Name Matching Between Chinese and Roman Scripts: Machine Complements Human

Ken Samuel, Alan Rubenstein, Sherri Condon, and Alex Yeh
The MITRE Corporation; M/S H305; 7515 Colshire Drive; McLean, Virginia 22102-7508
samuel@mitre.org, rubenstein@mitre.org, scondon@mitre.org, **and** asy@mitre.org

Abstract

There are generally many ways to transliterate a name from one language script into another. The resulting ambiguity can make it very difficult to "untransliterate" a name by reverse engineering the process. In this paper, we present a highly successful cross-script name matching system that we developed by combining the creativity of human intuition with the power of machine learning. Our system determines whether a name in Roman script and a name in Chinese script match each other with an F-score of 96%. In addition, for name pairs that satisfy a computational test, the F-score is 98%.

1 Introduction

There are generally many ways to transliterate a person's name from one language script into another. For example, writers have transliterated the Arabic name, الشكري, into Roman characters in at least 13 ways, such as Al Choukri, Ash-shukri, and al-Schoukri. This ambiguity can make it very difficult to "untransliterate" a name by reverse engineering the process.

We focused on a task that is related to transliteration. Cross-script name matching aims to determine whether a given name part in Roman script matches a given name part in Chinese (Mandarin) script,[1] where a name part is a single "word" in a person's name (such as a surname), and two names match if one is a transliteration of the other.[2] Cross-script name matching has many applications, such as identity matching, improving search engines, and aligning parallel corpora.

We combine a) the creative power of human intuition, which can come up with clever ideas and b) the computational power of machine learning, which can analyze large quantities of data. Wan and Verspoor (1998) provided the human intuition by designing an algorithm to divide names into pieces that are just the right size for Roman-Chinese name matching (Section 2.2.). Armed with Wan and Verspoor's algorithm, a machine learning approach analyzes hundreds of thousands of matched name pairs to build a Roman-Chinese name matching system (Section 3).

Our experimental results are in Section 4. The system correctly determines whether a Roman name and a Chinese name match each other with F = 96.5%.[3] And F = 97.6% for name pairs that satisfy the Perfect Alignment hypothesis condition, which is defined in Section 2.2.

2 Related Work

Wan and Verspoor's (1998) work had a great impact on our research, and we explain how we use it in Section 2.2. In Section 2.1, we identify other related work.

2.1 Chinese-English Name Matching

Condon et al. (2006) wrote a paper about the challenges of matching names across Roman and Chinese scripts. In Section 6 of their paper, they offered an overview of several papers related to Roman-Chinese name matching. (Cohen et al., 2003; Gao et al., 2004; Goto et al., 2003; Jung et al., 2000; Kang and Choi, 2000; Knight and Graehl, 1997; Kondrak, 2000; Kondrak and Dorr, 2004; Li et al., 2004; Meng et al., 2001; Oh

[1] In this paper, we often use the word "Roman" to refer to "Roman script", and similarly, "Chinese" usually stands for "Chinese script".

[2] Sometimes a third script comes between the Roman and Chinese versions of the name. For example, a Roman name might be transliterated into Arabic, which is then transliterated into Chinese, or an Arabic name could be transliterated into Roman and Chinese independently.

[3] F stands for F-score, which is a popular evaluation metric. (Andrade et al., 2009)

Proceedings of the 2009 Named Entities Workshop, ACL-IJCNLP 2009, pages 152–160,
Suntec, Singapore, 7 August 2009. ©2009 ACL and AFNLP

Roman Characters:	Albertson
Phonemes:	AE,L,B,ER,T,S,AH,N
Syllables:	AEL,BERT,SAHN
Subsyllable Units:	AE,L,BER,T,SAHN
Chinese:	阿尔贝特松
Chinese Phonemes:	/a/,/ər/,/pei/,/t^hə/,/suŋ/

Table 1. Subsyllable Units

and Choi, 2006; Virga and Khudanpur, 2003; Wellner et al., 2005; Winkler, 2002)

The Levenshtein algorithm is a popular way to compute string edit distance. (Levenshtein, 1966) It can quantify the similarity between two names. However, this algorithm does not work when the names are written in different scripts. So Freeman et al. (2006) developed a strategy for Roman-Arabic string matching that uses equivalence classes of characters to normalize the names so that Levenshtein's method can be applied. Later, Mani et al. (2006) transformed that system from Roman-Arabic to Roman-Chinese name matching and extended the Levenshtein approach, attaining F = 85.2%. Then when they trained a machine learning algorithm on the output, the performance improved to F = 93.1%

Mani et al. also tried applying a phonological alignment system (Kondrak, 2000) to the Roman-Chinese name matching task, and they reported an F-score of 91.2%. However, when they trained a machine learning approach on that system's output, the F-score was only 90.6%.

It is important to recognize that it would be inappropriate to present a side-by-side comparison between Mani's work and ours (F = 96.5%), because there are many differences, such as the data that was used for evaluation.

2.2 Subsyllable Units

Transliteration is usually based on the way names are pronounced.[4] However, each character in a Roman name generally corresponds to a single phoneme, while a Chinese character (CC) generally corresponds to a subsyllable unit (SSU). A phoneme is the smallest meaningful unit of sound, and a subsyllable unit is a sequence of one to three phonemes that conform to the following three constraints. (Wan and Verspoor, 1998)

(1) There is exactly one vowel phoneme.[5]
(2) At most, one consonant phoneme may precede the vowel phoneme.
(3) The vowel phoneme may be followed by, at most, one nasal phoneme.[6]

Consider the example in Table 1. The name "Albertson" consists of eight phonemes in three syllables.[7] The last syllable, SAHN, satisfies the definition of SSU, and the other two are split into smaller pieces, resulting in a total of five SSUs. There are also five CCs in the Chinese version, 阿尔贝特松. We note that the fourth and sixth rows in the table show similarities in their pronunciations. For example, the first SSU, AE, sounds like the first CC, /a/. And, although the sounds are not always identical, such as BER and /pei/, Wan and Verspoor claimed that these SSU-CC correspondences can be generalized in the following way:

Perfect Alignment (PA) hypothesis

If a Roman name corresponds to a sequence of n SSUs, S_1, S_2, ..., S_n, and the Chinese form of that name is a sequence of n CCs, C_1, C_2, ..., C_n, then C_i matches S_i for all $1 \le i \le n$.

In Section 4, we show that the PA hypothesis works very well. However, it is not uncommon to have more SSUs than CCs in a matching name pair, in which case, the PA hypothesis does not apply. Often this happens because an SSU is left out of the Chinese transliteration, perhaps because it is a sound that is not common in Chinese. For example, suppose "Carlberg" (KAA, R,L,BER,G) is transliterated as 卡尔贝里. In this example, the SSU, R, does not correspond to any of the CCs. We generalize this phenomenon with another hypothesis:

SSUs Deletion (SSUD) hypothesis

If a Roman name corresponds to a sequence of n+k SSUs (k>0), S_1, S_2, ..., S_{n+k}, and the Chinese form of that name is a sequence of n CCs, C_1, C_2, ..., C_n, then, for some set of k S_i's, if those SSUs are removed from the sequence of SSUs, then the PA hypothesis holds.

And in the case where the number of CCs is greater than the number of SSUs, we make the

[4] Of course, there are exceptions. For example, when a name happens to be a word, sometimes that name is *translated* (rather than transliterated) into the other language. But our experimental results suggest that the exceptions are quite rare.

[5] Wan and Verspoor treat the phoneme, /ər/, as in Albertson, as a vowel phoneme.

[6] The nasal phonemes are /n/ and /ŋ/, as in "nothing".

[7] To represent phonemes, we use two different standards in this paper. The symbols between slashes (like /ər/) are in the IPA format (International Phonetic Association, 1999). And the phonemes written in capital letters (like ER) are in the ARPABET format (Klatt, 1990).

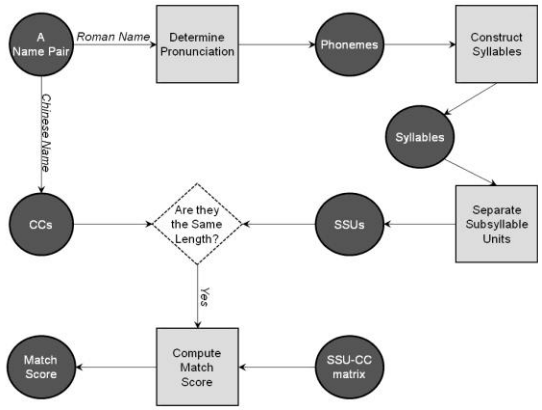

Figure 1. Application Mode

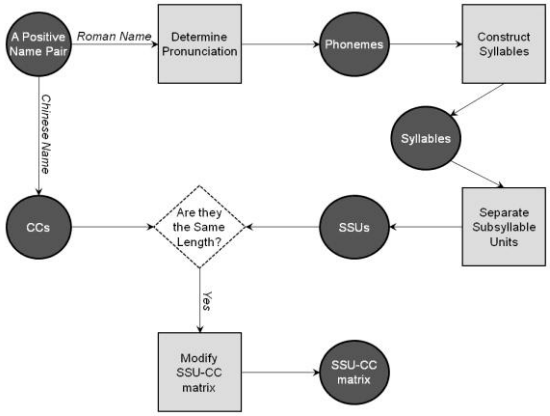

Figure 2. Training Mode

corresponding CCs Deletion (CCD) hypothesis. In the next section, we show how we utilize these hypotheses.

3 Machine Learning

We designed a machine learning algorithm to establish a mapping between SSUs and CCs. In Section 3.1, we show how our system can do Roman-Chinese name matching, and then we present the training procedure in Section 3.2.

3.1 Application Mode

Given a Roman-Chinese name pair, our system computes a match score, which is a number between 0 and 1 that is meant to represent the likelihood that two names match. This is accomplished via the process presented in Figure 1.

Starting in the upper-left node of the diagram with a Roman name and a Chinese name, the system determines how the Roman name should be pronounced by running it through the Festival system. (Black et al., 1999) Next, two algorithms designed by Wan and Verspoor (1998) join the phonemes to form syllables and divide the syllables into SSUs.[8] If the number of SSUs is equal to the number of characters in the Chinese name,[9] we apply the PA hypothesis to align each SSU with a CC.

The system computes a match score using a data structure called the SSU-CC matrix (subsyllable unit – Chinese character matrix), which has a nonnegative number for each SSU-CC pair, and this value should represent the strength of the correspondence between the SSU and the CC. Table 2 shows an example of an SSU-CC matrix. With this matrix, the name pair <Albert, 阿尔贝特> receives a relatively high match score, because the SSUs in Albert are AE, L, BER, and T, and the numbers in the SSU-CC matrix for <AE,阿>, <L,尔>, <BER,贝> and <T,特> are 2, 2, 3, and 2, respectively.[10] Alternatively, the system assigns a very low match score to <Albert, 尔贝特阿>, because the values of <AE,尔>, <L,贝>, <BER,格>, and <T,阿> are all 0.

	AE	BER	EH	G	KAA	L	LAHN	LIY	NAH	R	SAHN	T
伦	0	0	0	0	0	0	1	0	0	0	0	0
利	0	0	0	0	0	0	0	1	0	0	0	0
卡	0	0	0	0	1	0	0	0	0	0	0	0
叶	0	0	1	0	0	0	0	0	0	0	0	0
埃	0	0	1	0	0	0	0	0	0	0	0	0
娜	0	0	0	0	0	0	0	0	1	0	0	0
尔	**0**	0	0	0	0	**2**	0	0	0	1	0	0
松	0	0	0	0	0	0	0	0	0	0	1	0
特	0	**0**	0	0	0	0	0	0	0	0	0	**2**
贝	0	**3**	0	0	0	**0**	0	0	0	0	0	0
连	0	0	0	0	0	0	1	0	0	0	0	0
里	0	0	0	1	0	0	0	0	0	0	0	0
阿	**2**	0	0	0	0	0	0	0	0	0	0	**0**

Table 2. SSU-CC Matrix #1

3.2 Training Mode

To generate an SSU-CC matrix, we train our system on a corpus of Roman-Chinese name pairs

[8] This procedure passes through three separate modules, each of which introduces errors, so we would expect the system to suffer from compounding errors. However, the excellent evaluation results in Section 4 suggest otherwise. This may be because the system encounters the same kinds of errors during training that it sees in the application mode, so perhaps it can learn to compensate for them.

[9] Section 3.3 discusses the procedure used when these numbers are not equal.

[10] The equation used to derive the match score from these values can be found in Section 5.

Example #	1	2	3	4	5
Roman Characters	Albert	Albertson	Carly	Elena	Ellenberg
Subsyllable Units	AE,L,BER,T	AE,L,BER,T,SAHN	KAA,R,LIY	EH,LAHN,NAH	EH,LAHN,BER,G
Chinese Characters	阿尔贝特	阿尔贝特松	卡尔利	叶连娜	埃伦贝里

Table 3. Training Data

that match. Figure 2 shows a diagram of the training system. The procedure for transforming the Roman name into a sequence of SSUs is identical to that presented in Section 3.1. Then, if the number of SSUs is the same as the number of CCs,[9] we apply the PA hypothesis to pair the SSUs with the CCs. For example, the third name pair in Table 3 has three SSU-CC pairs: <KAA, 卡>, <R,尔>, and <LIY,利>. So the system modifies the SSU-CC matrix by adding 1 to each cell that corresponds to one of these SSU-CC pairs. Training on the five name pairs in Table 3 produces the SSU-CC matrix in Table 2.

3.3 Imperfect Alignment

The system makes two passes through the training data. In the first pass, whenever the PA hypothesis does not apply to a name pair (because the number of SSUs differs from the number of CCs), that name pair is skipped.

Then, in the second pass, the system builds another SSU-CC matrix. The procedure for processing each name pair that satisfies the PA hypothesis's condition is exactly the same as in the first pass (Section 3.2). But the other name pairs require the SSUD hypothesis or the CCD hypothesis to delete SSUs or CCs. For a given Roman-Chinese name pair:

CCs	Score	Scaled Score
Ø 卡尔贝里	0.00	0.00
卡 Ø 尔贝里	0.90	0.54
卡尔 Ø 贝里	0.76	0.46
卡尔贝 Ø 里	0.00	0.00
卡尔贝里 Ø	0.00	0.00

Table 4. Subsyllable Unit Deletion

```
For every d in D:
   Temporarily make the deletions in d.
   Evaluate the resulting name pair with Matrix #1.
Scale the evaluation scores of the d's to sum to 1.
For every d in D:
   Temporarily make the deletions in d.
   For every SSU-CC pair, ssu-cc, in the result:
      Add d's scaled score to cell [ssu,cc] in Matrix #2.
```

where D is the set of all deletion sets that make the PA hypothesis applicable. Note that the size of D grows exponentially as the difference between the number of SSUs and CCs grows.

As an example, consider adding the name pair <Carlberg, 卡尔贝里> to the data in Table 3. Carlberg has five SSUs: KAA,R,L,BER,G, but 卡尔贝里 has only four CCs. So the PA hypothesis is not applicable, and the system ignores this name pair in the first pass. Table 2 shows the values in Matrix #1 when it is completed.

In the second pass, we must apply the SSUD hypothesis to <Carlberg, 卡尔贝里> by deleting one of the SSUs. There are five ways to do this, as shown in the five rows of Table 4. (For instance, the last row represents the case where G is deleted — the SSU-CC pairs are <KAA,卡>, <R,尔>, <L,贝>, <BER,里>, and <G,Ø>.[11])

	Ø	BER	G	KAA	L	R	…
Ø		0.00	0.00	0.00	0.46	0.54	
卡	0.00	0.00	0.00	2.00	0.00	0.00	
尔	0.00	0.00	0.00	0.00	2.54	1.46	
贝	0.00	4.00	0.00	0.00	0.00	0.00	
里	0.00	0.00	2.00	0.00	0.00	0.00	
…							

Table 5. SSU-CC Matrix #2

Each of the five options are evaluated using the values in Matrix #1 (Table 2) to produce the scores in the second column of Table 4. Then the

[11] The Ø represents a deleted SSU. We include a row and column named Ø in Matrix #2 to record values for the cases in which the SSUs and CCs are deleted.

	LAHN (BER)	LAHN (NAH)	BER (G)	BER (T)
伦	**1**	0	0	0
贝	0	0	**1**	**2**
连	0	**1**	0	0

Table 6. Considering Context

system scales the scores to sum to 1, as shown in the third column, and it uses those values as weights to determine how much impact each of the five options has on the second matrix. Table 5 shows part of Matrix #2.

In application mode, when the system encounters a name pair that does not satisfy the PA hypothesis's condition it tries all possible deletion sets and selects the one that produces the highest match score.

3.4 Considering Context

It might be easier to estimate the likelihood that an SSU-CC pair is a match by using information found in surrounding SSU-CC pairs, such as the SSU that follows a given SSU-CC pair. We do this by increasing the number of columns in the SSU-CC matrix to separate the examples based on the surrounding context.

For example, in Table 2, we cannot determine whether LAHN should map to 伦 or 连. But the SSU that follows LAHN clears up the ambiguity, because when LAHN immediately precedes BER, it maps to 伦, but when it is followed by NAH, it corresponds to 连. Table 6 displays a portion of the SSU-CC matrix that accounts for the contextual information provided by the SSU that follows an SSU-CC pair.

3.5 The Threshold

Given an SSU-CC name pair, the system produces a number between 0 and 1. But in order to evaluate the system in terms of precision, recall, and F-score, we need the system to return a yes (a match) or no (not a match) response. So we use a threshold value to separate those two cases.

The threshold value can be manually selected by a human, but this is often difficult to do effectively. So we developed the following automated approach to choose the threshold. After the training phase finishes developing Matrix #2, the system processes the training data[12] one more time.

[12] We tried selecting the threshold with data that was not used in training, and we found no statistically significant improvement.

Alignment	% of Data
#SSUs - #CCs ≥ 3	1.62%
#SSUs - #CCs = 2	6.66%
#SSUs - #CCs = 1	20.00%
#SSUs - #CCs = 0	60.60%
#SSUs - #CCs = -1	10.48%
#SSUs - #CCs = -2	0.61%
#SSUs - #CCs ≤ -3	0.02%

Table 7. Statistics of the Data

But this time it runs in application mode (Section 3.1), computing a match score for each training example. Then the system considers all possible ways to separate the yes and no responses with a threshold, selecting the threshold value that is the most effective on the training data.

Building the SSU-CC matrices does not require any negative examples (name pairs that do not match). However, we do require negative examples in order to determine the threshold and to evaluate the system. Our technique for generating negative examples involves randomly rearranging the names in the data.[13]

4 Evaluation of the System

We ran several experiments to test our system under a variety of different conditions. After describing our data and experimental method, we present some of our most interesting experimental results.

We used a set of nearly 500,000 Roman-Chinese person name pairs collected from Xinhua News Agency newswire texts. (Huang, 2005) Table 7 shows the distribution of the data based on alignment. Note that the PA hypothesis applies to more than 60% of the data.

We used the popular 10-fold cross validation approach[14] to obtain ten different evaluation scores. For each experiment we present the average of these scores.

Our system's precision (P), recall (R), and F-score (F) are: P = 98.19%, R = 94.83%, and F = 96.48%. These scores are much better than we originally expected to see for the challenging task of Roman-Chinese name matching.

Table 8 shows P, R, and F for subsets of the test data, organized by the number of SSUs mi-

[13] Unfortunately, there is no standard way to generate negative examples.

[14] The data is divided into ten subsets of approximately the same size, testing the system on each subset when trained on the other nine.

Alignment	P	R	F
#SSUs - #CCs ≥ 3	72.38%	94.02%	81.79%
#SSUs - #CCs = 2	95.26%	92.67%	93.95%
#SSUs - #CCs = 1	99.07%	93.27%	96.08%
#SSUs - #CCs = 0	99.87%	95.33%	97.55%
#SSUs - #CCs = -1	98.33%	96.42%	97.37%
#SSUs - #CCs = -2	73.80%	94.98%	83.04%
#SSUs - #CCs ≤ -3	7.54%	78.04%	13.71%

Table 8. Varying Alignment of Name Pairs

nus the number of CCs in the name pairs. The differences between scores in adjacent rows of each column are statistically significant.[15] Perfectly aligned name pairs proved to be the easiest, with F = 97.55%, but the system was also very successful on the examples with the number of SSUs and the number of CCs differing by one (F = 96.08% and F = 97.37%). These three cases account for more than 91% of the positive examples in our data set. (See Table 7.)

4.1 Deletion Hypotheses

We ran tests to determine whether the second pass through the training data (in which the SSUD and CCD hypotheses are applied) is effective. Table 9 shows the results on the complete set of test data, and all of the differences between the scores are statistically significant.

The first row of Table 9 presents F when the system made only one pass through the training data. The second row's experiments utilized the CCD hypothesis but ignored examples with more SSUs than CCs during training. For the third row, we used the SSUD hypothesis, but not the CCD hypothesis, and the last row corresponds to system runs that used all of the training examples. From these results, it is clear that both of the deletion hypotheses are useful, particularly the SSUD hypothesis.

4.2 Context

In Section 3.4, we suggested that contextual information might be useful. So we ran some tests, obtaining the results shown in Table 10. For the second row, we used no contextual information. Row 5 corresponds to the case where we gave the system access to the SSU immediately following the SSU-CC pair being analyzed. In row 6's experiment, we used the SSU immediately preceding the SSU-CC pair under consideration, and row 7 corresponds to system runs that accounted for both surrounding SSUs.

We also tried simplifying the contextual information to boolean values that specify whether an SSU-CC pair is at a boundary of its name or not, and rows 1, 3, and 4 of Table 10 show those results. "Left Border" is true if and only if the SSU-CC pair is at the beginning of its name, "Right Border" is true if and only if the SSU-CC pair is at the end of its name, and "Both Borders" is true if and only if the SSU-CC pair is at the beginning or end of its name. All differences in the table are statistically significant, except for those between rows 2, 3, and 4. These results suggest that the right border provides no useful information, even if the left border is also included in the SSU-CC matrix. But when the SSU-CC matrix only accounted for the left border, the F-score was significantly higher than the baseline. Providing more specific information in the form of SSUs actually made the scores go down significantly.

Hypotheses	F
PA	75.25%
PA & CCD	83.74%
PA & SSUD	92.86%
PA & CCD & SSUD	96.48%

Table 9. Varying the Training Data

4.3 Sparse Data

We were initially surprised to discover that using the rich information in the surrounding SSUs made the results worse. The explanation for this is that adding contextual information increases the size of the SSU-CC matrix, and so several of the numbers in the matrix become smaller. (For example, compare the values in the "BER" columns in Table 2 and Table 6.) This means that the system might have been suffering from a sparse data problem, which is a situation where there are not enough training examples to distinguish correct answers from incorrect answers, and so incorrect answers can appear to be correct by random chance.

There are two factors that can contribute to a sparse data problem. One is the amount of training data available — as the quantity of training data increases, the sparse data problem becomes less severe. The other factor is the complexity of

#	Contextual Information	F
1	Left Border	96.48%
2	No Context	96.25%
3	Both Borders	96.24%
4	Right Border	96.19%
5	Next SSU	87.53%
6	Previous SSU	85.89%
7	Previous SSU and Next SSU	47.89%

Table 10. Evaluation with Context

Contextual Info.	All Cells	Cells > 10^{-7}
No Context	0.128	4.35
Right Border	0.071	3.45
Left Border	0.069	3.45
Both Borders	0.040	3.13
Next SSU	0.002	1.12
Previous SSU	0.001	0.78
Both SSUs	*far less*	*far less*

Table 11. Num. SSU-CC Pairs per Matrix Cell

the learned model — as the model becomes more complex, the sparse data problem worsens.

Our system's model is the SSU-CC matrix, and a reasonable measure of the its complexity is the number of entries in the matrix. The second column of Table 11 shows the number of SSU-CC pairs in training divided by the number of cells in the SSU-CC matrix. These ratios are quite low, suggesting that there is a sparse data problem. Even without using any context, there are nearly 8 cells for each SSU-CC pair, on average.[16]

It might be more reasonable to ignore cells with extremely low values, since we can assume that these values are effectively zero. The third column of Table 11 only counts cells that have values above 10^{-7}. The numbers in that column look better, as the ratio of cells to training pairs is better than 1:4 when no context is used. However, when using the previous SSU, there are still more cells than training pairs.

Another standard way to test for sparse data is to compare the system's results as a function of the quantity of training data. As the amount of training data increases, we expect the F-score to rise, until there is so much training data that the F-score is at its optimal value.[17] Figure 3 shows the results of all of the context experiments that we ran, varying the amount of training data. (90% of the training data was used to get the F-scores in Table 10.) The t test tells us that "No Context" is the only curve that does not increase significantly on the right end. This suggests that all of the other curves might continue increasing if we used more training data. So even the "Both SSUs" case could potentially achieve a competitive score, given enough training examples. Also,

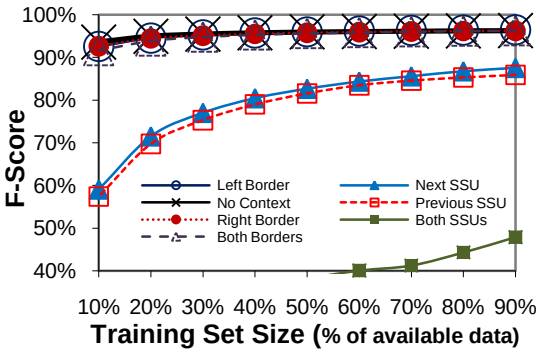

Figure 3. Testing for Sparse Data

more training data could produce higher scores than 96.48%.

5 Summary

We designed a system that achieved an F-score of 96.48%, and F = 97.55% on the 60.61% of the data that satisfies the PA hypothesis's condition.

Due to the paper length restriction, we can only provide short summaries of the other experiments that that we ran.

1) We experimentally compared six different equations for computing match scores and found that the best of them is an arithmetic or geometric average of Prob(SSU|CC) and Prob(CC|SSU).
2) We attempted to make use of two simple handcrafted rules, but they caused the system's performance to drop significantly.
3) We compared two approaches for automatically computing the pronunciation of a Roman name and found that using the Festival system (Black et al., 1999) alone is just as effective as using the CMU Pronunciation Dictionary (CMUdict, 1997) supplemented by Festival.
4) We tried computing the threshold value with data that was not used in training the system. However, this failed to improve the system's performance significantly.

6 Future Work

There are so many things that we still want to do, including:

1. modifying our system for the task of transliteration (Section 6.1),
2. running fair comparisons between our work and related research,
3. using Levenshtein's algorithm (Levenshtein, 1966) to implement the SSUD and

[16] It is true that a name pair can have multiple SSU-CC pairs, but even if the average number of SSU-CC pairs per name pair is as high as 8 (and it is not), one training name pair per SSU-CC matrix cell is still insufficient.

[17] Note that this value may not be 100%, because there are factors that can make perfection difficult to achieve, such as errors in the data.

CCD hypotheses, instead of exhaustively evaluating all possible deletion sets (Section 3.3),[18]

4. developing a standard methodology for creating negative examples,
5. when using contextual information, splitting rows or columns of the SSU-CC matrix only when they are ambiguous according to a metric such as Information Gain (Section 3.4),[19]
6. combining our system with other Roman-Chinese name matching systems in a voting structure (Van Halteren, Zavrel, and Daelemans, 1998),
7. independently evaluating the modules that determine pronunciation, construct syllables, and separate subsyllable units (Section 3),
8. converting phonemes into feature vectors (Aberdeen, 2006),
9. modifying our methodology to apply it to other similar languages, such as Japanese, Korean, Vietnamese, and Hawaiian.
10. manually creating rules based on information in the SSU-CC matrix, and
11. utilizing graphemic information.

6.1 Transliteration

We would like to modify our system to enable it to transliterate a given Roman name into Chinese in the following way. First, the system computes the SSUs as in Section 3.1. Then it produces a match score for every possible sequence of CCs that has the same length as the sequence of SSUs, returning all of the CC sequences with match scores that satisfy a predetermined threshold restriction.

For example, in a preliminary experiment, given the Roman name *Ellen*, the matcher produced the transliterations below, with the match scores in parentheses.[20]

埃 伦 (0.32)

埃 兰 (0.14)

埃 隆 (0.11)

埃 朗 (0.05)

Based on our data, the first and fourth results are true transliterations of *Ellen*, and the only true transliteration that failed to make the list is 埃连.

7 Conclusion

There was a time when computational linguistics research rarely used machine learning. Researchers developed programs and then showed how they could successfully handle a few examples, knowing that their programs were unable to generalize much further. Then the language community became aware of the advantages of machine learning, and statistical systems almost completely took over the field. Researchers solved all kinds of problems by tapping into the computer's power to process huge corpora of data. But eventually, the machine learning systems reached their limits.

We believe that, in the future, the most successful systems will be those developed by people cooperating with machines. Such systems can solve problems by combining the computer's ability to process massive quantities of data with the human's ability to intuitively come up with new ideas.

Our system is a success story of human-computer cooperation. The computer tirelessly processes hundreds of thousands of training examples to generate the SSU-CC matrix. But it cannot work at all without the insights of Wan and Verspoor. And together, they made a system that is successful more than 96% of the time.

References

Aberdeen, J. (2006) "geometric-featurechart-jsa-20060616.xls". *Unpublished*.

Andrade, Miguel. Smith, S. Paul. Cowlisha, Mike F. Gantner, Zeno. O'Brien, Philip. Farmbrough, Rich. et al. "F1 Score." (2009) *Wikipedia: The Free Encyclopedia*. http://en.wikipedia.org/wiki/F-score.

Black, Alan W. Taylor, Paul. Caley, Richard. (1999) *The Festival Speech Synthesis System: System Documentation*. Centre for Speech Technology Research (CSTR). The University of Edinburgh. http://www.cstr.ed.ac.uk/projects/festival/manual

CMUdict. (1997) *The CMU Pronouncing Dictionary*. v0.6. The Carnegie Mellon Speech Group. http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Cohen, W. Ravikumar, P. Fienberg, S. (2003) "A Comparison of String Distance Metrics for Name-

[18] We thank a reviewer for suggesting this method of improving efficiency.

[19] We thank a reviewer for this clever way to control the size of the SSU-CC matrix when context is considered.

[20] A manually-set threshold of 0.05 was used in this experiment.

Matching Tasks." *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*. Eds. Kambhampati, S. Knoblock, C. 73-78.

Condon, Sherri. Aberdeen, John. Albin, Matthew. Freeman, Andy. Mani, Inderjeet. Rubenstein, Alan. Sarver, Keri. Sexton, Mike. Yeh, Alex. (2006) "Multilingual Name Matching Mid-Year Status Report."

Condon, S. Freeman, A. Rubenstein, A. Yeh, A. (2006) "Strategies for Chinese Name Matching."

Freeman, A. Condon, S. Ackermann, C. (2006) "Cross Linguistic Name Matching in English and Arabic: A 'One to Many Mapping' Extension of the Levenshtein Edit Distance Algorithm." *Proceedings of NAACL/HLT*.

Gao, W. Wong, K. Lam, W. (2004) "Phoneme-Based Transliteration of Foreign Names for OOV Problem." *Proceedings of the First International Joint Conference on Natural Language Processing*.

Goto, I. Kato, N. Uratani, N. Ehara, T. (2003) "Transliteration Considering Context Information Based on the Maximum Entropy Method." *Proceedings of MT-Summit IX*.

Huang, Shudong. (2005) "LDC2005T34: Chinese <-> English Named Entity Lists v 1.0." *Linguistics Data Consortium*. Philadelphia, Pennsylvania. ISBN #1-58563-368-2. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T34.

International Phonetic Association. (1999) *Handbook of the International Phonetic Association : A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press, UK. ISBN 0521652367. http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=0521652367.

Jung, S. Hong, S. Paek, E. (2000) "An English to Korean Transliteration Model of Extended Markov Window." *Proceedings of COLING*.

Kang, B.J. Choi, K.S. (2000) "Automatic Transliteration and Back-Transliteration by Decision Tree Learning." *Proceedings of the 2nd International Conference on Language Resources and Evaluation*.

Klatt, D.H. (1990) "Review of the ARPA Speech Understanding Project." *Readings in Speech Recognition*. Morgan Kaufmann Publishers Inc. San Francisco, CA. ISBN 1-55860-124-4. 554-575.

Knight, K. Graehl, J. (1997) "Machine Transliteration." *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Kondrak, G. (2000) "A New Algorithm for the Alignment of Phonetic Sequences." *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*. Seattle, Washington. 288-295.

Kondrak, G. Dorr, B. (2004) "Identification of Confusable Drug Names: A New Approach and Evaluation Methodology." *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING)*. 952-958.

Levenshtein, V.I. (1966) "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." *Sov. Phys. Dokl.* 6. 707-710.

Li, H. Zhang, M. Su, J. (2004) "A Joint Source-Channel Model for Machine Transliteration." *Proceedings of ACL 2004*.

Mani, Inderjeet. Yeh, Alexander. Condon, Sherri. (2006) "Machine Learning from String Edit Distance and Phonological Similarity."

Meng, H. Lo, W. Chen, B. Tang, T. (2001) "Generating Phonetic Cognates to Handle Named Entities in English-Chinese Cross-Language Spoken Document Retrieval." *Proceedings of ASRU*.

Oh, Jong-Hoon. Choi, Key-Sun. (2006) "An Ensemble of Transliteration Models for Information Retrieval." *Information Processing & Management*. 42(4). 980-1002.

"Student's t Test." (2009) *Wikipedia: The Free Encyclopedia*. http://en.wikipedia.org/wiki/T_test#Equal_sample_sizes.2C_equal_variance.

Van Halteren, H., Zavrel, J. Daelemans, W. (1998) "Improving Data Driven Word-Class Tagging by System Combination." *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*. Montréal, Québec, Canada. 491-497.

Virga, P. Khudanpur, S. (2003) "Transliteration of Proper Names in Cross-Lingual Information Retrieval." *Proceedings of the ACL Workshop on Multi-lingual Named Entity Recognition*.

Wan, Stephen. Verspoor, Cornelia Maria. (1998). "Automatic English-Chinese Name Transliteration for Development of Multilingual Resources." *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*. Montréal, Québec, Canada.

Wellner, B. Castano, J. Pustejovsky, J. (2005) "Adaptive String Similarity Metrics for Biomedical Reference Resolution." *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies, and Databases: Mining Biological Semantics*. 9-16. http://www.cs.brandeis.edu/~wellner/pubs/Wellner-StringSim-BioLINK.pdf.

Winkler, W. "Methods for Record Linkage and Bayesian Networks." (2002) *Proceedings of the Section on Survey Research Methods, American Statistical Association*. http://www.census.gov/srd/www/byyear.html.