Multilingual Name Mapping

Tatjana Scheffler
University of the Saarland
tasc@coli.uni-sb.de

May 22, 2002

Abstract

This paper addresses the problem of coreferences between European names and their transcription into Chinese.

Name processing plays an important role in multilingual text processing. As a first important step in order to cross-lingually process names, relations between named entities across the languages must be established or rejected.

Some big dictionaries of name transcriptions e.g. into Chinese exist, but they are not sufficient for the task as new names are formed each day. In contrast, a dynamic tool implementing our intuition of "phonological regularities" behind the process of "name mapping" is needed.

We propose the use of support vector machines (SVMs) in order to approach the task. SVMs are highly parametrizable, trainable tools adaptable to various sorts of problems. We show that their application to our task, despite first, non-evolved heuristics, yields good results, averaging an f-measure of up to 0.76. We also give examples of the vast possibilities of further improving the performance.

1 Introduction

1.1 Multilingual Name Processing

The processing of proper nouns, e.g. personal, company or place names, constitutes an important part in modern systems for text processing. Currently, more and more applications have to deal with *multilingual* text sources, like in projects on multilingual information extraction. In such contexts, name processing faces special difficulties, as it encounters proper names transcribed into different scripts.

By now, name processing mainly comprises Named Entity Recognition, sometimes followed by some easy inferences helping to establish coreferences between the names. However, for multilingual name processing, it is of great importance to find respectively reject possible coreferences between proper names in texts in either languages.

A big part of this problem is equivalent to the task of finding coreferences within a text or between texts in *one* language. This part comprises the resolution of coreferent entities like "Walt Disney" and "Mr. Disney" (or e.g. in a German text, "Herr Disney"). We will not deal with these questions, which have been addressed occasionally in the past. Instead, we concentrate on the special problems which arise when a name is transcribed into a different script. Then, the name appears e.g. as 华特 迪斯尼. Its relation to "Walt Disney" is not easily recognizable, even if one transforms it into its standard romanization (Hanyu Pinyin) "Huate Disini".

To resolve these kinds of cross-lingual coreferences (or as we will further call it, *name mappings*) is obviously the first, necessary step in finding relations between texts of different language relying on or concerning proper names.[1]

European-to-Chinese Transcription

Name mapping resolution is inevitable in applications, where the same names can be found in different texts, written in different scripts. For the case of romanization, transcription standards often exist, but not always for the reverse direction. For example, no common standard is used for the transcription of European names into Chinese characters.

Transcription into Chinese follows two different principles, *semantic transcription* (finding "good characters") and *phonetic transcription*. Although for company names the first principle is of great importance (see Schmitt and Zhang [12]), the phonetic paradigm might be the most significant. For human beings, the sound is the most important when identifying the same name in different scripts.

1.2 Task

This paper addresses the problem of coreferences between European names and their transcription into Chinese. In Multilingual Information Extraction, a certain number of named entities is recognized in each text. To find cross-lingual coreferences between these named entities, we must solve the following problem:

Goal 1 (General Case) *For two given sets of named entity candidates, which ones of the first set are possible transcriptions of which names in the second set?*

Such information is necessary for further processing of multilingual corpora in information extraction or other systems that need to use alignment. Once the name mappings have been established, they can be used for content-based alignment of texts. Proper names as indicators are especially useful when reliable deep textual analysis is not available, which is usually the case.

Furthermore, making use of large dictionaries is not sufficient for the task. While some transcription dictionaries exist (e.g. English given names and common place names), they can never be exhaustive. Some types of proper names, e.g. company names, even form an open class, where new names are built each day. In addition, name transcription in the Chinese mainland, Hongkong and Taiwan is not always identical, even if one abstracts over differences originating from Traditional vs. Simplified Chinese inequalities. Finally, actual use of transcriptions in newspapers etc. is not necessarily consistent. The same company can even be transcribed in two different ways in the same article:

> 看准移动内容服务 美**迪斯尼**集团投资香港互联网
> http://www.sina.com.cn 2001年11月27日 10:19 新华网
> 新华网香港11月27日电 最近，美国华特·**迪士尼**互联网集团在香港推出新的投资项目 [...] [15]

Here, "Disney" is written twice, with a different character for "si" in the heading than in the rest of the article.

This effect is even worse for uncommon names. All these factors call for the use of a generative module, that can be applied alone or in addition to the available name dictionaries.

[1] Note that for tasks like Topic Detection and Tracking, relying on proper names can be of great help, as they may occur rarely in the background (corpus), but very often in one or some relevant topics.

1.3 Previous Work

To our knowledge, no previous attempts on our above goal have been made in the fields of information extraction or multilingual text processing. Research in name processing has been concentrating mainly on Named Entity Recognition, and in a wider sense, on coreference resolution for named entities.

However, research on the phonetic similarity (of proper nouns) has been done. Closest to our topic, Richard Lutz and Stephan Greene of Language Analysis Systems, Inc. (LAS) have built an industrial application, making use of phonological information in automatic name searching.

Lutz and Greene address the task of finding all similar names, ordered by similarity, to a certain input in a database ([7]). They take a knowledge engineering approach, computing a similarity measure based on phonological features of the names' sounds. They give two different sets of weights they assigned to "differences" in the names' phonological representations (or two "Edit Distance Cost Tables"). The first assigns equal weights to each feature violation (taking the features from Chomsky and Halle's *Sound Pattern of English* [2]). The second set varies the weights according to an adaptation of the *Optimality Theory* (OT, Prince and Smolensky, 1993; [9]). Here too, the actual weights are assigned manually, as the constraints posed by OT do not unambiguously identify the weights. Furthermore, some inaccuracies following from the constraints have to be corrected by hand.

The *goal* of name processing in the above application differs from ours in several ways:

- monolinguality (within English only),
- generativity (generate all similar names),
- use for *searching* (overgeneration hurts much less than undergeneration).

Thus, it is not directly applicable to our task.

While the approach takes into account some phonological regularities, it strongly depends on human analysis of the problem. Therefore, the proposed method is not easily adaptable to (noisy) data, e.g. other kinds of names. For example for company names, the transcription process might be more flexible than for personal names, allowing for more phonetic differences when trying to find "semantically positive" Chinese characters.

Furthermore, improvement of the system output is not easily or monotonically obtainable. Manual changes to the Edit Distance Cost Tables require great professional knowledge and can also cause decrease in the overall performance. For these reasons, the system is suboptimal for application to goal 1. However, in section 4.1 we propose several fundamental changes to the approach that could adapt it to our goal.

2 Approach

Our approach employs a statistical method based on supervised machine learning. We first reformulated the general goal to the equivalent:

Goal 2 *For a pair of names, decide whether they can form a cross-lingual mapping.*

2.1 Requirements

The module we envisage is to be used as a online preprocessing tool. It must therefore fulfill the following requirements:

efficiency Offline training is possible. But as an online tool in a preprocessing stage (where other modules may depend on its output), actual application of the system must be very efficient.

adaptability The system should be applied to different kinds of proper nouns that are transcribed into Chinese. Therefore, it must include means in order to adapt itself to new domains.

Adaptability also includes that the tool should be applyable not only to the transcription of English names into Chinese, but also of German, French, etc. proper nouns, given some small variations.

easy trainability Adaptation of the system should not call for annotation of large amounts of data, the system should be easily re-trainable after changes in the specifications.

modularity & re-usability Through good separation of means and training data, large parts of the system should be re-usable in the case of a change.

improvability The system should provide parts where its performance can be improved without the need for a whole new approach to the problem.

sortable output (not only a T/F-decision) As the main use lies not in a standalone application, but a preprocessing tool whose results are used by later modules, it can be necessary to rank the outputs in further processing. Therefore, the decision (of whether it can be a mapping, or not) should not be on a true/false basis, but rather give a probability measure of "mapping-ness".

For our first experiments, which concentrated on the applicability of statistical methods to the problem, we also set up a secondary requirement:

availability The statistical method should be easily obtainable from a public domain. It furthermore must be parametrizable to be adapted to our application.

2.2 Methods

We chose to use Support Vector Machines (SVMs [14],[3]) to approach the problem. SVMs are a relatively new technique in machine learning. In the field of computational linguistics, they have been successfully applied to text classification (see Joachims, 1998 [6]).

Support Vector Machines

SVMs learn a function classifying their input data into positive or negative according to a set of training examples. The examples are vectors in a multidimensional vector space, within which the basic form of SVMs learns an optimal soft margin hyperplane separating positive and negative data points.

Through the use of *kernels*, the complexity of the learned function can be adapted to the task. SVMs can use a function ϕ mapping the training data into a higher order vector space, where the hyperplane is computed. Reversely mapping the hyperplane into the original, lower order vector space results in a polynomial or gaussian classifying function, depending on the used kernel (ϕ). Furthermore, kernels can be parametrized as to the order of the polynomial resp. σ for the radial base function.

Why use SVMs?

SVMs are a good means for classifying data. They can be applied to a variety of problems. For our task, SVMs show a great number of virtues. The main need for time and space is located in the training phase. Once the machine has been successfully trained (the classifying function learned), its application to actual data is very efficient.

For training, SVMs need a set of positive and negative examples. One very important benefit of SVMs is that even if the training set is much smaller than the number of features, good results can be achieved. Other than the indication of whether an example is positive or negative, no annotation of the data is needed. This makes adaptation to new kinds of data convenient, as it only requires a new (small) set of examples.

Furthermore, SVMs learn an internal function. This reflects our intiution of phonological *regularities* leading the transcription of English names into Chinese. The broad applicability of SVMs allows us also to test our expectation that these are in fact *phonological* regularities. The task can be performed with or without first transforming strings into their phonemic representations. We expect the latter will perform better, as orthography blurrs the relations between sounds.

Through employment of different kernels, support vector machines are highly parametrizable. The class and complexity of the target function can be specified. Furthermore, a parameter C accounts for how close the function shall fit for the training data, or how many classification errors may be acceptible for the gain of a less complex result.

Finally, output of SVMs is a number x with $-\infty \leq x \leq \infty$. Usually, 0 is used as a classification boundary, and the machine classifies test data into positive or negative. If a ranking is needed, though, x's absolute value can be used as an indicator of reliability.

2.3 Design

The first intention of our research was to study the applicability of a statistical, trainable method to the perceived problem. The general design is therefore very straightforward. We have also put up with many abstractions in order to obtain first results on our intuition that name transcription follows phonological regularities.

Within the general design, however, many improvements are still possible. If some of our necessary (and, admittedly, very simplifying) abstractions are further developed, improvement of the overall performance can also be expected.

General Design

The input of the module constitutes a pair of words, a Chinese and an English name candidate. As support vector machines work on vectors, the input word pair must be transformed into a single vector, first. Output is a classification value c with $c = -1$ or $c = 1$.

For the SVM, we used the implementation by Ryan Rifkin at MIT, SvmFu ([10], or see [11] for download and documentation). SvmFu is an implementation in C++, which is not only very fast, but also easy to use.

Implementation

We tested our approach on a set of 476 English personal names and their Chinese transcriptions ([13]). The transformation of a name pair into one vector representation is performed in two steps. First we obtain a vector representation from each word. Then we combine the two vectors into one.

Vector representations are computed in two different ways:

1. We obtain the unicode number for each character in the string, and compute its bit representation. Appending all the bits determines the vector.
2. First, we retrieve a phonemic representation of each word. Then, we convert the phonemic representation into a vector following method 1. For English, Bertjan Busser provided us with the phonemic representations of the complete set of names by applying on them his grapheme-phoneme conversion system TreeTalk ([1]). For Chinese, we used a tool adding Pinyin to files ([8]). Of course, Pinyin is not a phonemic representation. But for our purpose, using Pinyin may be a permissible first approximation, as Pinyin pronunciation is very regular (in contrast to, e.g., English pronunciation).

We expected better results from the SVMs using the second kind of input. The vectors retrieved from the unicode encoded strings actually incorporate two kinds of arbitrary relations. The first one is orthography, that equal sounds can be actually written in very different ways (see "Lewis" and "Louis" as an example). The second one is the unicode numbering: similar sounds do not necessarily get similar unicode numbers. By first transforming the strings into their phonemic representations, we can remove the first kind of noise. Nevertheless, using ascii numbers as the basis of our vectors still includes a certain amount of arbitrariness. How to avoid this abstraction is one of the further tasks.

For combination of the two single-word vectors, in the current version we just appended them, filling the first part of the vector up to a certain length (thus accounting for differences in the length of names, which are irrelevant).

Example Sets

For the experiments, we need separate training and testing sets. To obtain them, we apply the Leave-One-Out method on our name list (leaving out every 10th line in the alphabetically ordered list), producing 10 sets of 428 (429) positive training examples and 48 (47) positive testing examples.

A major problem was the necessity of negative training examples. Obviously, no lists of how names should *not* be transcribed into Chinese exist. We employed a simple heuristics, building negative examples out of our positive ones. In each example file, we shifted the English names by 13 lines, thus creating new pairs from the existing names. Actually, it is only by luck that in such a way we did not create contradictory data, because some very similar-sounding names (which can be written very differently in English) can have the same transcription into Chinese. For the phonemic case, duplicates may also exist.

As bad negative examples may harm system performance, this leaves great room for improvement. Negative examples too close to the positive examples may result in too conservative classification functions, while too far negative examples can cause overgeneration.

3 Results

3.1 Experiments

We conducted a set of experiments testing our approach on the set of names. For this, we varied some of the SVM parameters, in order to find out which one could work best for our application. Preliminary tests suggested the use of a linear kernel would not yield good results. The complexity of the problem (besides other factors also the complexity introduced by simply appending the two word vectors) indicates that a satisfactory linear solution may not be possible.

Instead, we performed tests using polynomial as well as radial base (also called gaussian) kernels. Each test was taken once with the unicode encoded data set, and once with the data in "phonemic representations", as described above. For the polynomial kernel, we tested degrees 2 and 3, and additionally 4 in the phonemic case. The gaussian kernel promised slightly better results, we tested it with various σ. The results are shown in table 1 for the unicode tests and table 2 for the phonemic ones.[2]

	polyn.		gaussian									
	2	3	1	1.5	2	2.1	2.2	2.3	2.4	2.5	3	10
cor.	.66	.68	.61	.64	.70	.70	.71	**.72**	.72	.72	.71	.50
prc.	.68	.70	.56	.60	.70	.71	.72	.75	.75	.75	**.76**	.50
rec.	.62	.62	**.96**	.81	.72	.70	.68	.68	.67	.65	.60	**1**
f	.65	.66	.71	.69	.71	.70	.70	**.71**	.71	.70	.67	.67

Table 1: Results for unicode input.

	polynomial			gaussian									
	2	3	4	1	1.5	2	2.1	2.2	2.3	2.4	2.5	3	10
cor.	.75	**.77**	.76	.63	.69	.76	**.78**	.77	.77	.77	.77	.75	.50
prc.	.79	.82	.82	.58	.65	.78	.81	.81	.82	.83	.83	**.83**	.50
rec.	.69	.69	.68	**.95**	.82	.74	.72	.71	.70	.69	.68	.64	**1**
f	.74	.75	.74	.72	.72	.76	**.76**	.76	.75	.75	.75	.72	.67

Table 2: Results for phonemic input.

3.2 Discussion

For all parameter settings, SVM training and testing was performed for each of the 10 testsuites. The given numbers are the rounded means over the testsuites. The best values within each line are set in bold face. Furthermore, some notable values are also set in bold face, namely the almost perfect recall for gaussian kernel with $\sigma = 1$ and the also notable correctness for the phonemic 3rd degree polynomial kernel (it almost reaches the best correctness value).

For both input types, a gaussian kernel performed better than the best polynomial kernel. With the unicode encoded input, the differences are very significant, about five percent points for both f-measure and correctness. In the phonemic case, the best result is still achieved using a gaussian kernel, but the polynomial kernel in the 3rd degree comes very close.

Our expectation that phonemic input will perform better than unicode encoded input, has been met by the experiments. A significant improvement in system performance can be noted if the names are transformed into their phonemic representations first. The improvement averages 5 f-measure points for gaussian kernels and almost 10 for polynomial kernels.

[2]**Legend:**
cor. = percentage of correct classifications
prc. = precision; $\frac{\text{correct positive classifications}}{\text{positive classifications}}$
rec. = recall; $\frac{\text{correct positive classifications}}{\text{all positives}}$
f = f-measure; $\frac{2*prc*rec}{prc+rec}$

Using a gaussian kernel, best results were achieved with $2 \leq \sigma \leq 2.5$. Too large σ can obviously overfit the data. As our training and test sets were strictly separate, this would result in bad recall. In contrast, too small σ leads to bad precision, as it is too permissive. Interestingly, recall is high on both ends of our tested scale. Further tests with other parameters are needed in order to determine whether this indicates the existence of other local maxima.

4 Conclusions

Our results show that statistical methods are applicable to the problem defined in goal 1. The use of support vector machines for classifying possible mappings of European names into Chinese proved feasible. In spite of the rather strong abstractions employed in our first design (which was aimed at using easy heuristics in order to quickly yield information about feasibility), the approach brought acceptable results.

One measure lessening the level of abstractions (performing – almost for the Chinese case – grapheme-to-phoneme conversion), resulted in a measurable improvement of the system. This indicates other suggested changes (see subsection 4.1) might further improve the performance.

We showed the application of SVMs to the task to be especially promising because of its easy adaptability. Tests indicated that a bigger training set (ours had less than 430 examples) is not necessary. For most name categories, extensive dictionaries already exist, which could be used as training input. Further tests are needed in order to find out whether other name categories (other than personal first names) can be classified using the same SVMs or whether for each category an own SVM should be trained, using examples from their field.

4.1 Future Work

The presented approach leaves great room for improvements. As noted above, our heuristics producing the negative training (and testing) examples are clearly suboptimal. A better way of producing them, or – even better – finding real examples from texts, could be a method to improve performance.

The process of converting the two name strings into one vector still involves many abstractions. Not only that the phonemic representation of the strings is still converted into a vector by taking the characters' encoding number into account, the two vectors are then combined into one by simply appending them. Of course this does not reflect the nature of their relation.

A third point of possible improvements is of computational character. Our tests have not yet been extensive. The best received results could therefore reflect a local maximum of SVM performance given our input data. Therefore, extensive further tests with more variation of the parameters are necessary. A different possibility is to try to automatically adapt the kernel parameters to our data, using a method as e.g. the one proposed by Cristianini et al. in 1998 [4].

Finally, we plan to integrate the different processing steps into one standalone tool, which can be used within multilingual information extraction systems or other related applications.

Using features By now, there are no comparable results of other approaches to the same task. Further work lies in the adaptation of the mentioned approach to a similar task using phonological similarity and features by Lutz and Greene [7]. The names could be converted into their feature representations, and the algorithm mentioned by Lutz and Greene could be applied.

An important change would probably have to be made in the "Edit Distance Cost Table". As the weights were manually adapted to their (slightly different) task, they might not yield optimal results for ours. Instead, one should attempt to learn the optimal weights for each "transition" automatically using an example set. Thus, we can account for different types of names with different "tolerance" in the transcription process, as described above for the case of company names.

By encoding sounds as feature vectors and applying the adaptation of Hall and Dowling's *edit distance algorithm* ([5]) described in [7], one obtains a vector of (numerically encoded) necessary changes to get from the English name to the Chinese one. Following a suggestion by Sebastian Padó, one can give these "change vectors" as input to an SVM (or maybe another method for statistical learning which is more suitable to the task), attempting to learn what changes are tolerable in a transcription process, and which ones result in a different name.

Acknowledgement

The original recognition of the problem is due to Markus Becker of DFKI Saarbrücken, with whom the author worked in a project on multilingual information extraction at Shanghai Jiaotong University, China.

References

[1] Busser, Bertjan, TreeTalk: Memory-Based Grapheme-Phoneme Conversion Demo; Online at http://ilk.kub.nl/g2b-www-demo.html

[2] Chomsky, Noam and Halle, Morris, 1968, *The Sound Pattern of English.* Harper and Row, New York, quoted in [7].

[3] Cortes, C. and Vapnik, V., Nov. 1995, Support-vector Networks. *Machine Learning*, 20:273-297, quoted in [6].

[4] Cristianini, N., Campbell, C. and Shawe-Taylor, J., 1998, *Dynamically Adapting Kernels in Support Vector Machines*, NeuroCOLT2 Technical Report Series.

[5] Hall, Patrick A. V. and Dowling, Geoff R., Dec. 1980, Approximate String Matching. *ACM Computing Surveys*, 12(4):381–402.

[6] Joachims, Thorsten, 1998, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.* Retrieved May 10, 2002 from http://www.cs.cornell.edu/People/tj/publications/joachims_98a.pdf

[7] Lutz, Richard and Greene, Stephan, 2001, *Measuring Phonological Similarity: The Case of Personal Names.* Retrieved May 10, 2002, from LAS Web Site: http://www.las-inc.com/nameinfo/wp_lsa.htm

[8] Peterson, Erik, n.d., DimSum: Adding Pinyin to Files. Retreived May 10, 2002, from http://www.mandarintools.com/dimsum.html

[9] Prince, Alan and Smolensky, Paul, 1993, *Optimality Theory: Constraint Interaction in Generative Grammar.* MIT Press, quoted in [7].

[10] Rifkin, Ryan and Nadermann, Michelle, 2000, SvmFu: Software For SVMs. Unpublished research abstract; Retrieved May 10, 2002, from http://www.ai.mit.edu/research/abstracts/abstracts2000/pdf/z-rifkin.pdf

[11] Rifkin, Ryan, SvmFu 3. Online documentation; http://five-percent-nation.mit.edu/SvmFu/

[12] Schmitt, Bernd and Zhang Shi, n.d., Global Naming, Unpublished research; Retreived May 10, 2002, from http://www.globalbrands.org/research/working/globanaming.pdf

[13] Shan, Jun, Chinese Names. List of popular English first names and their corresponding Chinese names by gender; Retreived May 10, 2002, from http://chineseculture.about.com/library/weekly/aa103199.htm and subpages.

[14] Vapnik, Vladimir, 1995, *The Nature of Statistical Learning Theory.* Springer, New York, quoted in [6].

[15] Xinhua News Agency, published on http://www.sina.com.cn (emphasis added)