IMPROVED NAME RECOGNITION WITH USER MODELING

Dong Yu, Kuansan Wang, Milind Mahajan, Peter Mau, Alex Acero
{dongyu, kuansanw, milindm, petermau, alexac}@microsoft.com

Speech Technology Group
Microsoft Research
Redmond, Washington 98052, USA

ABSTRACT

Speech recognition of names in Personal Information Management (PIM) systems is an important yet difficult task. The difficulty arises from various sources: the large number of possible names that users may speak, different ways a person may be referred to, ambiguity when only first names are used, and mismatched pronunciations. In this paper we present our recent work on name recognition with User Modeling (UM), i.e., automatic modeling of user's behavior patterns. We show that UM and our learning algorithm lead to significant improvement in the perplexity, Out Of Vocabulary rate, recognition speed, and accuracy of the top recognized candidate. The use of an exponential window reduces the perplexity by more than 30%.

1. INTRODUCTION

Personal Information Management (PIM) continues to be one of the major application areas of Automatic Speech Recognition (ASR). In PIM systems, Name Recognition (NR) is an important task. For example, one of the main features of MiPad [1] is email processing which requires email recipient recognition. Another example is voice dialing which also requires NR.

The difficulty in name recognition arises from the following four factors:

1) *The large number of names*: There are many distinct proper names in a given corporation or community. Such large perplexity will lead to a large error rate. In addition, many such names would be acoustically confusable.

2) *Different ways a person may be referred to*: Different people call other people in different ways. For example, you may call Bill Gates as Bill. Other people may call him William Gates, Bill Gates, or even Bill G. Moreover, the same person may call different people differently. For example, he/she may call Bill Gates with Bill's first name but call Steve Ballmer with Steve's last name - Ballmer.

3) *Ambiguity when only a first name is used*: There are common first names, such as David, such that even if an ASR system correctly recognizes the word "David", you may still feel frustrated browsing through all David's to find your colleague.

4) *Mismatched pronunciation*: Some names are very difficult to pronounce. Examples of such names are names with foreign origin. There are two facets to this problem. First, the pronunciation module may not generate the correct pronunciation and thus the ASR system does not recognize the name even if you pronounce the name correctly. Second, sometimes the pronunciation module gets it right but a given user may pronounce the name in a completely different way. For example, Chinese people tend to pronounce Chinese names in Chinese Pinyin which is different from what generated by the Letter-To-Sound (LTS) module.

In this paper we show that name recognition can be significantly improved by using User Modeling (UM). We define UM as modeling of user's behavior patterns. In other words, UM is automated personalization. It's the process of tailoring the application to the user's personal needs.

UM has been successfully applied to dialog systems [2-4] in the past. As Orwant [5] pointed out: "The more a computer knows about a user, the better it can serve that user." and "there are techniques for personalization that can—and should—be built into today's systems." We can improve recognition accuracy and usability if the ASR system can automatically adapt to the usage patterns of individual users. In this paper, we present our recent work in NR with UM to solve the first three of the above four problems. Although our current UM system does not address the fourth problem directly, it does substantially alleviate the dependency on correct pronunciation. Our experiments show that perplexity, OOV rate, recognition speed, and accuracy of the top recognized candidate are all improved significantly with our UM system.

The remaining of the paper is organized as follows. In Section 2, we explore the reasons for using UM in NR tasks. We present our learning algorithm in Section 3. In Section 4, experimental results are presented and discussed. We present conclusions in Section 5.

2. USER MODELING: WHY

We perceive three main reasons why UM should play an important role in the NR task:

- User's usage history contains valuable information.
- It is hard to design a good general purpose name grammar.
- Having the system adapt to the user is preferable to forcing the user to adapt to the system.

2.1. User's Usage History Contains Information

In a large corporation, such as Microsoft, there are more than 100,000 employees and mailing lists. Without knowledge about a specific user, all these employees and mailing lists are equally likely to be recipients of an email. A user's email usage data contains a large quantity of information that can greatly help reduce ambiguity in the NR task.

Our study shows that most users send emails to fewer than 500 distinct recipients (each mailing list is counted as one recipient) in a three-month period of time. Figure 1 shows the result of our survey. We gathered seven employees' email header information from their Exchange Server email storage upon their agreements. To provide a wide coverage, these employees include developers, testers, program managers, support engineers, and editors.

From Figure 1, we observe that all of these surveyed users send emails to fewer than 300 distinct recipients, and that over 80% of emails are sent to the first 100 distinct recipients. This tells us that users' email usage data contains much information that can be used to effectively reduce the perplexity in the NR task.

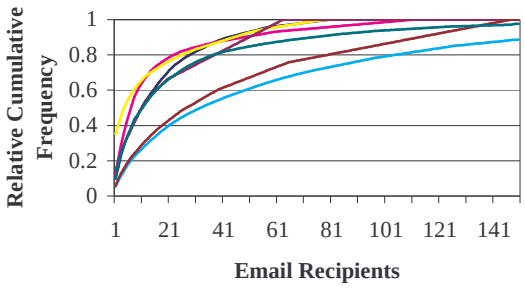

Figure 1. Relative cumulative frequency of email recipients (Each curve corresponds to one surveyed user. The X-axis indicates top *n* email recipients. The Y-axis indicates the relative cumulative frequency of top *n* email recipients).

2.2. It Is Hard to Design a Good General Purpose Name Grammar

It is very difficult to design a good grammar that includes most names without knowledge about the user. Using the general dictation statistical language model behaves poorly when dictating names.

Building a company wide name grammar can work sometimes, but has several limitations. One limitation is its incompleteness. While employees send most emails to internal recipients, they do sometimes send emails to outsiders. This is especially true for employees, such as support engineers and researchers, whose job requires networking with people outside the company. For these employees, OOV rate is high with a company-wide-only grammar. For example, one of the employees we surveyed sends more than 70% of his emails to recipients outside of Microsoft. Another limitation is its size. For a company with more than 100,000 employees and mailing lists, there will be more than 100,000 branches in the grammar. If we allow flexible names such as email aliases, first name, last name, etc, this number can easily go to 600,000. With a grammar of such a size, recognition usually becomes slow and accuracy goes down. The last limitation is the generalization of the approach. While one can pre-build such a grammar for a given company, one can not do it for home users. The name grammar for each home user has to be generated based on specific user's usage patterns.

2.3. System Instead of User Should Adapt

Instead of asking users to adapt to the ASR system, the system should and can automatically adapt to the usage patterns of individual users through UM. With UM, we can significantly reduce the size of the name grammar, leaving more space for the system to include more flexible ways of referring the same person. UM also allows the system to incrementally learn how a person is usually referred by a given user and who the most likely recipient is when ambiguities occur.

3. LEARNING ALGORITHM

Typical ASR systems recognize speech based on the following criteria:

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

where w is a candidate and $P(w)$ is the prior probability (or LM probability). UM estimates the prior probabilities based on user's usage history, and hence reduces the perplexity and increases recognition accuracy. In this paper, we propose an efficient learning algorithm. The result presented in Section 4 confirms that with our learning algorithm, UM can significantly improve the performance of the whole system.

In the following discussion, we use email recipient recognition task as an example. We estimate two probabilities in our UM for the NR task: probability of recipients and probability that a recipient is uttered in a particular way.

3.1. Probability of Recipients

The problem of estimating the probability of recipients can be described this way: given a series of recipient samples tagged with time, what's the probability of each recipient being next. In general, whether recipient i occurred at day t can be expressed as:

$$x_i(t) = \sum_k \delta(t - t_{ik}),$$

where t_{ik} indicates the time of the k^{th} occurrence the user sent an email to recipient i and δ is the Kronecker delta. The total number of times that the recipient i appears up to time T can be expressed as:

$$c_i(T) = \sum_t x_i(t)w(T-t) = \sum_k \sum_t \delta(t - t_{ik})w(T-t) = \sum_k w(T - t_{ik}),$$

where $w(t)$ is the window function applied. The probabilities of recipients thus can be estimated as:

$$\hat{p}_i = \frac{c_i(T)}{\sum_j c_j(T)} = \frac{\sum_k w(T - t_{ik})}{\sum_j \sum_k w(T - t_{jk})},$$

If the underlying stochastic process is stationary, *w(t)* should be a rectangular window, and the above estimation can be simplified to:

$$\hat{p}_i = \frac{n_i}{\sum_j n_j},$$

where n_i is the number of times recipient *i* occurs in the past.

Unfortunately, the above assumption does not hold. The probability that recipient *i* is the next recipient varies over time. Moreover, the changing patterns are different for different recipients. For example, when a user changes group, he won't send emails to the colleagues in the old team as often as before. However, he may continue to send emails to his friends very often.

To compensate for the time varying characteristic of the underlying stochastic process, an exponential window is applied:

$$w(t) = e^{-\lambda t},$$

where λ is the forgetting factor. It is so chosen that the recent data have higher weight. The larger the λ, the more weight is put on the new data. Biasing too much to new data, however, may cause an overfitting problem. In our system, λ is a slow changing parameter automatically tuned with held out set to minimize the KL distance of the held out set:

$$\hat{E} = \sum_i \bar{p}_i \log \frac{\bar{p}_i}{\hat{p}_i},$$

where $\bar{p}_i$ is estimated based on the occurrences of recipient *i* in held out set:

$$\bar{p}_i = \frac{n_i}{\sum_j n_j},$$

The system tunes λ with a gradient descent algorithm, where the gradient is

$$\frac{d\hat{E}}{d\lambda} = \frac{d\left(-\sum_i \bar{p}_i \log \hat{p}_i\right)}{d\lambda} = -\sum_i \frac{\bar{p}_i}{\hat{p}_i} \bullet \frac{d\hat{p}_i}{d\lambda}$$

$$= \sum_i \frac{\bar{p}_i}{\hat{p}_i} \bullet \left(\frac{\sum_k (T - t_{ik}) e^{-\lambda(T - t_{ik})} - \hat{p}_i \sum_j \sum_k (T - t_{jk}) e^{-\lambda(T - t_{jk})}}{\sum_j \sum_k e^{-\lambda(T - t_{jk})}} \right)$$

$$= \frac{\sum_i \frac{\bar{p}_i}{\hat{p}_i} \sum_k (T - t_{ik}) e^{-\lambda(T - t_{ik})} - \sum_j \sum_k (T - t_{jk}) e^{-\lambda(T - t_{jk})}}{\sum_j \sum_k e^{-\lambda(T - t_{jk})}},$$

So far, we only deal with recipients the user actually sent emails to. However, all people found in the user's contact list and all people who sent emails to the user are potential recipients even if the user never sent emails to them. For those people, the system provides a base probability which equals to the probability that the user sent email to the recipient once, say 100 days ago, for such a potential recipient *r*,

$$\hat{p}_r = \frac{\sum_k w(T - t_{ik})}{\sum_j \sum_k w(T - t_{jk})} = \frac{e^{-100\lambda}}{\sum_j \sum_k w(T - t_{jk})},$$

With this base probability, the effect of the potential recipients is integrated into the estimation formula we just described.

3.2. Preferred Way to Refer a Recipient

During the learning process, the system also learns the way a recipient is being referred by a specific user. To do this, the system starts with a set of rules that provide priors for all possible ways. The system then records the frequencies a specific way is used to refer each recipient and updates the probabilities accordingly in the context free grammar (PCFG). This PCFG is used by the ASR engine to recognize names.

4. EXPERIMENTS

We carried out an evaluation on the system based on the email data collected from seven Microsoft employees who were willing to share their email information (The real system, however, is used by far more users). To protect privacy, user IDs are anonymous and all names in the email headers are substituted with "name1", "name2", etc. The data span over three months for each employee, and were tagged with time as well as other information such as whether it's sent out by the employee or received from other people. We used the most recent 15% of emails sent out by the user as the test set. The most recent 10% of emails in the training set were used as the held out set to tune the forgetting factor λ whose initial value is zero. The adaptation step is initially set to 0.02 and adjusted to 90% of the old value after each iteration. The average convergence time is 17 iterations with threshold $\Delta\hat{E}$ set to 0.001.

User ID	Total Sent	Test Set	OOV (COM)	OOV (UM)	OOV (UM+COM)
user1	522	104	1.9%	0.0%	0.0%
user2	746	149	0.0%	4.7%	0.0%
user3	834	166	21.7%	3.0%	1.2%
user4	1287	257	37.0%	12.1%	7.4%
user5	127	25	32.0%	8.0%	4.0%
user6	339	67	71.6%	22.4%	17.9%
user7	2205	441	7.7%	5.2%	2.7%
average	866	173	18.4%	6.9%	3.8%

Table 1. Out-of-Vocabulary rate (OOV) for different users when using a company-wide (COM) grammar, a user-model (UM) grammar and combined grammar.

4.1. OOV

Table 1 compares the OOV for the company-wide-only email recipient grammar (COM), UM generated grammar (UM), and combined grammar (UM+COM). Note that company-wide-only grammar contains more than 100,000 email recipients while the UM generated grammar contains fewer than 500

recipients. However, as we can see from Table 1, the average OOV using company-wide-only grammar is 18.4% and the rate is 6.9% with the UM generated grammar. This is a 62% OOV rate reduction along with a 99.5% reduction on grammar size. When UM generated grammar and company-wide-only grammar are combined together, we can see additional 3.1% absolute reduction on OOV.

4.2. Perplexity

Table 2 summarizes the perplexity. With the company-wide-only grammar, the perplexity is larger than 100,000. Using the UM generated grammar, the average perplexity is 86 with the automatically tuned forgetting factor λ. This is a 99.9% reduction. With the reduction of perplexity there is a significant improvement of speed. The average time of Name Recognitions is reduced from 11ms to 5ms on a P4 2GHz computer.

From Table 2, we can also see the benefit of using an exponential window since the perplexity value is reduced from 130 with rectangular window to 86 with the exponential window. Another thing we can see is the effectiveness of our λ tuning algorithm. For most users the forgetting factor is small, which means that their usage patterns change very slowly. A special case is user6 whose forgetting factor is 0.14. From Table 1 we see that most of this users' emails are sent outside of the company. Actually, he is a support engineer who needs to contact new customers every day. Our tuning algorithm successfully catches this pattern.

User ID	Total Sent	Test Set	Perplexity (when λ=0)	λ Learned	Perplexity (learned λ)
user1	522	104	86	0.021	40
user2	746	149	48	0.052	21
user3	834	166	60	0.036	56
user4	1287	257	164	0.008	165
user5	127	25	342	0.009	172
user6	339	67	146	0.140	97
user7	2205	441	62	0.024	54
average	866	173	130	0.04	86

Table 2. Perplexity for different users using a rectangular window (λ=0) and an exponential window with the optimal forgetting factor λ.

4.3. Recognition Accuracy

We didn't conduct a thorough accuracy comparison since the accuracy improvement is so obvious in our prototype that even with strong accent or slightly incorrect pronunciation of names, accuracy is over 99% for names in the list. The average top candidate accuracy is improved from less than 10% to over 80% for normal users when using first names.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed using UM in an ASR system, i.e. the system continuously and automatically monitors user's usage pattern and models user's behavior. We use NR task as an example to demonstrate that UM can significantly decrease perplexity, OOV rate, and processing time, and can significantly increase the recognition accuracy especially when first names are used.

To successfully apply UM in NR task, we also proposed a learning algorithm to dynamically adapt the probability of each recipient to model user's changing usage pattern over time. Our experiments show that the algorithm is very effective.

Our UM can be improved in several ways and our future work will focus on these areas:

First, in our current system, we assumed that the probability associated with each person is stationary for the whole day. This, however, may not be true. People tend to send emails to different recipients at different time. For example, they may send emails mainly to colleagues during day time but send more emails to friends at night.

Another behavior that our current system does not model is the correlation among people. In our current system, the probability of each person is independent of each other. However, this is not true in most cases. For example, suppose Bob and Alice are working on the same project and frequently appear together in emails sent out by the user. If Bob is dictated in the *to* field of an email, the probability that Alice will also be in the *to* or *cc* field is very high even though Alice is not a high probability recipient in general. This behavior can be modeled through learning the correlations between recipients and dynamically adjusting probabilities based on the context.

The third area that should be improved is the handling of the OOV. As we can see from our result, the OOV rate is high for some users. These users usually change communication parties very often or do not have history data to learn from. We may improve the experience for those users with better NR or detection and correction methodologies.

6. REFERENCES

[1] X. Huang and et al, "MIPAD: A Next Generation PDA Prototype", *Proceedings of Int. Conf. on Spoken Language Processing (ICSLP) 2000*, Beijing, China, 2000.

[2] R. Kass and T. Finin, "Modeling the User in Natural Language Systems," *Computational Linguistics*, Vol. 14, No. 3, pp. 5–22, September 1988.

[3] S. Elzer, J. Chu-Carroll, and S. CarbNRy, "Recognizing and utilizing user preferences in collaborative consultation dialogues," *Proceedings of the Fourth International Conference on User Modeling*, pp. 19-24, 1994.

[4] A. Tomoyosi and H. Tanaka. "A bayesian approach for user modeling in dialogue systems," *Technical report, Dept. of Computer Scienc*e, Tokyo Inst. Technology, ISSN 0918-2802, August 1994.

[5] J. Orwant, "For want of a bit the user was lost: Cheap user modeling," *IBM Systems Journal*, Vol. 35, No. 3&4, pp. 398-416, 1996.