Learning to Match Names Across Languages

Inderjeet Mani
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730, USA
imani@mitre.org

Alex Yeh
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730, USA
asy@mitre.org

Sherri Condon
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102, USA
scondon@mitre.org

Abstract

We report on research on matching names in different scripts across languages. We explore two trainable approaches based on comparing pronunciations. The first, a cross-lingual approach, uses an automatic name-matching program that exploits rules based on phonological comparisons of the two languages carried out by humans. The second, monolingual approach, relies only on automatic comparison of the phonological representations of each pair. Alignments produced by each approach are fed to a machine learning algorithm. Results show that the monolingual approach results in machine-learning based comparison of person-names in English and Chinese at an accuracy of over 97.0 F-measure.

1 Introduction

The problem of matching pairs of names which may have different spellings or segmentation arises in a variety of common settings, including integration or linking database records, mapping from text to structured data (e.g., phonebooks, gazetteers, and biological databases), and text to text comparison (for information retrieval, clustering, summarization, coreference, etc.). For named entity recognition, a name from a gazetteer or dictionary may be matched against text input; even within monolingual applications, the forms of these names might differ. In multi-document summarization, a name may have different forms across different sources. Systems that address this problem must be able to handle variant spellings, as well as abbreviations, missing or additional name parts, and different orderings of name parts.

In multilingual settings, where the names being compared can occur in different scripts in different languages, the problem becomes relevant to additional practical applications, including both multilingual information retrieval and machine translation. Here special challenges are posed by the fact that there usually aren't one-to-one correspondences between sounds across languages. Thus the name *Stewart*, pronounced / s t u w ə r t / in IPA, can be mapped to Mandarin "斯图尔特", which is Pinyin "si tu er te", pronounced /s i t^h u a ɻ t^h e/, and the name *Elizabeth* / I l I z ə b ɛ θ/ can map to "伊丽莎白", which is Pinyin "yi li sha bai", pronounced /I l I ʂ ɑ p aI/. Further, in a given writing system, there may not be a one-to-one correspondence between orthography and sound, a well-known case in point being English. In addition, there may be a variety of variant forms, including dialectical variants, (e.g., *Bourguiba* can map to *Abu Ruqayba*), orthographic conventions (e.g., Anglophone *Wasim* can map to Francophone *Ouassime*), and differences in name segmentation (*Abd Al Rahman* can map to *Abdurrahman*). Given the high degree of variation and noise in the data, approaches based on machine learning are needed.

The considerable differences in possible spellings of a name also call for approaches which can compare names based on pronunciation. Recent work has developed pronunciation-based models for name comparison, e.g., (Sproat, Tao and Zhai 2006) (Tao et al. 2006). This paper explores trainable pronunciation-based models further.

Coling 2008: Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization, pages 2–9
Manchester, August 2008

<table>
<tr><th></th><th>Roman</th><th>Chinese (Pinyin)</th><th>Alignment</th><th>Score</th></tr>
<tr><td><b>LEV</b></td><td>ashburton</td><td>ashenbodu</td><td>| a s h b u r t o n |
| a s h e n b o d u |</td><td>0.67</td></tr>
<tr><td><b>MLEV</b></td><td>ashburton</td><td>ashenbodu</td><td>| a s h - - b u r t o n |
| a s h e n b o - d u - |</td><td>0.72</td></tr>
<tr><td><b>MALINE</b></td><td>asVburton</td><td>aseCnpotu</td><td>| a sV - b < u r t o | n
| a s eC n p o - t u | -</td><td>0.48</td></tr>
</table>

Table 1: Matching “Ashburton” and “阿什伯顿”

Consider the problem of matching Chinese script names against their English (Pinyin) Romanizations. Chinese script has nearly 50,000 characters in all, with around 5,000 characters in use by the well-educated. However, there are only about 1,600 Pinyin syllables when tones are counted, and as few as 400 when they aren’t. This results in multiple Chinese script representations for a given Roman form name and many Chinese characters that map to the same Pinyin forms. In addition, one can find multiple Roman forms for many names in Chinese script, and multiple Pinyin representations for a Chinese script representation.

In developing a multilingual approach that can match names from any pair of languages, we compare an approach that relies strictly on **monolingual** knowledge for each language, specifically, grapheme-to-phoneme rules for each language, with a method that relies on **cross-lingual** rules which in effect map between graphemic and/or phonemic representations for the specific pair of languages.

The monolingual approach requires finding data on the phonemic representations of a name in a given language, which (as we describe in Section 4) may be harder than finding more graphemic representations. But once the phonemic representation is found for names in a given language, then as one adds more languages to a system, no more work needs to be done in that given language. In contrast, with the cross-lingual approach, whenever a new language is added, one needs to go over all the existing languages already in the system and compare each of them with the new language to develop cross-lingual rules for each such language pair. The engineering of such rules requires bilingual expertise, and knowledge of differences between language pairs. The cross-lingual approach is thus more expensive to develop, especially for applications which require coverage of a large number of languages.

Our paper investigates whether we can address the name-matching problem without requiring such a knowledge-rich approach, by carrying out a comparison of the performance of the two approaches. We present results of large-scale machine-learning for matching personal names in Chinese and English, along with some preliminary results for English and Urdu.

2 Basic Approaches

2.1 Cross-Lingual Approach

Our cross-lingual approach (called MLEV) is based on (Freeman et al. 2006), who used a modified Levenshtein string edit-distance algorithm to match Arabic script person names against their corresponding English versions. The Levenshtein edit-distance algorithm counts the minimum number of insertions, deletions or substitutions required to make a pair of strings match. Freeman et al. (2006) used (1) insights about phonological differences between the two languages to create rules for equivalence classes of characters that are treated as identical in the computation of edit-distance and (2) the use of normalization rules applied to the English and transliterated Arabic names based on mappings between characters in the respective writing systems. For example, characters corresponding to low diphthongs in English are normalized as “w”, the transliteration for the Arabic “و”character, while high diphthongs are mapped to “y”, the transliteration for the Arabic “ي” character.

Table 1 shows the representation and comparison of a Roman-Chinese name pair (shown in the title) obtained from the Linguistic Data Consortium’s LDC Chinese-English name pairs corpus (LDC 2005T34). This corpus provides name part pairs, the first element in English (Roman characters) and the second in Chinese characters, created by the LDC from Xinhua Newswire's proper name and who's who databases. The name part can be a first, middle or last name. We compare the English form of the name with a Pinyin Romanization of the Chinese. (Since the Chinese is being compared with English, which is toneless, the tone part of Pinyin is being ignored throughout this paper.) For this study, the Levenshtein edit-distance score (where a perfect match scores zero) is normalized to a similarity score as in (Freeman et al. 2006), where the score ranges from 0 to 1, with 1 being a perfect match. This edit-distance score is shown in the LEV row.

The MLEV row, under the Chinese Name column, shows an "Englishized" normalization of the Pinyin for *Ashburton*. Certain characters or character sequences in Pinyin are pronounced differently than in English. We therefore apply certain transforms to the Pinyin; for example, the following substitutions are applied at the start of a Pinyin syllable, which makes it easier for an English speaker to see how to pronounce it and renders the Pinyin more similar to English orthography: "u:" (umlaut "u") => "u", "zh" => "j", "c" => "ts", and "q" => "ch" (so the Pinyin "Qian" is more or less pronounced as if it were spelled as "Chian", etc.). The MLEV algorithm uses equivalence classes that allow "o" and "u" to match, which results in a higher score than the generic score using the LEV method.

2.2 Monolingual Approach

Instead of relying on rules that require extensive knowledge of differences between a language pair[2], the monolingual approach first builds phonemic representations for each name, and then aligns them. Earlier research by (Kondrak 2000) used dynamic programming to align strings of phonemes, representing the phonemes as vectors of phonological features, which are associated with scores to produce similarity values. His program ALINE includes a "skip" function in the alignment operations that can be exploited for handling epenthetic segments, and in addition to 1:1 alignments, it also handles 1:2 and 2:1 alignments. In this research, we made extensive modifications to ALINE to add the phonological features for languages like Chinese and Arabic and to normalize the similarity scores, producing a system called MALINE.

In Table 1, the MALINE row[3] shows that the English name has a palato-alveolar modification on the "s" (expressed as "sV"), so that we get the sound corresponding to "sh"; the Pinyin name inserts a centered "e" vowel, and devoices the bilabial plosive /b/ to /p/. There are actually sixteen different Chinese 'pinyinizations' of *Ashburton*, according to our data prepared from the LDC corpus.

3 Experimental Setup

3.1 Machine Learning Framework

Neither of the two basic approaches described so far use machine learning. Our machine learning framework is based on learning from alignments produced by either approach. To view the learning problem as one amenable to a statistical classifier, we need to generate labeled feature vectors so that each feature vector includes an additional class feature that can have the value 'true' or 'false.' Given a set of such labeled feature vectors as training data, the classifier builds a model which is then used to classify unlabeled feature vectors with the right labels.

A given set of attested name pairs constitutes a set of positive examples. To create negative pairs, we have found that randomly selecting elements that haven't been paired will create negative examples in which the pairs of elements being compared are so different that they can be trivially separated from the positive examples. The experiments reported here used the MLEV score as a threshold to select negatives, so that examples below the threshold are excluded. As the threshold is raised, the negative examples should become harder to discriminate from positives (with the harder problems mirroring some of the "confusable name" characteristics of the real-world name-matching problems this technology is aimed at). Positive examples below the threshold are also eliminated. Other criteria, including a MALINE score, could be used, but the MLEV scores seemed adequate for these preliminary experiments.

Raising the threshold reduces the number of negative examples. It is highly desirable to balance the number of positive and negative examples in training, to avoid the learning being

[2] As (Freeman et al., 2006) point out, these insights are not easy to come by: "These rules are based on first author Dr. Andrew Freeman's experience with reading and translating Arabic language texts for more than 16 years" (Freeman et al., 2006, p. 474).

[3] For the MALINE row in Table 1, the ALINE documentation explains the notation as follows: "every phonetic symbol is represented by a single lowercase letter followed by zero or more uppercase letters. The initial lowercase letter is the base letter most similar to the sound represented by the phonetic symbol. The remaining uppercase letters stand for the feature modifiers which alter the sound defined by the base letter. By default, the output contains the alignments together with the overall similarity scores. The aligned subsequences are delimited by '|' signs. The '<' sign signifies that the previous phonetic segment has been aligned with two segments in the other sequence, a case of compression/expansion. The '-' sign denotes a "skip", a case of insertion/deletion."

biased by a skewed distribution. However, when one starts with a balanced distribution of positive and negatives, and then excludes a number of negative examples below the threshold, a corresponding number of positive examples must also be removed to preserve the balance. Thus, raising the threshold reduces the size of the training data. Machine learning algorithms, however, can benefit from more training data. Therefore, in the experiments below, thresholds which provided woefully inadequate training set sizes were eliminated.

One can think of both the machine learning method and the basic name comparison methods (MLEV and MALINE) as taking each pair of names with a known label and returning a system-assigned class for that pair. Precision, Recall, and F-Measure can be defined in an identical manner for both machine learning and basic name comparison methods. In such a scheme, a threshold on the similarity score is used to determine whether the basic comparison match is a positive match or not. Learning the best threshold for a dataset can be determined by searching over different values for the threshold.

In short, the methodology employed for this study involves two types of thresholds: the MLEV threshold used to identify negative examples and the threshold that is applied to the basic comparison methods, MLEV and MALINE, to identify matches. To avoid confusion, the term *negative threshold* refers to the former, while the term *positive threshold* is used for the latter.

The basic comparison methods were used as baselines in this research. To be able to provide a fair basic comparison score at each negative threshold, we "trained" each basic comparison matcher at twenty different positive thresholds on the same training set used by the learner. For each negative threshold, we picked the positive threshold that gave the best performance on the training data, and used that to score the matcher on the same test data as used by the learner.

3.2 Feature Extraction

Consider the MLEV alignment in Table 1. It can be seen that the first three characters are matched identically across both strings; after that, we get an "e" inserted, an "n" inserted, a "b" matched identically, a "u" matched to an "o", a "r" deleted, a "t" matched to a "d", an "o" matched to a "u", and an "n" deleted. The match unigrams are thus "a:a", "s:s", "h:h", "-:e", "-:n", "b:b", "u:o", "r:-", "t:d", "o:u", and "n:-". Match bigrams were generated by considering any insertion, deletion, and (non-identical) substitution unigram, and noting the unigram, if any, to its left, prepending that left unigram to it (delimited by a comma). Thus, the match bigrams in the above example include "h:h,-:e", "-:e,-:n", "b:b,u:o", "u:o,r:-", "r:-,t:d", "t:d,o:u", "o:u,n:-".

These match unigram and match bigram features are generated from just a single MLEV match. The composite feature set is the union of the complete match unigram and bigram feature sets. Given the composite feature set, each match pair is turned into a feature vector consisting of the following features: string1, string2, the match score according to each of the basic comparison matchers (MLEV and MALINE), and the Boolean value of each feature in the composite feature set.

3.3 Data Set

Our data is a (roughly 470,000 pair) subset of the Chinese-English personal name pairs in LDC 2005T34. About 150,000 of the pairs had more than 1 way to pronounce the English and/or Chinese. For these, to keep the size of the experiments manageable from the point of view of training the learners, one pronunciation was randomly chosen as the one to use. (Even with this restriction, a minimum negative threshold results in over half a million examples). Chinese characters were mapped into Hanyu Pinyin representations, which are used for MLEV alignment and string comparisons. Since the input to MALINE uses a phonemic representation that encodes phonemic features in one or more letters, both Pinyin and English forms were mapped into the MALINE notation.

There are a number of slightly varying ways to map Pinyin into an international pronunciation system like IPA. For example, (Wikipedia 2006) and (Salafra 2006) have mappings that differ from each other and also each of these two sources have changed its mapping over time. We used a version of Salafra from 2006 (but we ignored the ejectives). For English, the CMU pronouncing dictionary (CMU 2008) provided phonemic representations that were then mapped into the MALINE notation. The dictionary had entries for 12% of our data set. For the names not in the CMU dictionary, a simple grapheme to phoneme script provided an approximate phonemic form. We did not use a monolingual mapping of Chinese characters (Mandarin pronunciation) into IPA because we did not find any.

Note that we could insist that all pairs in our dataset be distinct, requiring that there be exactly one match for each Roman name and exactly one match for each Pinyin name. This in our view is unrealistic, since large corpora will be skewed towards names which tend to occur frequently (e.g., international figures in news) and occur with multiple translations. We included attested match pairs in our test corpora, regardless of the number of matches that were associated with a member of the pair.

4 Results

A variety of machine learning algorithms were tested. Results are reported, unless otherwise indicated, using SVM Lite, a Support Vector Machine (SVM[4]) classifier[5] that scales well to large data sets.

Testing with SVM Lite was done with a 90/10 train-test split. Further testing was carried out with the weka SMO SVM classifier, which used built-in cross-validation. Although the latter classifier didn't scale to the larger data sets we used, it did show that cross-validation didn't change the basic results for the data sets it was tried on.

4.1 Machine Learning with Different Feature Sets

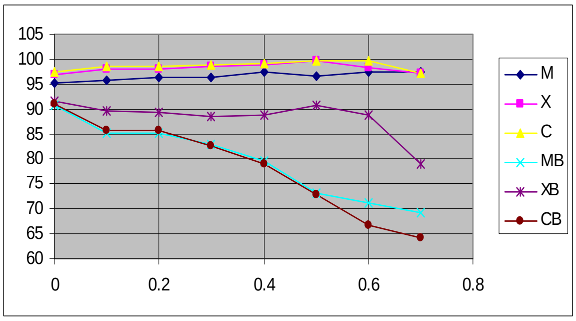

Figure 1: F-measure with Different Feature Sets

Figure 1 shows the F-measure of learning for monolingual features (M, based on MALINE), cross-lingual features (X, based on MLEV), and a combined feature set (C) of both types of features[6] at different negative thresholds (shown on the horizontal axis). Baselines are shown with the suffix B, e.g., the basic MALINE without learning is MB. When using both monolingual and cross-lingual features (C), the baseline (CB) is set to a system response of "true" only when both the MALINE and MLEV baseline systems by themselves respond "true". Table 2 shows the number of examples at each negative threshold and the Precision and Recall for these methods, along with baselines using the basic methods shown in square brackets.

The results show that the learning method (i) outperforms the baselines (basic methods), and (ii) the gap between learning and basic comparison widens as the problem becomes harder (i.e., as the threshold is raised).

For separate monolingual and cross-lingual learning, the increase in accuracy of the learning over the baseline (non-learning) results[7] was statistically significant at all negative thresholds except 0.6 and 0.7. For learning with combined monolingual and cross-lingual features (C), the increase over the baseline (non-learning) combined results was statistically significant at all negative thresholds except for 0.7.

In comparing the mono-lingual and cross-lingual learning approaches, however, the only statistically significant differences were that the cross-lingual features were more accurate than the monolingual features at the 0 to 0.4 negative thresholds. This suggests that (iii) the monolingual learning approach is as viable as the cross-lingual one as the problem of confusable names becomes harder.

However, using the combined learning approach (C) is better than using either one. Learning accuracy with both monolingual and cross-lingual features is statistically significantly better than learning with monolingual features at the 0.0 to 0.4 negative thresholds, and better than learning with cross-lingual features at the 0.0 to 0.2, and 0.4 negative thresholds.

[4] We used a linear kernel function in our SVM experiments; using polynomial or radial basis kernels did not improve performance.

[5] From svmlight.joachims.org.

[6] In Figure 1, the X curve is more or less under the C curve.

[7] Statistical significance between F-measures is not directly computable since the overall F-measure is not an average of the F-measures of the data samples. Instead, we checked the statistical significance of the increase in accuracy (accuracy is not shown for reasons of space) due to learning over the baseline. The statistical significance test was done by assuming that the accuracy scores were binomials that were approximately Gaussian. When the Gaussian approximation assumption failed (due to the binomial being too skewed), a looser, more general bound was used (Chebyshev's inequality, which applies to all probability distributions). All statistically significant differences are at the 1% level (2-sided).

4.2 Feature Set Analyses

The unigram features reflect common correspondences between Chinese and English pronunciation. For example, (Sproat, Tao and Zhai 2006) note that Chinese /l/ is often associated with English /r/, and the feature l:r is among the most frequent unigram mappings in both the MLEV and MALINE alignments. At a frequency of 103,361, it is the most frequent unigram feature in the MLEV mappings, and it is the third most frequent unigram feature in the MALINE alignments (56,780).

Systematic correspondences among plosives are also captured in the MALINE unigram mappings. The unaspirated voiceless Chinese plosives /p,t,k/ contrast with aspirated plosives /p^h,t^h,k^h/, whereas the English voiceless plosives (which are aspirated in predictable environments) contrast with voiced plosives /b,d,g/. As a result, English /b,d,g/ phonemes are usually transliterated using Chinese characters that are pronounced /p,t,k/, while English /p,t,k/ phonemes usually correspond to Chinese /p^h,t^h,k^h/. The examples of *Stewart* and *Elizabeth* in Section 1 illustrate the correspondence of English /t/ and Chinese / t^h/ and of English /b/ with Chinese /p/ respectively. All six of the unigram features that result from these correspondences occur among the 20 most frequent in the MALINE alignments, ranging in frequency from 23,602 to 53,535.

Negative Threshold	Examples	Monolingual (M)		Cross-Lingual (X)		Combined (C)	
		P	R	P	R	P	R
0	538,621	94.69 [90.6]	95.73 [91.0]	96.5 [90.0]	97.15 [93.4]	97.13 [90.8]	97.65 [91.0]
0.1	307,066	95.28 [87.1]	96.23 [83.4]	98.06 [89.2]	98.25 [89.9]	98.4 [87.6]	98.64 [84.1]
0.2	282,214	95.82 [86.2]	96.63 [84.4]	97.91 [88.4]	98.41 [90.3]	98.26 [86.7]	98.82 [84.7]
0.3	183,188	95.79 [80.6]	96.92 [85.3]	98.18 [86.3]	98.8 [90.7]	98.24 [80.6]	99.27 [84.8]
0.4	72,176	96.31 [77.1]	98.69 [82.3]	97.89 [91.8]	99.61 [86.2]	98.91 [77.1]	99.64 [80.9]
0.5	17,914	94.62 [64.6]	98.63 [84.3]	99.44 [89.4]	100.0 [91.9]	99.46 [63.8]	99.89 [84.7]
0.6	2,954	94.94 [66.1]	100 [77.0]	98.0 [85.2]	98.66 [92.8]	99.37 [61.3]	100.0 [73.1]
0.7	362	95.24 [52.8]	100 [100.0]	94.74 [78.9]	100.0 [78.9]	100.0 [47.2]	94.74 [100.0]

Table 2: Precision and Recall with Different Feature Sets (Baseline scores in square brackets)

4.3 Comparison with other Learners

To compare with other machine learning tools, we used the WEKA toolkit (from www.weka.net.nz). Table 3 shows the comparisons on the MLEV data for a fixed size at one threshold. Except for SVM Light, the results are based on 10-fold cross validation. The other classifiers appear to perform relatively worse at that setting for the MLEV data, but the differences in accuracy are not statistically significant even at the 5% level. A large contributor to the lack of significance is the small test set size of 66 pairs (10% of 660 examples) used in the SVM Light test.

4.4 Other Language Pairs

Some earlier experiments for Arabic-Roman comparisons were carried out using a Conditional Random Field learner (CRF), using the Carafe toolkit (from sourceforge.net/projects/carafe). The method computes its own Levenshtein edit-distance scores, and learns edit-distance costs from that. The scores obtained, on average, had only a .6 correlation with the basic comparison Levenshtein scores. However, these experiments did not return accuracy results, as ground-truth data was not specified for this task.

Several preliminary machine learning experiments were also carried out on Urdu-Roman comparisons. The data used were Urdu data extracted from a parallel corpus recently produced by the LDC (LCTL_Urdu.20060408). The results are shown in Table 4. Here a .55 MALINE score and a .85 MLEV score were used for selecting positive examples by basic comparison, and negative examples were selected at random. Here the MALINE method (row 1) using the weka SMO SVM made use of a threshold based on a MALINE score. In these earlier experiments, machine learning does not really improve the system performance (F-measure decreases with learning on one test and only increases by 0.1% on the other test). However, since these earlier experiments did not benefit from the use of different negative thresholds, there was no control over problem difficulty.

5 Related Work

While there is a substantial literature employing learning techniques for record linkage based on the theory developed by Fellegi and Sunter (1969), researchers have only recently developed applications that focus on name strings and that employ methods which do not require features to be independent (Cohen and Richman 2002). Ristad and Yianilos (1997) have developed a generative model for learning string-edit distance that learns the cost of different edit operations during string alignment. Bilenko and Mooney (2003) extend Ristad's approach to include gap penalties (where the gaps are contiguous sequences of mismatched characters) and compare this generative approach with a vector similarity approach that doesn't carry out alignment. McCallum et al. (2005) use Conditional Random Fields (CRFs) to learn edit costs, arguing in favor of discriminative training approaches and against generative approaches, based in part on the fact that the latter approaches "cannot benefit from negative evidence from pairs of strings that (while partially overlapping) should be considered dissimilar". Such CRFs model the conditional probability of a label sequence (an alignment of two strings) given a sequence of observations (the strings).

A related thread of research is work on automatic transliteration, where training sets are typically used to compute probabilities for mappings in weighted finite state transducers (Al-Onaizan and Knight 2002; Gao et al. 2004) or source-channel models (Knight and Graehl 1997; Li et al. 2004). (Sproat et al. 2006) have compared names from comparable and contemporaneous English and Chinese texts, scoring matches by training a learning algorithm to compare the phonemic representations of the names in the pair, in addition to taking into account the frequency distribution of the pair over time. (Tao et al. 2006) obtain similar results using frequency and a similarity score based on a phonetic cost matrix

The above approaches have all developed special-purpose machine-learning architectures to address the matching of string sequences. They take pairs of strings that haven't been aligned, and learn costs or mappings from them, and once trained, search for the best match given the learned representation

Positive Threshold	Examples	Method	P	R	F	Accuracy
.65	660	SVM Light	90.62	87.88	89.22	89.39
.65	660	WEKA SMO	80.6	83.3	81.92	81.66
.65	660	AdaBoost M1	84.9	78.5	81.57	82.27

Table 3: Comparison of Different Classifiers

Method	Positive Threshold	Examples	P	R	F
WEKA SMO	.55 (MALINE)	206 (MALINE)	84.8 [81.5]	86.4 [93.3]	85.6 [87.0]
WEKA SMO	.85 (MLEV)	584 (MLEV)	89.9 [93.2]	94.7 [91.2]	92.3 [92.2]

Table 4: Urdu-Roman Name Matching Results with Random Negatives (Baseline scores in square brackets)

Our approach, by contrast, takes pairs of strings *along with an alignment*, and using features derived from the alignments, trains a learner to derive the best match given the features. This offers the advantage of modularity, in that any type of alignment model can be combined with SVMs or other classifiers (we have preferred SVMs since they offer discriminative training). Our approach allows leveraging of any existing alignments, which can lead to starting the learning from a higher baseline and less training data to get to the same level of performance. Since the learner itself doesn't compute the alignments, the disadvantage of our approach is the need to engineer features that communicate important aspects of the alignment to the learner.

In addition, our approach, as with McCallum et al. (2005), allows one to take advantage of both positive and negative training examples, rather than positive ones alone. Our data generation strategy has the advantage of generating negative examples so as to vary the difficulty of the problem, allowing for more fine-grained performance measures. Metrics based on such a control are likely to be useful in understanding how well a name-matching system will work in particular applications, especially those involving confusable names.

6 Conclusion

The work presented here has established a framework for application of machine learning techniques to multilingual name matching. The results show that machine learning dramatically outperforms basic comparison methods, with F-measures as high as 97.0 on the most difficult problems. This approach is being embedded in a larger system that matches full names using a vetted database of full-name matches for evaluation.

So far, we have confined ourselves to minimal feature engineering. Future work will investigate a more abstract set of phonemic features. We also hope to leverage ongoing work on harvesting name pairs from web resources, in addition applying them to less commonly taught languages, as and when appropriate resources for them become available.

References

Al-Onaizan, Y. and K. Knight, K. 2002. Machine Transliteration of Names in Arabic Text. Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages.

Bilenko, M. and Mooney, R.J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of SIGKDD-2003.*

CMU. 2008. The CMU Pronouncing nary. ftp://ftp.cs.cmu.edu/project/speech/dict/

Cohen, W. W., and Richman, J. 2002. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002).*

Fellegi, I. and Sunter, A. 1969. A theory for record linkage. *Journal of the American Statistical Society,* 64:1183-1210, 1969.

Freeman, A., Condon, S. and Ackermann, C. 2006. Cross Linguistic Name Matching in English and Arabic. *Proceedings of HLT.*

Gao, W., Wong, K., and Lam, W. 2004. Phoneme-based transliteration of foreign names for OOV problem. In *Proceedings of First International Joint Conference on Natural Language Processing.*

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 (ANLP-NAACL 2000), 288-295.

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

Li, H., Zhang, M., & Su, J. 2004. A joint source-channel model for machine transliteration. In *Proceedings of Conference of the Association for Computation Linguistics (ACL).*

McCallum, A., Bellare, K. and Pereira, F. 2005. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance. *Conference on Uncertainty in AI (UAI).*

Ristad, E. S. and Yianilos, P. N. 1998. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence.*

Salafra. 2006. http://www.safalra.com /science /linguistics/pinyin-pronunciation/

Sproat, R., Tao, T. and Zhai, C. 2006. Named Entity Transliteration with Comparable Corpora. In *Proceedings of the Conference of the Association for Computational Linguistics*. New York.

Tao, T., Yoon, S. Fister, A., Sproat, R. and Zhai, C. 2006. Unsupervised Named Entity Transliteration Using Temporal and Phonetic Correlation. In *Proceedings of the ACL Empirical Methods in Natural Language Processing Workshop*.

Wikipedia. 2006. http://en.wikipedia.org/wiki/Pinyin