DOMAIN ADAPTATION WITH CLUSTERED LANGUAGE MODELS

J.P. Ueberla
Forum Technology - DRA Malvern, St.Andrews Road
Malvern, Worcestershire, WR14 3PS, UK
email:ueberla@signal.dra.hmg.gb

ABSTRACT

In this paper, a method of domain adaptation for clustered language models is developed. It is based on a previously developed clustering algorithm, but with a modified optimisation criterion. The results are shown to be slightly superior to the previously published 'Fillup' method, which can be used to adapt standard n-gram models. However, the improvement both methods give compared to models built from scratch on the adaptation data is quite small (less than 11% relative improvement in word error rate). This suggests that both methods are still unsatisfactory from a practical point of view.

1. INTRODUCTION

Current large vocabulary speech recognition systems can achieve good performance on domains for which large quantities (e.g. millions of words) of textual data are available to train a language model. In real world applications, however, this is quite often not the case. The issue of language model domain adaptation is therefore of great practical importance.

One approach to tackle this problem is to try to learn from an analogy to the speaker dependence issue: current systems perform well by training speaker independent models, which can then be adapted with relatively little data from a given speaker (see [8]). Can the same approach be applied to language model adaptation?

In section 2., previous work in this area is reviewed and a rough working definition of domain is given. A method to perform domain adaptation with clustered language models is then developed (Section 3.). Experimental results to evaluate the method are given in Section 4., followed by conclusions in Section 5..

2. BACKGROUND

In order to make the description of domain adaptation more precise, a definition of domain is needed. One might be tempted to define domain in the sense of semantic topic. However, texts might differ in other aspects (e.g. style), which could still require language model adaptation. A more general definition of domain, more in line with the term sublanguage, is therefore required. According to [10], there are many different definitions of the term, but most of them seem to agree on the following characteristics of a sublanguage:

1. it is part of a natural language
2. it is of a specialised form
3. it behaves like a complete language
4. it is used in special circumstances (e.g. expert communication)
5. it is limited to a particular subject domain

Some of these points seem very useful for the concept of domain (2.4), others less so (1). What properties should an acceptable definition of domain have? The following spring to mind:

- there should be a continuum (e.g. an infinite number) of domains
- each domain may contain an infinite number of elements (e.g. documents/sentences/words)
- for a given element, one should be able to decide whether or not it belongs to a given domain
- all elements of a domain should have a common feature (which defines the domain)

This leads to the following rather wide working definition of domain and hence domain adaptation: A domain D is a (often infinite) set of documents such that each document satisfies a property PD (e.g. ‘the document deals with some aspect of law’). Given a sample S_B of domain D_B (background domain) and a sample S_A of domain D_A (target domain), the problem of language model domain adaptation is to produce a language model for D_A by using S_A and by carrying over some of the information contained in S_B.

Domain adaptation can be divided into static and dynamic domain adaptation, depending on the time scale used to perform adaptation. Dynamic adaptation tries to capture phenomena with a shorter time scale (e.g. topic shifts) and is performed on line, whereas static adaptation can be used to perform a one-time shift from one domain to another and is performed off line. Previous work has shown improvements by using both dynamic adaptation of n-gram models ([7], [5], [2], [9], [6], [13], [3], [4]) and by using static adaptation of n-gram models ([9], [6], [1], [13]). Since the ‘Fillup’ method presented in [1] gives better performance than linear interpolation, the ‘Fillup’ method is used as method of comparison for the adaptive clustering, which will be developed in the next section.
3. ADAPTIVE CLUSTERING

The task of a language model is to calculate \( p(w_i|c_i) \), the probability of the next word being \( w_i \) given the current context \( c_i \). Language models differ in the way this probability is modelled and how the context \( c_i \) is defined. A quite general model proposed in [12] makes use of a state mapping function \( S \) and a category mapping function \( G \). The idea behind the state mapping \( S : c \rightarrow s_c = S(c) \) is to assign each of the large number of possible contexts \( c \in C \) to one of a smaller number of context-equivalent states. Similarly, the category mapping \( G : w \rightarrow g_w = G(w) \) assigns each of the large number of possible words \( w \in V \) to one of a smaller number of categories (similar to parts of speech). The probability of the next word is then calculated as

\[
p(w_i|c_i) = p(G(w_i)|S(c_i)) * p(w_i|G(w_i)).
\]  

(1)

In order to determine \( S \) and \( G \) automatically, a clustering algorithm as shown in Figure 1 can be used. It is a greedy, hill-climbing algorithm that moves elements to the best available choice at any given time. Based on equation 1 and on the leaving-one-out likelihood of the model generating the training data, an optimisation criterion can be derived (see [14] for a detailed description). Let \( N(e) \) denote the number of times event \( e \) appeared in the training data, let \( B \) denote the smoothing parameter used for absolute discounting ([11]), and let \( n_0, n_1, n_4 \) denote the number of pairs \( (s, g) \) that have appeared zero, one and one or more times in the training data. The resulting optimisation criterion \( F \) (as derived in [14]) is

\[
F = \sum_{s,g:N(s,g)>1} N(s,g) \times \log(N(s,g) - 1 - B) + n_1 \times \log\left(\frac{B \times (n_4 + 1)}{(n_0 + 1)}\right) - \sum_s N(s) \times \log(N(s) - 1) - \sum_g N(g) \times \log(N(g) - 1).
\]  

(2)

The basic building block in the derivation of equation 2 is the likelihood of one event in the training corpus, as estimated from the training corpus in which this one event has been removed (leaving-one-out likelihood). The main idea behind the adaptive clustering is to use as basic building block the likelihood of one event in \( S_{Adept} \) as estimated from a linear interpolation of counts from \( S_{Back} \) and \( S_{Adept} \) from which this one event has been removed. The motivation for this is that the clustering can thus optimise the perplexity on \( S_{Adept} \) while having access to a linear combination of counts from \( S_{Back} \) and \( S_{Adept} \).

Let \( N_A(e) \) \( (N_B(e) \) denote the number of times event \( e \) appeared in \( S_{Adept} \) \( (S_{Back}) \). Define \( N_C(e) \) to be the linear interpolation of the two counts

\[
N_C(e) = \text{Round}(\lambda \times N_A(e) + (1 - \lambda) \times N_B(e))
\]  

(3)

where \( \text{Round}(x) \) returns the integer nearest to \( x \).

The only events that can contribute to the optimisation function are events that occur at least once in \( S_{Adept} \) (because, as explained above, the likelihood of \( S_{Adept} \) is taken as optimisation function). However, their probability is calculated based on the combined counts. Therefore, the smoothing has to apply to the combined counts. Define \( n_{bs,0}, n_{bs,1}, n_{bs,+} \) as the number of pairs \( (s,g) \) that have a combined count \( N_C(s,g) \) of 0, 1, and larger than 0. In order to introduce absolute discounting for the unigram estimates as well, also define \( n_{s,0}, n_{s,1}, n_{s,+} \) as the number of states \( s \) that have a combined count \( N_C(s) \) of 0, 1, and larger than 0 (similarly, define \( n_{g,0} \) etc. for the unigram estimates involving \( g \)). Changing equation 2 according to the basic idea outlined above, this leads to

\[
F_{Adept} = \sum_{s,g:N_A(s,g)>1,N_C(s,g)>1} N_A(s,g) \times \log(N_C(s,g) - 1 - B)
\]

\[
+ n_{bs,1} \times \log\left(\frac{B \times (n_{bs,+} - 1)}{(n_{bs,0} + 1)}\right)
\]

\[
- \sum_{s:N_A(s)>1,N_C(s)>1} N_A(s) \times \log(N_C(s) - 1 - B)
\]

\[
- \sum_{g:N_A(g)>1,N_C(g)>1} N_A(g) \times \log(N_C(g) - 1 - B)
\]

\[
- n_{s,1} \times \log\left(\frac{B \times (n_{s,+,1} - 1)}{(n_{s,0} + 1)}\right)
\]

\[
- n_{g,1} \times \log\left(\frac{B \times (n_{g,+,1} - 1)}{(n_{g,0} + 1)}\right).
\]

(4)

By using the same clustering algorithm as before, but with \( F_{Adept} \) instead of \( F \) as optimisation criterion, language model domain adaptation can be performed.

4. RESULTS

In order to test different adaptation methods, two textual samples \( S_{Back} \) and \( S_{Adept} \) and acoustic testing data from \( D_{Adept} \) are required. Since the WSJ domain has the associated acoustic data, it is used as \( D_{Adept} \). As \( D_{Back} \), the patent domain (PAT) was chosen, for which a large sample \( S_{Adept} \) (about 35 million words are used) is also available from the LDC as part of the Tipster database.

The recognition system is a state-of-the-art HMM-based system (continuous densities, mixtures, triphones). All experiments are based on bigram language models, either clustered (500 clusters) or backoff (singleton bigrams were ignored). The different methods evaluated were

- **BackBo**: a backoff model built on the background corpus
- **BackCl**: a clustered model built on the background corpus
<table>
<thead>
<tr>
<th>Model</th>
<th>PP</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackCI</td>
<td>955</td>
<td>49.3</td>
</tr>
<tr>
<td>BackBo</td>
<td>954</td>
<td>48.4</td>
</tr>
</tbody>
</table>

Table 1. Baseline results

<table>
<thead>
<tr>
<th>Adapt. words</th>
<th>PP</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>6130</td>
<td>57.0</td>
</tr>
<tr>
<td>1000</td>
<td>2740</td>
<td>54.0</td>
</tr>
<tr>
<td>5000</td>
<td>1740</td>
<td>47.6</td>
</tr>
<tr>
<td>25000</td>
<td>966</td>
<td>39.2</td>
</tr>
<tr>
<td>125000</td>
<td>593</td>
<td>33.0</td>
</tr>
</tbody>
</table>

Table 2. Results for AdaptBo

- AdaptCI: a clustered model built on the adaptation data
- AdaptBo: a backoff model built on the adaptation data
- Fillup: a model built according to the 'Fillup' method presented in [1]
- ClustAdapt: a model built with the adaptive clustering presented in the previous section; the initial starting point for the clustering is taken to be the clustering produced by BackCI; one global λ parameter was used and optimised iteratively at the end of each iteration;

For all methods except BackBo and BackCI, the vocabulary was defined to be all the words that appeared in SAdapt, plus additional words from SBack until 20K words were reached. For BackBo and BackCI, the vocabulary consisted of the 20K most frequent words in SBack. Because of this difference, the perplexities of BackBo and BackCI are not directly comparable to those of the other models. For each method and a given amount of adaptation material, the perplexity of the resulting model was calculated on a held-out section of SAdapt and a recognition run was performed on the acoustic data.

Table 1 gives the results of the two baseline methods, which do not use any of the adaptation material. The high perplexities show that the PAT and WSJ domains are considerably different. The rate of out-of-vocabulary words is about 15%, which is one reason for the very high error rate.

Tables 2, 3, 4 and 5 give the results for the different methods and different amounts of adaptation material.

Comparing Table 3 to Table 2, one can see that AdaptCI is more robust than AdaptBo and it leads to better recognition results for almost the entire range of adaptation material. This is consistent with previous results (see [15]), which showed that clustered models are more robust in terms of perplexity.

Comparing Table 4 to Table 3, one can see that Fillup outperforms AdaptCI in almost all cases.

<table>
<thead>
<tr>
<th>Adapt. words</th>
<th>PP</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>4170</td>
<td>57.0</td>
</tr>
<tr>
<td>1000</td>
<td>2150</td>
<td>51.1</td>
</tr>
<tr>
<td>5000</td>
<td>1210</td>
<td>46.4</td>
</tr>
<tr>
<td>25000</td>
<td>765</td>
<td>37.0</td>
</tr>
<tr>
<td>125000</td>
<td>498</td>
<td>33.4</td>
</tr>
</tbody>
</table>

Table 3. Results for AdaptCI

<table>
<thead>
<tr>
<th>Adapt. words</th>
<th>PP</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>848</td>
<td>49.9</td>
</tr>
<tr>
<td>1000</td>
<td>772</td>
<td>49.6</td>
</tr>
<tr>
<td>5000</td>
<td>628</td>
<td>44.9</td>
</tr>
<tr>
<td>25000</td>
<td>543</td>
<td>38.2</td>
</tr>
<tr>
<td>125000</td>
<td>420</td>
<td>33.0</td>
</tr>
</tbody>
</table>

Table 4. Results for Fillup

<table>
<thead>
<tr>
<th>Adapt. words</th>
<th>PP</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>941</td>
<td>50.9</td>
</tr>
<tr>
<td>1000</td>
<td>1040</td>
<td>48.1</td>
</tr>
<tr>
<td>5000</td>
<td>821</td>
<td>44.8</td>
</tr>
<tr>
<td>25000</td>
<td>801</td>
<td>38.0</td>
</tr>
<tr>
<td>125000</td>
<td>623</td>
<td>32.8</td>
</tr>
</tbody>
</table>

Table 5. Results for ClustAdapt

By looking at Table 5, one can see that ClustAdapt outperforms Fillup in almost all cases.

Finally, When comparing table 5 to table 3, one can see that the relative improvements in word error rate by using ClustAdapt instead of AdaptCI are 10.7%, 5.87%, 3.45%, -2.70% and 1.80%.

5. CONCLUSION

Compared to the success of some methods for acoustic adaptation, the results obtained here are somewhat disappointing. In particular, they seem to suggest that the improvements from the adaptation techniques compared to starting from scratch on the adaptation data become quite small when several tens of thousands of words are available. One reason for this could be the fact that the acoustic space has an underlying distance metric and thus allows the comparison of two elements. Moreover, one can specify the kind of transformations one would want the adaptation to be able to perform. Both of these points seem more difficult in the case of language model adaptation.

Even though the adaptation method for clustered language models developed in this paper gives slightly better results than the Fillup method, the accuracies obtained with the adaptive clustering and the Fillup method are still very low compared to the about 80% or more the system can achieve with a backoff unigram trained on about 40 million words of the WSJ corpus. Both adaptation methods are therefore still unsatisfactory from a practical point of view.

REFERENCES


\(^{1}\) However, it is important to note that this threshold will depend on how dissimilar the two domains are. Moreover, the more fine grained analysis for different amounts of adaptation data would be beneficial, especially since the results for 25,000 words seem to be falling somewhat outside the trend.


