Name Matching Between Roman and Chinese Scripts

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 was developed by combining the creativity of human intuition with the power of machine learning. Our system correctly determines whether a name in Chinese script and a name in Roman script match 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, the Arabic name, الشكري, has been transliterated 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.

In this work, we have focused on the task of determining whether a name part in Chinese (Mandarin) script matches a name part in Roman script,[1] where a name part is a single "word" in a person's name (such as a middle name or a surname), and two names match if one is a transliteration of the other.[2] This problem, which is called cross-script name matching, has many applications, such as identity matching, improving search engines, and aligning parallel corpora.

Our system was developed by combining 1) the creative power of human intuition, which can come up with clever ideas and 2) the computational power of machine learning, which can analyze large quantities of data. Wan and Verspoor (1998) designed an algorithm that divides names into pieces that are just the right size for Roman-Chinese name matching. These "subsyllable units" are discussed in Section 2.2. And, armed with Wan and Verspoor's algorithm, a machine learning approach determines the relationships between Chinese characters and subsyllable units. The details can be found in Section 3.

Our experimental results, are in Section 4.3. The system correctly determines whether a Chinese name and a Roman name match with F = 96.5%.[3] And, for name pairs that satisfy the "Perfect Alignment" hypothesis condition, which will be presented in Section 2.2, F = 97.6%.

2 Related Work

2.1 Chinese-English Name Matching

The challenges of matching names across Chinese and Roman scripts are discussed by Condon et al. (2006). Also, in Section 6 of their paper, they offer 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 and Choi, 2006; Virga and Khudanpur, 2003; Wellner et al., 2005; Winkler, 2002)

[1] In this paper, we often use the word "Chinese" to refer to "Chinese script", and "Roman script" is usually abbreviated as "Roman".

[2] A third script might separate 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. However, we believe that there are very few, if any, examples in our test data that passed through a third script.

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

Roman Characters:	Albertson
Roman 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

The Levenshtein algorithm is an efficient way to compute string edit distance (Levenshtein, 1966), which can quantify the similarity between two names. However, to apply this algorithm to cross-script name matching, the names must be transformed from different scripts into a common format. Freeman et al. (2006) developed an strategy for Arabic-Roman string matching that used equivalence classes of characters to normalize the names so that Levenshtein's method could be used. Mani et al. (2006) modified their system for Chinese and extended the Levenshtein approach, attaining F = 85.2%. Then when they trained a machine learning algorithm on that system's output, it improved to F = 93.1%

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

2.2 Subsyllable Units

When Chinese names are transliterated into Roman names and vice versa, it is usually based on the way the names are pronounced.[4] However, it is not possible to derive a character-to-character mapping between the two versions of a name. This is because 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 as 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) The vowel phoneme may be preceded by, at most, one consonant 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 requirements of an SSU, but the other two need to be broken into smaller pieces, resulting in five SSUs. There are also five CCs in the Chinese transliteration, 阿尔贝特松, and the second and sixth rows in Table 1 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/, we hypothesize 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 4.3, we show that the PA hypothesis works very well when its antecedent is true. However, it is not uncommon to have more SSUs than CCs in a matching name pair. Often this can be explained by an SSU being left out of the Chinese transliteration, perhaps because it is a sound that is not common to Chinese. For example, "Carlberg" (KAA,R,L,BER,G) may be transliterated as 卡尔贝里. 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 have the corresponding hypothesis.

CCs Deletion (CCD) hypothesis
If a Roman name corresponds to a sequence of n

[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. However, our experimental results suggest that the exceptions are quite rare.

[5] The phoneme ɚr/, as in Alb ertson, is treated as a single vowel phoneme.

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

[7] To represent phonemes, we use two different standards, The sounds between slashes (like ɚr/) are in IPA format (International Phonetic Association, 1999), and the phonemes in all capital letters (like ER) are in ARPABET format. (Klatt, 1990)

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

We will show how we utilize these hypotheses in Section 3.

3 Machine Learning

Our group took a machine learning approach to generate a mapping between SSUs and CCs. We will first show how our system can do Roman-Chinese name matching, and then we will present the training procedure.

3.1 The Application Phase

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 the Chinese name matches the Roman name. This is accomplished via the process presented in Figure 1.

Given a Roman-Chinese name pair, the system determines how the Roman name should be pronounced by running it through the Festival system. (Black et al., 1999) Next, Wan and Verspoor's algorithms join the phonemes to form syllables and divide the syllables into SSUs. If the number of SSUs is equal to the number of characters in the Chinese name, the PA hypothesis applied to align each SSU with a CC. The match score is computed using a data structure called the SSU-CC matrix (subsyllable unit – Chinese character matrix), which has a value for each SSU-CC pair, and this value is supposed to 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, we would expect the name pair <Albert, 阿尔贝特> to have a relatively high match score, because the SSU form of Albert is AE,L,BER,T, and the scores in the SSU-CC matrix for <AE,阿>, <L,尔>, <BER,贝> and <T,特> are 2, 2, 3, and 2, respectively.[8] Alternatively, <Albert, 尔贝特阿> should be assigned a very low match score, because the values of <AE,尔>, <L,贝>, <BER,格>, and <T,阿> are all 0.

[8] Lack of space prevents us from discussing the equation we used to compute a match score from these four values.

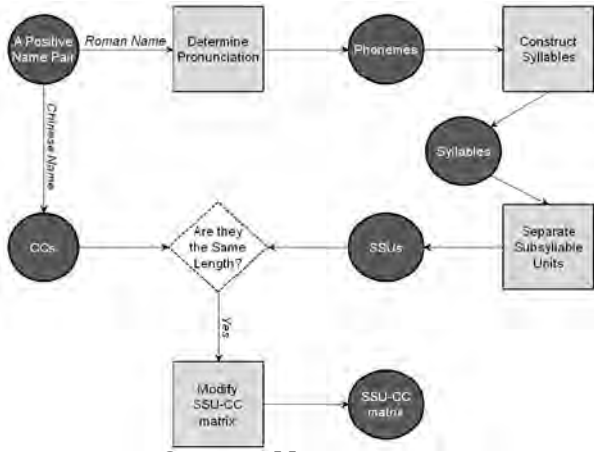

Figure 2: Training Mode

	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: An Example SSU-CC Matrix

3.2 Computing Match Scores

Given an SSU-CC matrix and a Roman-Chinese name pair, there are multiple options for computing a match score for the name pair. Consider the case where there are the same number of CCs and SSUs. A simple approach is to use Formula (1), where S_1, S_2, …, S_n are the SSUs that correspond to the Chinese name with characters C_1, C_2, ..., C_n, m(X,Y) is the value of the entry for the SSU, X, and the CC, Y, in the SSU-CC matrix, and MS is the resulting match score.

$$(1) \quad MS = \prod_{i=1}^{n} m(S_i, C_i)$$

With the SSU-CC matrix in Table 2, the name pair, <Albert, 阿尔贝特> has a score of 2x2x3x2 = 24.

It is usually desirable for a metric's values to range from 0 to 1. One way to do this is to design a formula that estimates the *probability* that the Roman name is paired with the Chinese name:

(2) $\Pr([S_1 \ldots S_n] \& [C_1 \ldots C_n])$

Making the simplifying assumption that context is irrelevant, we get:

(3) $\prod_{i=1}^{n} \Pr(S_i \& C_i)$.

Since this is a product of n probabilities, it is reasonable to take the n[th] root of it in order to produce a match score that is comparable in magnitude to any of the other match scores. (Otherwise, there would be a bias in favor of shorter names.) The result is:

(4) $\sqrt[n]{\prod_{i=1}^{n} \Pr(S_i \& C_i)}$

We can estimate[9] the probabilities from values in the matrix:

(5) $\Pr(S_i \& C_i) \approx \frac{m(S_i, C_i)}{\sum_S \sum_C m(S, C)}$

So, the result is:

(6) $MS = \sqrt[n]{\prod_{i=1}^{n} \frac{m(S_i, C_i)}{\sum_S \sum_C m(S, C)}}$

With this formula and the SSU-CC matrix in Table 2, <Albert, 阿尔贝特> gets a score of $\sqrt{\frac{2}{19} \times \frac{2}{19} \times \frac{3}{19} \times \frac{2}{19}} = 0.014$.

As an alternative, instead of basing the formula on $Pr(S_i \& C_i)$, it might be reasonable to estimate $Pr(S_i|C_i)$ or $Pr(C_i|S_i)$. Actually, since it is unclear which of those two choices are better (they both seem to be reasonable options), we might consider multiplying them together or adding them together. (The details of these four options have been omitted for lack of space.)

3.3 The Training Phase

To generate an SSU-CC matrix, our system trained on a large corpus of Roman-Chinese name pairs that was obtained through the Linguistic Data Consortium (Huang, 2005). The part of this corpus we used has 406,860 personal name pairs drawn from a database of the Xinhua news agency. Since all of these pairs are human-attested cross-script name matches, they can be used as positive examples (true matches) to train and evaluate an automated system.

Figure 2 shows a diagram of the training system. The procedure for transforming the Roman name to a sequence of SSUs is identical the that presented in Section 3.1. Then, if the number of SSUs is the same as the number of CCs, the PA hypothesis is applied 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
LIY ↔ 利

So the SSU-CC matrix is modified 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.

3.4 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, a second SSU-CC matrix is built. Each name pair that satisfies the PA hypothesis's condition is treated exactly the same as it was in the first pass (Section 3.1). But the other name pairs are handled based on the SSUD hypothesis or the CCD hypothesis by deleting some SSUs or CCs in the following way. For a given Roman-Chinese name pair:

	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

```
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 to the name pair.

A column named Ø and a row named Ø are included in the second SSU-CC matrix to hold values that represent the frequency of each SSU and CC being deleted.

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 4 CCs. So the PA hypothesis is not applicable, and so this name pair is ignored in the first pass. The first pass generates the first SSU-CC matrix (Table 2).

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>.)

For each possibility, the name pair is evaluated using the values in Table 2 producing the scores in the second column of Table 4. Then the scores are scaled to sum to 1, as shown in the third column. These numbers are used as weights to determine how much impact each of the five options has on the second matrix. They are scaled to sum to 1 so that the name pair has the same total influence as any name pair that the PA hypothesis applies to.

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

Table 5 shows part of the second SSU-CC matrix after the second pass has completed. This matrix is saved as the final trained model, which can be used to evaluate new name pairs in the application phase. In that phase, the system evaluates a name pair that does not satisfy the PA hypothesis's condition by trying all possible deletions and selecting the one that produces the highest score.

	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: Second-Pass SSU-CC Matrix

3.5 Considering Context

It might be easier to estimate the likelihood that an SSU-CC pair is a match if information found in surrounding SSU-CC pairs (such as the SSU that follows a given SSU-CC pair) is taken into account. This can be done by increasing the number of columns in the SSU-CC matrix to separate examples based on the surrounding context. For example, in Table 2, we cannot determine whether LAHN should map to 伦 or 连. But if the SSU that follows a given SSU-CC pair is considered, the ambiguity is cleared up, because when LAHN is followed by BER, it maps to 伦, but when NAH follows, it is mapped to 连. Table 6 displays a portion of the SSU-CC matrix that accounts for the contextual information provided by the next SSU.

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

Table 6: Considering Context

3.6 The Threshold

Given an SSU-CC a 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 needed the system to return a yes (a match) or no (not a match) response. So we used a threshold value to separate the two responses.

The threshold can be manually selected, but it may be difficult for a person to come up with a good threshold value. So we developed an automated approach to choose the threshold. After

the training phase has finished developing the final SSU-CC matrix, the training data[10] is run through the system again, but this time it is not being used for learning. Instead, it is running in application mode (Section 3.1). All of the training examples are ordered by their match scores, and the system considers all possible ways to separate the yes and no responses with a threshold. The selected threshold is the one that produces the highest F-score.[11]

4 Evaluation of the System

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

4.1 The Data

We ran experiments on a set of 471,621 Roman-Chinese name pairs that match (positive examples), which was collected from Xinhua News Agency newswire texts. (Huang, 2005) We also had a set of 451,358 negative examples (name pairs that do not match). Table 7 shows how many positive and negative examples we had for various alignments. Note that the PA hypothesis applies to more than 60% of the positive examples.

Alignment	Positive Examples	Negative Examples
#SSUs - #CCs ≥ 3	7,653	34,266
#SSUs - #CCs = 2	31,433	57,384
#SSUs - #CCs = 1	94,338	97,420
#SSUs - #CCs = 0	285,779	130,150
#SSUs - #CCs = -1	49,431	76,372
#SSUs - #CCs = -2	2,886	37,249
#SSUs - #CCs ≤ -3	101	18,517

Table 7: Statistics of the Data

4.2 Experimental Design

To evaluate the system, we used the popular 10-fold cross validation approach[12] to obtain ten different evaluation scores for each experiment.

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

[11] This procedure requires negative examples. Our method for obtaining negative examples is presented in Section 5.2.

[12] In this experimental method, the data is divided into ten subsets of approximately the same size, testing the system on each subset when trained on the other nine.

#	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	Both SSUs	47.89%

Table 8: Evaluation with Context

We present the averages of these scores and use the homoscedastic t-test ("Student's", 2009) to decide whether the difference between two results is statistically significant.

4.3 Experiments

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

Table 9 shows P, R, and F for separate subsets of the test data based on numbers of SSUs and CCs in the name pairs. The difference between scores in adjacent rows of each column are statistically significant. Perfectly aligned name pairs (#SSUs = #CCs) proved to be the easiest, with F = 97.55%, but the system was also successful when the number of SSUs and the number of CCs differ by 1 (F = 96.08% and F = 97.37%). These three cases account for 91% of the positive examples in our data set.

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	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 9: Varying Alignment of Name Pairs

Deletion Hypotheses

We ran tests to determine whether the second pass through the training data (in which the SSUD and CCD hypotheses are applied) improved the results. Table 10 shows the results, and all of the differences are statistically significant. The first row of the table presents F when we only used perfectly aligned examples in training. For the second row, examples with more SSUs than CCs were ignored, and the CCD hypothesis was applied for examples with more

CCs than SSUs. For the third row, the SSUD hypothesis was applied, and examples with more CCs than SSUs were ignored. And the last row represents the case where all of the training examples were used. From these results, it is clear that both of the deletion hypotheses are useful.

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

Table 10: Deletion Strategies During Training

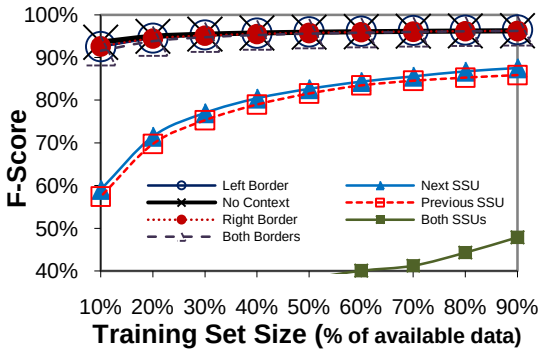

Figure 3: Testing for Sparse Data

Context

In Section 3.5, we hypothesized that contextual information might be useful. So we ran some tests, and the results are presented in Table 8. For the 2nd row, we used no contextual information. Row 6 shows the results when we gave the system access to the SSU immediately preceding the SSU-CC pair that is being analyzed. In row 5's experiment, we used the SSU immediately following the SSU-CC pair under consideration. And for row 7, both contextual SSUs were considered.[13]

We also considered simplifying the contextual information to boolean values that specify whether or not a name boundary exists on each side of the SSU-CC pair being analyzed. The results of these experiments are shown in rows 1, 3, and 4 of Table 8. All differences in the table are statistically significant, except for those between rows 2, 3, and 4. These results suggests that the right boundary provides no useful information, even if the left boundary is also considered. However, when the left boundary alone is considered, the scores go up significantly. But we were surprised to find that providing more information in the form of SSUs actually made the scores go down. We will now provide an explanation for these unexpected results.

Sparse Data

We were surprised that the surrounding SSUs should make the results worse, However, since the contextual information makes the matrix larger, most of the numbers in the matrix are lower. This means that we are more susceptible to a sparse data problem.

A sparse data problem 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 simply by random chance. There are two factors that can contribute to a sparse data problem. One, of course, is the quantity of training data; as the quantity of training data increases, the sparse data problem becomes less severe. The other factor is the complexity of the learned model. As the model becomes more complex, the sparse data problem increases.

Our system's learned model is the SSU-CC matrix, and a reasonable measure of the matrix's complexity is the number of entries in it. In the experiment reported here, the number of cells in the matrix are given in the second column of Table 11. These numbers are huge, suggesting that there is a sparse data problem. Even without using any context, there are 8 cells for each name pair in the training set.[14] (These matrices were built using all 471,621 name pairs.) But it might also be reasonable to discount the cells that had very low values, since we could assume that these are SSU-CC pairs that do not exist in reality. The third column shows how many cells have values above 10^{-7} in the matrix. These numbers look better, as the ratio of cells to training examples is 1:4 when no context is used. However, when using the previous SSU, there are more cells than training examples.

Contextual Information	#Cells	#Cells > 10^{-7}
No Context	3,699,528	108,695
Right Border	6,644,120	136,138
Left Border	6,787,592	137,067
Both Borders	11,688,150	151,873
Next SSU	266,330,650	419,438
Previous SSU	390,944,120	605,170

[13] We did not experiment with context that was not adjacent to the SSU-CC under consideration or with contextual information that included CCs.

[14] It's true that a name pair can have multiple SSU-CC pairs, but even if the average number of SSU-CC pairs per name pair was as high as 8 (which it's not), one training example per cell in the model is sparse.

Both SSUs	*even more*	*even more*

Table 11: Model Complexity

It is possible to test for sparse data with the following experiment. We compared the results for different amounts of training data. As more training data is used, we can expect the scores to increase, until there is so much training data that the scores are at the highest possible value. Note that this value may not be 100%, as there are other factors that can make perfection difficult to achieve, such as errors in the data and name pairs that do not conform to our hypotheses in Section 2.2.

Figure 3 shows the results of all of the context experiments that we ran. The t-test tells us that all of these curves are still increasing at the right end, except for "No Context". However, if the name boundaries is the only contextual information used, the matrix is much smaller than if any SSU can be considered, so the sparse data problem is not as serious for boundaries.

In every case, we can clearly see that we have passed the bend of the curve, which is the point where the slope is exactly 1. The "80-20 rule" states that 80% of the job can be achieved with 20% of the work, but the remaining 80% of the work is required to get the other 20% of the job completed. So, because of diminishing returns, we can conclude the results are nearly as high as they're ever going to be.

Therefore, based on this analysis, we can explain the results in Table 8. Row 7 has the most serious sparse data problem, so that is why its scores are the lowest. For the same reason, rows 5 and 6 are worse than the other rows. And although we may conclude that the previous SSU provides less benefit than the next SSU, Figure 3 suggests that the previous SSU and the next SSU have sparse data problems of about the same degree (though we can't be sure of that), so that, given enough training data, row 6 might become better than row 5.

The left boundary gets the highest score, suggesting that it truly is the best, and even though there is a sparse data problem for the left boundary, it appears that it is not serious enough to counter the benefits.

The Evaluation Formula

In Section 3.2, we explained that we could not decide what formula should be used for evaluating name pairs. We presented 5 reasonable alternatives, each of which produces a score from 0 to 1 for each name pair, where higher numbers represent increased likelihood that the name pair is a match. However, as we were unable to derive any theoretical conclusions about which formula(s) would be the most effective, we decided to settle this matter experimentally. Table 12 shows the results for each formula. Statistically significant differences are separated by solid lines. P(SSU & CC) is the worst, followed by P(CC | SSU), and then P(SSU | CC). The best scores were produced using the most complex of the formulas, the sum and product of P(SSU | CC) and P(CC | SSU). The results did not show a preference for one of these over the other, so the decision to use the product instead of the sum was more or less arbitrary.

Formula	P	R	F
P(SSU|CC) + P(CC|SSU)	98.34%	94.81%	96.54%
P(SSU|CC) × P(CC|SSU)	98.19%	94.83%	96.48%
P(SSU|CC)	98.03%	94.71%	96.38%
P(CC|SSU)	96.96%	93.19%	95.04%
P(SSU & CC)	94.75%	91.14%	92.91%

Table 12: Evaluation Formulas

5 Discussion

5.1 Past Work

We designed a system that achieved an F-score of 96.48%, which is higher than results previously reported on the Roman-Chinese name matching task. And F = 97.55% on the 60.61% of the data that satisfied the PA hypothesis's condition.

We ran several experiments that, due to space constraints, we were unable to present:

1) We experimentally compared six equations for computing match scores and found that the best was an arithmetic or geometric average of Prob(SSU|CC) and Prob(CC|SSU).
2) Our system automatically selects a threshold value to determine how high a match score must be to declare that a name pair matches, and we discovered that using the training data for this purpose is as effective as using a held-out tune set.
3) We analyzed the effectiveness of including contextual information in the SSU-CC matrix discovering that it caused a sparse data problem, but performance still improved when the system considered whether or not each SSU-CC pair was found at the beginning of a name.
4) We discovered that the second pass of training significantly improves the system's performance.

5) We attempted to make use of two simple handcrafted rules, but they caused the system's performance to drop significantly.
6) 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 was just as effective as using the CMU Pronunciation Dictionary (CMUdict, 1997) supplemented by Festival.
7) We began an investigation into modifying our system to do the task of transliteration and showed some promising preliminary results.

5.2 Future Work

There are many things that we still want to do, including 1) using our system for the task of transliteration, 2) developing a methodology for creating negative examples, 3) running fair comparisons between our system and other cross-script name matching systems, 4) attempting to apply our methodology to other languages, such as Korean and Japanese, 5) utilizing graphemic information, 6) converting phonemes into feature vectors (Aberdeen, 2006), 7) combining our system with other systems in a voting structure (Van Halteren, Zavrel, and Daelemans, 1998), and 8) manually creating rules based on information in the SSU-CC matrix. The first two ideas are discussed below.

1) Transliteration

Our system can be modified to transliterate a given Roman name into Chinese in the following way.The SSUs are computed following the procedure presented in Section 3.1. Then a match score is produced for every possible sequence of CCs with the same length as the sequence of SSUs. Finally, any of the CC sequences with a match score below a predetermined threshold are dropped.

For example, in a preliminary experiment, given the Roman name *Ellen* the matcher produced the transliterations[15] below, assigning a match score to each transliteration.

(9)
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.

2) Negative Examples

In our experiments, the data included 451,358 name pairs that do not match. These negative examples were generated by taking the positive examples in the test data and randomly moving the Chinese names around so that no Chinese name was paired with the correct Roman name.[16] Mani et al. (2006) suggested that this might not be a good way to generate negative examples, since the names in nearly all of these name pairs are very different. So it is quite easy for the system to label them correctly, which inflates the statistical results. Mani et al. used a special technique to filter out the easiest negative name pairs by estimating their similarity using another algorithm. It might be reasonable to have our system select negative test examples in the same manner. Another option is to test with the same data that Mani et al. used, to make a direct comparison between the systems. Alternatively, it might also be useful to pair every Roman name with every Chinese name (except for the ones that it matches), as then the most difficult examples would be included in the test data.

When deciding which name pairs should be used as negative examples, it is best to consider the type of data that the system is being designed to evaluate. If the target application would have the system comparing many name pairs that are very different, then the same type of data should be used in evaluating the system. Alternatively, if the system is going to be asked to make very fine distinctions between names, then the test data should be selected accordingly.

5.3 Conclusions

There was a time when computational linguistics research rarely used statistical machine learning approaches. Researchers would develop programs and then show how they could successfully handle a few examples, but their programs were unable to generalize much further. Then the language community became aware of the ad-

[15] A threshold of 0.05 was used in this experiment.

[16] We also required that each Chinese name in the negative examples was paired with Roman names with the same number of SSUs as a Roman name that the Chinese name corresponded to in a positive example.

vantages of machine learning, and statistical systems almost completely took over the field. The power of the computer to process huge corpora of data was shown to effectively solve all kinds of problems. But eventually, these purely statistical systems will reach their limits. We expect that the most successful systems in the future will be those that are developed by people and machines working in cooperation. Such systems solve problems by combining the computer's ability to apply statistical approaches to massive quantities of data with a human's ability to intuitively generate 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 would never be as effective as it is without the algorithms created by Wan and Verspoor. And together, they can generate 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 burgh. 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-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 Foreing 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. 554-575. ISBN 1-55860-124-4.

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 Bayersian Networks.” (2002) *Proceedings of the Section on Survey Research Methods, American Statistical Association*. http://www.census.gov/srd/www/byyear.html.

.