $p \in U_k$ define the various meta-data projections for history h. Given the meta-data value v, m is identified as the projection applicable to the current test data. Estimating a language model on the subset of the training data represented by m will likely produce a less accurate model due to sparse data resulting from the reduction of the training set size. To prevent this, all data from all meta-data projections $p \in U_k$ are used, but weighted based on the distance from m to p, i.e. $D(m,p)$.

The history dependent distribution is estimated in a two step process.

1. Estimate the discounting factors for the observed words w based on the unweighted, summed counts

$$C(w) = \sum_{q \in U_k} \mathcal{C}(w \mid q) \quad (12)$$

$$z(w) = d(C(w)) \quad (13)$$

2. Estimate the history dependent distribution based on weighted, discounted counts

$$\mathrm{P}(w \mid h) = \begin{cases} \frac{\sum_{q \in U_k} \tau_q z(w) \mathcal{C}(w|q)}{\sum_{q \in U_k} \sum_{w \in V} \tau_q \mathcal{C}(w|q)} & \text{if } \mathcal{C}(w) > 0 \\ \gamma \mathrm{P}(w \mid h') & \text{otherwise} \end{cases} \quad (14)$$

where h' is the history h without its initial word. The value of γ is chosen to ensure proper normalization.

The count scales τ_q are derived from the distances $D(m,p)$ as

$$\tau_q = \begin{cases} \frac{1}{D(m,q)} & \text{if } \frac{1}{D(m,q)} \leq \Phi \\ \Phi & \text{otherwise} \end{cases} \quad (15)$$

where Φ is a parameter of the algorithm[2].

3. EXPERIMENTS

We evaluated the proposed algorithm in a controlled experiment. For this experiment, we set up a scenario using two corpora. The Scanmail corpus is a general voicemail corpus, described in detail in [2]. The SSNIFR corpus consists of voicemail message received at a network center customer care voicemail box, and was previously used in the work described in [8]. Although both corpora consist of voicemail messages, the language differs significantly. As shown in [8], adaptation of a language model built on Scanmail messages to the SSNIFR domain provides as much as a 7% accuracy improvement.

The experiment is controlled in the sense that we know what accuracy improvement can be obtained if it is known that the SSNIFR corpus differs in distributional characteristics to the Scanmail corpus. The question is how much of this accuracy improvement can be obtained if the data partition is not known and only a highly fragmented meta-data variable related to that partition is given. The meta-data used in this experiment are callerID strings provided by the telephone switch for every incoming message. For a subset of both corpora, this callerID information is available. Using these subsets we constructed training and test sets for both the Scanmail and SSNIFR domains using random partitions. Statistics for these sets are given in table 1. Out of the 169 CallerID strings associated with the Scanmail test messages, 136 were seen in the training set. Out of the 120 CallerID strings associated with the SSNIFR test messages, 78 were seen in the training set. There were no CallerID strings overlapping between the Scanmail and SSNIFR corpora.

Corpus name	Set type	Message count	Unique CallerID strings	Word count	Speech duration (minutes)
Scanmail	Train	6489	2072	803838	4302
Scanmail	Test	169	149	21736	114
SSNIFR	Train	300	183	33386	195
SSNIFR	Test	120	95	13164	79

Table 1. Corpus statistics.

For the experiments, three conventional Katz backoff trigram models were trained on different data sets. The model trained on the SSNIFR training set will be referred to as SSNIFR, the one trained on the Scanmail training set will be referred to as SM. The model referred to as MRG was obtained by estimation on the combined SSNIFR and SM training sets. The fourth model, referred to as MAP was obtained using the weighted count merging approach described in detail in [8]. The weight parameter was set to 5 which was empirically determined to be the optimal value. The results obtained using these models give the performance bounds using the domain information assuming this is know.

Two meta-data dependent models were trained using the proposed algorithm on the combined SSNIFR and Scanmail training sets. As meta-data the callerID strings were used, hence the size of the set X was 2255 corresponding to the number of unique callerID strings seen in the combined corpora. Both meta-data dependent models used the same threshold values $T_{history} = 1000$ and $T_{ngram} = 0$ and used the same scaling parameter $\Phi = 5$. The likelihood threshold $T_{unigram}$, however, was set to 1500 for the model referred to as SpkrS and 1000 for the model referred to as SpkrL. These models represent the scenario where the domain shift is not known but a highly fragmented meta-data variable is available possibly revealing that fact.

At test time, the meta-data dependent language models were created on a per message basis, using the callerID strings associated with the test messages. If a callerID string was not seen in the training data, the system would default to the MRG model. In the absence of any knowledge about the heterogeneity of the corpus, this would be the most appropriate model.

The run-time vs. accuracy curves showing the performance of all the models on the SSNIFR test set are given in figure 1. The performance of the SM, MRG, SpkrS and SpkrL models on the Scanmail test set are given in figure 2. It can be seen that on the SSNIFR test data, the MAP model gives an additional gain over the MRG model. Both models outperform the SSNIFR model, as reported in [8]. It also shows that the SpkrS model matches the MAP model performance. The SpkrL model does not perform as well as the SpkrS model but still provides a performance gain over the MRG model.

On the Scanmail test data, both the SM and MRG model give

[2]One can interpret this parameter as setting the maximum distance for a sibling node to be considered 'in-domain', and hence receive the same weight as the node m itself. If Φ=5, then another node must be within KL-distance 0.2 to be considered in-domain.

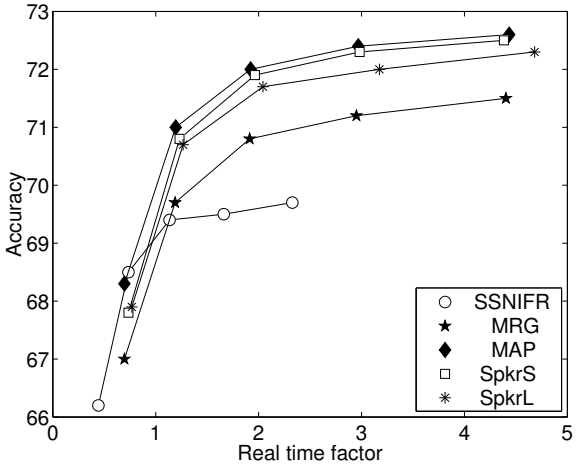

Fig. 1. Recognition performance on the SSNIFR test set.

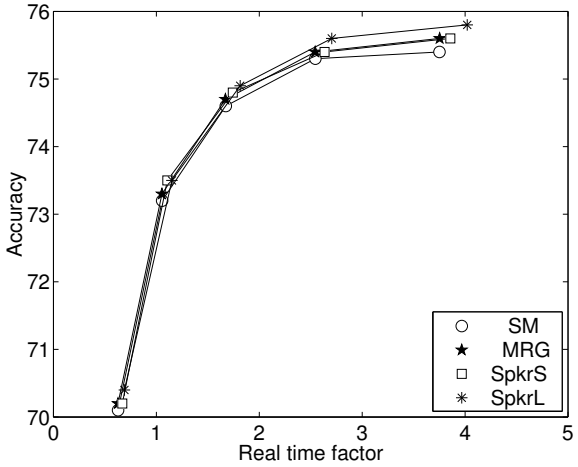

Fig. 2. Recognition performance on the Scanmail test set.

equal performance. The SpkrS and SpkrL models perform as well or show a small improvement over the baseline models.

4. CONCLUSIONS

The experimental results show that the algorithm succeeds in finding meta-data projections useful for conditioning the language model. When provided only with very fragmented meta-data and no explicit knowledge about a domain shift for a small subset of the data, the algorithm successfully created a model that matched the performance of a model adapted using the domain shift knowledge.

The weight estimation based on the inverse Kullback-Liebler distance computation appears to provide a reasonable estimate of the proximity of different meta-data projections. This is shown by the small performance difference between SpkrS and SpkrL. In the SpkrS model, 182 out of the 183 SSNIFR CallerID strings were in a single leaf of the unigram tree, along with some Scanmail messages. This means the model effectively found the SSNIFR subset from the rest of the data. In the SpkrL model, due to the lower $T_{unigram}$ parameter, that leaf was split further into 3 subsets. The resulting model performed almost as well since the distance between these subsets was found to be small and hence the SSNIFR data was weighted approximately equally even though it was partitioned into multiple subsets.

The fact that the meta-data dependency did also provide a small accuracy improvement on the Scanmail data, where little or no gain from meta-data conditioning was expected reinforces the view that the distance is appropriate.

That the SpkrS model matches the MAP performance shows that the algorithm is using the leaf distances to appropriately weight the contributions of various meta-data projections. Note that only 78 out of the 120 test messages used a meta-dependent model as the other messages defaulted to the MRG model due to a novel CallerID string.

Various aspects of the algorithm can possibly benefit from further investigation. First, the set definitions inferred from the unigram tree might not be the optimal choice for use in the higher order trees. Second, other distance and weight relationships can be considered that might give better performance. Besides algorithmic improvements, one can envision many empirical trials testing the conditioning benefit of various meta-data sources.

5. REFERENCES

[1] *Proceedings of the Speech Transcription Workshop, 2000*.

[2] M. Bacchiani, "Automatic transcription of voicemail at AT&T," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[3] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer Speech and Language*, vol. 10, pp. 187–228, 1996.

[4] K. Seymore and R. Rosenfeld, "Using story topics for language model adaptation," in *Proceedings of Eurospeech*, 1997.

[5] S. Chen, K. Seymore, and R. Rosenfeld, "Topic adaptation for language modeling using unnormalized exponential models," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998.

[6] R. Rosenfeld, S. F. Chen, and X. Zhu, "Whole-sentence exponential language models: a vehicle for linguistic-statistical integration," *Computer Speech and Language*, vol. 15, no. 1, 2001.

[7] J. Wu and S. Khudanpur, "Building a topic-dependent maximum entropy language model for very large corpora," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, pp. 777–780.

[8] M. Bacchiani and B. Roark, "Unsupervised language model adaptation," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003, pp. 224–227.

[9] I. Good, "The population frequencies of species and the estimation of population parameters," *Biometrica V*, vol. 40, no. 3,4, pp. 237–264, 1953.

[10] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recogniser," *IEEE Transactions on Acoustic, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[11] P. Chou, "Optimal partitioning for classification and regression trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 340–354, 1991.