DESPERATELY SEEKING IMPOSTORS: DATA-MINING FOR COMPETITIVE IMPOSTOR TESTING IN A TEXT-DEPENDENT SPEAKER VERIFICATION SYSTEM

Matthieu Hébert

Nuance Communications
1380 Willow Road
Menlo Park, CA 94025
hebert@nuance.com

Nikki Mirghafori[*]

International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704
nikki@icsi.berkeley.edu

ABSTRACT

Precise determination of the operating point of a real-world verification application is of great importance. For a text-dependent password-based security system, this can be a challenging task, as lexically matched impostor test data may be non-existent. In this work we present a data mining approach for extracting suitable impostor data. The approach may be applied to either the Target database (the application data itself) or the Stock databases (data from other applications). The method entails 1) determining Levenstein distances of impostor text utterances with respect to the claimant password 2) selecting subsets of impostor data at various levels of lexical distance, 3) calculating the score threshold using such subsets, 4) extrapolating the score threshold (and hence the operating point) for lexically perfectly-matched data. Experiments on four databases in two languages are presented. This approach, as applied to the Target database, provides an accurate and inexpensive solution to a formidable real-world problem.

1. INTRODUCTION

It is crucial to both set and measure the operating point of a real-world application accurately. When a password-based security application is in the deployment phase, claimant data can be easily collected for measuring the performance. However, assuming that each user has a unique text password, lexically competitive impostor data is often non-existent, as only the true speaker speaks her/his own password. This is a real problem for real-world applications, as without lexically competitive impostor data, it is possible to neither measure nor set the operating point of the application accurately.

How can we get around this lack of impostor data? Some options may be:

* Perform a separate data collections to gather challenging, impostor attempts for a given application. Obviously, this approach is expensive and does not scale with the size of deployment.
* Use impostor score distribution from other deployments. This approach is simple, but suffers from potential mismatch in acoustic condition of the databases.
* Generate synthetic impostors synthetically [1].
* Use lexically mismatched data for impostor testing. The mismatch may be somewhat alleviated by collecting a common utterance – an utterance which all speakers speak. This utterance may be used to train the speaker model (in addition to the password) and for impostor testing. We discuss the drawbacks of this approach below.
* Mine existing data to extract tokens that have the most similar lexical content to the target utterance. This paper is dedicated to this topic.

The idea behind using a common utterance is that it would be easy to solicit speakers to say, for example, zero through nine, at enrollment time. This utterance would be used to both train the speaker model and conduct impostor testing. The reasoning is that, although the common utterance is lexically mismatched to the password, it may be more competitive than a completely mismatched utterance since the speaker model is adapted on this utterance. The following experiment elucidates the problem with mismatching data: We chose an American English digits database and trained 313 speaker models using an GMM system (details of the system are described in [4]. There were 4,833 true speaker trials and 5,730 impostor trials. The following configurations were compared:

* <u>A</u> – Lexically matched:
  Enrollment: 3 repetitions of an 8-digits password
  True speaker trial: 1 repetition of the 8-digits password
  Impostor trial: 1 repetition of the 8-digits password

* <u>B</u> – Lexically mismatched (using common utterance)
  Enrollment: 3 repetitions of an 8-digits password **plus** the common utterance
  True speaker trial: 1 repetition of the 8-digits password
  Impostor trial: 1 repetition of the common utterance

Note that the two experiments were set up such that the *(impostor, speaker model)* pairing was conserved; that is, the same impostors attempted to break into the same speaker models in both A and B, except that in setup B, they uttered the common utterance. The EER of the two systems were 6.33% and 3.06%, where system B produced a much more optimistic view of the performance than system A. To make matters worse, consider if system B is used for tuning the system and to set the threshold. At FA of 0.5%, the threshold would be set to 0.3, where, for system B the performance would be: FA = 0.5% and FR = 8.15%. However, setting

[*]The second author performed the work while at Nuance Communications.

the threshold at 0.3 for system A, the operating point would be: FA = 5.2% and FR = 7.06%. That is, the FA rate would be higher by a factor of 10. Clearly, using lexically mismatched impostor test data can create a real problem. Figure 1 shows the impostor score distribution for the two systems. We see that lexical match of the impostor utterance has a great impact on the impostor score. Even if there is no text-based ASR verification of the password, this result confirms the conventional wisdom that impostors who know use the password are more challenging.

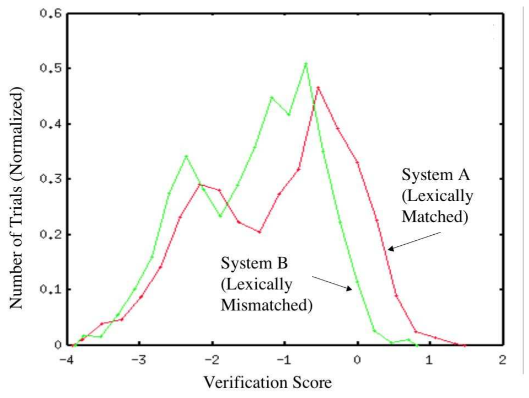

Fig. 1. The plots show the impostor score distribution for System A (lexically matched) and System B (lexically mismatched).

Clearly there is a need for a way to estimate the operating point and set thresholds using a better lexically matched impostor set. This paper proposes a data mining method to find strings with the most overlap with the target password. This method is both inexpensive and simple, and obviates the need for collecting extra information from users. In Section 2 we explain our data-mining solution. In Section 3, the results of our experiments on four databases in two languages are reported. Conclusions and future work are presented in Section 4.

2. THE APPROACH

In the previous section, we delineated the problem of using lexically mismatched impostors to determine the operating point and set thresholds in a text-dependent system. The crux of the solution that we propose is the following: since we do not have access to data that is completely lexically matched, we can calculate the score threshold for multiple impostor sets with varying degrees of lexical difference from the target password. We can then extrapolate the score threshold for lexically competitive impostor attempts. Figure 2 portrays the main idea of this paper schematically.

We use Levenstein distance to select impostor subsets which have varying distances from the targets. Our preliminary experiments were done using very simple dialogs, i.e., digits passwords, hence Levenstein distance on the word transcription level was calculated. This approach can be easily implemented for general text passwords and the distances calculated on the phonetic transcription level, using forced alignment of the word transcriptions.

In applying Levenstein distance, we have assigned a favorable bias for N-plets. The distances used in the algorithm were:

* substitution: d=2
* insertion: d=1

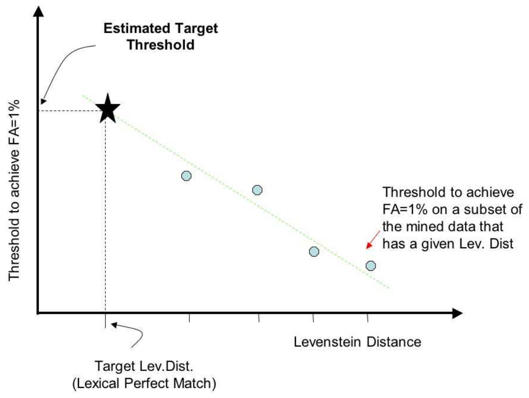
Estimated Target Threshold
Threshold to achieve FA=1%
Threshold to achieve FA=1% on a subset of the mined data that has a given Lev. Dist
Levenstein Distance
Target Lev.Dist. (Lexical Perfect Match)

Fig. 2. The figure is a schematic drawing of the main idea of this paper. Our goal is to extrapolate the score threshold for the system with lexically competitive impostors (depicted by the star).

* deletion: d=1
* 3-plet: d=-1
* 4-plet: d=-2

This assignment of weights means that we can actually have negative distances. The rationale is to bias strings that a) contain the correct digits, and b) include them in the correct order. For example, the distance between the pair (1234,1234) would be -2, for pair (1234, 4321) is 6, and for (1234, 1626364) is 3. Obviously, the smaller the distance, the higher the degree of match between the strings.

The data mining procedure is simple. For each target account in a given experiment, we calculate the Levenstein distance between the account's lexicon and the lexicon of every possible impostor attempt in our database. We can then subset the data (target account - attempts pair) based on the distance and calculate the FA rate for each subset (see Figure 2).

The next question is, what kind of data should we mine? The simple answer is: both the *Stock Database* (data from other applications) and *Target Database* (true speaker data from the application we want to tune). The argument for mining Stock data is that we may find sufficient number of lexically perfect-matched impostor tokens that we may not even have to resort to the extrapolation technique. The argument for using the Target database is that Stock data may introduce another level of unpredictable variability (channel, regional accent, and other acoustic mismatches). Experiments on both types of databases were performed and reported in the next section.

Note that the proposed technique is vaguely related to the body of work on score normalization for speaker verification systems. The closest approaches are H-Norm [3] and Z-Norm [2]. There exists a notable difference between these studies and the present: our goal is not to improve the raw performance of a speaker verification system, but rather to accurately set the operating point of that system.

3. EXPERIMENTS

We chose three American English digits databases and one UK English database for our experiments. The password length for all databases was four digits. To be able to measure the accuracy of the data mining approach, we chose datasets where we had lexically matched impostor data. Each claimant model was trained on three repetitions of the password. For impostor Stock database, we used the four digit utterances from the training corpus of the Nuance speech recognition system.

We generated multiple impostor data sets, where the impostor utterances in each set had a particular Levenstein distance from the target password. There were between 700 to 28,000 impostor trials in each subset, with an average of 12,000 trials per subset.

Figures 3 and 4 show the results generated from all the impostor subsets. On the x-axis, we see Levenstein distance and on the y-axis, the score threshold to achieve FA rate of 1%. Note that for a string length of 4, a distance of -2 constitutes a perfect match. Some data points corresponding to particular distances are absent from Figure 4, as Target databases tend to be lexically more sparse. That is, there is not a lot of lexical variety and there may not be any utterances which have a particular distance from the password string of the speaker. Overall, the trends look promising and seem to correspond to our hypothesis which was depicted in Figure 2. Next, we examine the results in more detail.

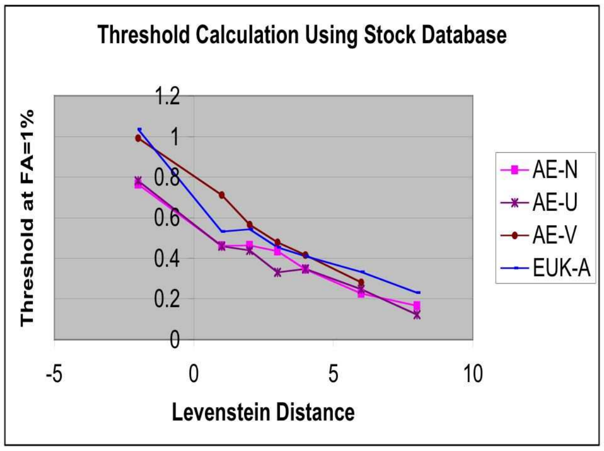

Fig. 3. Threshold estimates using impostor data mined from the Stock database. A distance of -2 constitutes a perfect match.

Earlier, we hypothesized that if we can mine the Stock databases to find impostor data which has perfect lexical match to the password, there may be no need for the extrapolation technique. Figure 5 shows the thresholds for FA=1% for lexically perfectly matched impostor utterances. The columns on the left are from the Target database (hence are the actual target thresholds) and the columns on the right are from the Stock database. If the channel conditions of the Target and Stock databases are matched, Stock data may be a good source for mining. However, if the Stock data is acoustically either more or less challenging, the thresholds may be over- or under-estimated. For example, our EA Stock database is relatively well balanced with all types of channels. EA-N Target data, however, is composed of a significant fraction of hands-free electret and cellular data. Given this mismatch, we see that using EA Stock data causes an underestimation of the Threshold for EA-N. Overall, it appears that there is a another degree of variability, which may not be precisely characterizable, if a Stock database is used.

Figure 6 shows the result of the extrapolation process when

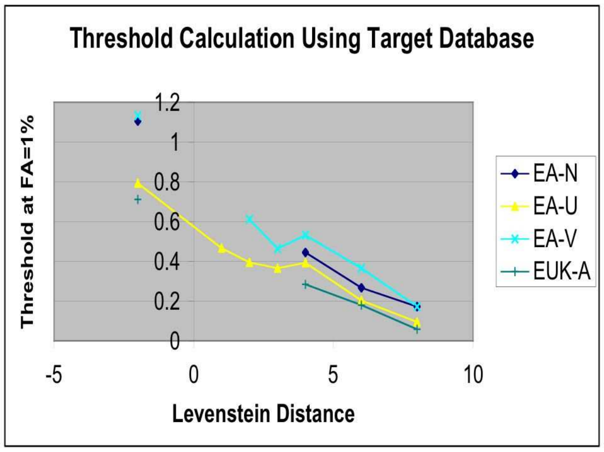

Fig. 4. Threshold estimates using impostor data mined from the Target database. A distance of -2 constitutes a perfect match.

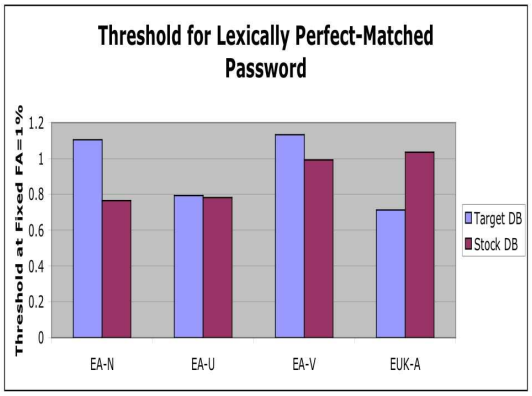

Fig. 5. Thresholds for FA=1% for lexically perfectly matched impostor utterances, from both Target and Stock databases.

mining the Target database. Using all points (except for the perfect match, distance=-2), we have estimated the threshold at distance=-2, using a linear fit to the data. The x's on the top left of the figures are the actual threshold. We see that the data mining approach consistently underestimates the actual threshold by roughly 0.1 to 0.3. Specifically, the delta between the actual threshold and the estimated threshold for the four databases shown in Figure 6, from top left to bottom right, are: (-0.27, -0.16, -0.26, -0.08). The estimate of the threshold is close and the error pattern is consistent. The technique seems viable and the consistent underestimation could be alleviated by adding a calibration factor of 0.15 to the extrapolated estimates.

For due diligence, we also extrapolated the thresholds estimates using data from the Stock databases, as shown in Figure 7. Again, using all points (except for the lexical perfect match, distance=-2), we estimated the threshold at the lexical perfect match point using a linear fit to the data. The stars on the top left of the figures are the actual target threshold and the x's on the top left are the estimate of the threshold based on lexically perfectly-matched Stock data. As we previously discussed, the lexically perfectly-matched estimates from the Stock data are sometimes

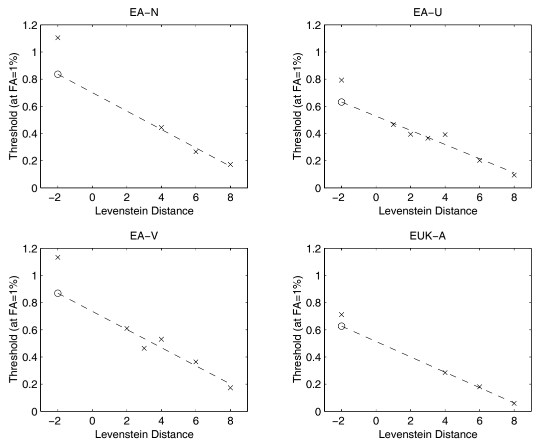

Fig. 6. The plot shows extrapolation of score threshold using data from the Target database. The estimate of the threshold is close and the error pattern is consistent.

an under- and sometimes an over-estimate of the actual threshold. However, if we compare the *extrapolated* Stock thresholds to the real threshold (x's and *'s), we see that the extrapolation is consistently below the actual threshold. The delta is larger than using Target data, as the range is between 0 and 0.5 (for Target data, the range was between 0.1 and 0.3). Both of these factors contribute to our recommendation that Stock data makes a poorer choice for data mining in this application.

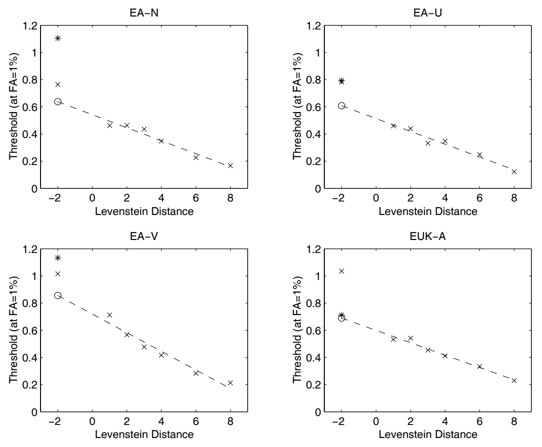

Fig. 7. The plot shows extrapolation of score threshold using data from the Stock database. The estimate of the threshold is more variable and less consistent.

Finally, we applied the extrapolated thresholds (with and without the 0.15 calibration factor) from the mining of the Target database to calculate the actual FA. Table 1 shows the effective FAs for the four tested database. The target FA was 1.0%. The Table shows the necessity of the calibration factor. The effective FAs (with the calibration factor) range from 0.8% to 1.6%, with an average of 1.2%. Considering how challenging the task of setting the operating point (thresholds) is, and recalling that the FA rates with the common utterance approach could be as much as ten times the Target FA, this result is very satisfactory.

Database	**FA** (no calibration factor)	**FA**
EA-N	2.6%	1.5%
EA-U	2.1%	1.0%
EA-V	3.0%	1.6%
EUK-A	1.5%	0.8%
Average	**2.3%**	**1.2%**

Table 1. Table shows actual FA rates for the four tested databases. The target FA is 1.0%.

4. CONCLUSIONS AND FUTURE WORK

In this work we presented a data mining approach for extracting suitable impostor data for calibrating real-world text-dependent applications. This viable solution is simple and fast, and presents an inexpensive solution to a formidable real-world problem. Although currently presented for simple dialogues (i.e., digit strings), could easily be applied to more complex ones.

Either the Target database (the application data itself) or the Stock databases (data from other applications) may be mined. The method entail 1) determining Levenstein distances of impostor text with respect to the claimant password 2) selecting subsets of impostor data at various levels of lexical distance, 3) calculating the score threshold using such data sets, and 4) extrapolating the score threshold (and hence the operating point) for lexically perfectly-matched data.

Experiments on four databases in two languages (American English and UK English) were presented. The application of this approach to Target database resulted in a simple and inexpensive way to estimate thresholds and determine the operating point for a system without lexically matched impostor data. When the data mining approach was applied to the Stock database, however, the results were less consistent due to channel and acoustic mismatches. Perhaps if the channel distribution of the Target database is maintained when mining Stock data, this variability would be reduced. This is a topic for future investigation.

5. REFERENCES

[1] D. Genoud and G. Chollet. Deliberate imposture: a challenge for automatic speaker verification systems. In *EUROSPEECH*, pages 1971–1974, 1999.

[2] M. Carey R. Auckenthaler and H. Lloyd-Thomas. Score normalization for text-independent speaker verification systems. In *Digital Signal Processing*, volume 10, pages 42–54, 2000.

[3] D.A. Reynolds. Comparison of background normalization methods for text-independent speaker verification. *EUROSPEECH*, 1997.

[4] R. Teunen, B. Shahshahani, and L.P. Heck. A model-based transformational approach to robust speaker recognition. In *ICSLP*, Bejing, China, 2000.