Matching Person Names through Name Transformation

Jun Gong
Information System Department
Beihang University
Beijing, China
jungong@ymail.com

Lidan Wang
CS/UMIACS CLIP Lab
University of Maryland
College Park, Maryland USA
lidan@cs.umd.edu

Douglas W. Oard
iSchool/UMIACS CLIP Lab
University of Maryland
College Park, Maryland USA
oard@umd.edu

ABSTRACT

Matching person names plays an important role in many applications, including bibliographic databases and indexing systems. Name variations and spelling errors make exact string matching problematic; therefore, it is useful to develop methodologies that can handle variant forms for the same named entity. In this paper, a novel person name matching model is presented. Common name variations in the English speaking world are formalized, and the concept of name transformation paths is introduced; name similarity is measured after the best transformation path has been selected. Supervised techniques are used to learn a similarity function and a decision rule. Experiments with three datasets show the method to be effective.

Categories and Subject Descriptors
H.3.3 [Information Systems]: Information Search and Retrieval – *search process, clustering.*

General Terms
Algorithms, Performance, Languages, Experimentation

Keywords
Name matching, String similarity, String distance metrics

1. INTRODUCTION

In information retrieval and knowledge management, a user frequently encounters the problem of whether two strings refer to the same person. For example, a user may be confused by similar strings for person names when searching scientific literature and citation indices. The same problem occurs when people analyze citation statistics to assess the impact of various authors' work. Due to the variety of formats used to cite the same author in different publications, the spelling variants of different languages, and the misspellings that people often make, it is important to develop methodologies that can efficiently match names that refer to the same entity under these conditions.

The task of matching entity names has been well explored. Cohen and his colleagues [6] made a comprehensive study of known techniques and conducted a comparison of several string distance measures for the tasks of matching and clustering lists of named entities. In their experiments, Soft-TFIDF [6], Jaro-Winkler [14] and Monge-Elkan [11] did well. Subsequently, many researchers focused on person name matching problems under different conditions. Piskorski et al. [12] focused on knowledge-poor methods for a person name matching task in Polish, a highly inflected language, and Arehart and Miller [1] performed experiments with a multicultural Romanized name data set. Each optimized the matching methods for characteristics of specific datasets. Person name variation has also attracted the attention of digital library researchers. Christen [4] discussed the characteristics of person names and presented potential sources of variations and errors. Galvez and Moya-Anegon [9] classified person name variants as valid or invalid and presented finite-state graphs to match person names. Form the above, we can see that person name matching and person name variation have been well explored previously, but those two problems have to some extent been studied independently. In this paper, we aim to jointly study these problems by leveraging name variation features to perform name matching.

2. UNSUPERVISED NAME MATCHING

Before introducing our model, we first give a formal definition of person name and name format. In the English speaking world, a person name is a character string which can be tokenized into a token list. For each token in the list, it may be a single-character initial (*I*) or a multi-character "non-initial" token (*N*). We define **Person Name** (*Name*) and **Name Format** (*F (Name)*) as follows:

$$Name = (t_1,\ldots t_i \ldots t_n)$$

$$F(Name) = (f_1 \ldots f_i \ldots f_n),\ f_i \in \{I, N\}.$$

For example, the name "Michael Joe Penn" can be defined as (*"Michael", "Joe", "Penn"*) and its name format *F("Michael Joe Penn")*=(*N,N,N*). Unlike traditional methods, our model measures name similarity only after the name pair has been transformed into the same format. Next, we describe our model in detail.

2.1 Name Variation and Transformation Path

In the English speaking world we can identify three main name variation types: *abbreviation*, *omission* and *sequence changing*. *Abbreviation (A)*: a non-initial name token is rewritten as an initial; *Omission (O)*: certain tokens in the full name are omitted; *Sequence changing (S)*: some publications put the given name first; others put the family name first. Given these name variation types, we define a **Name Transformation Operation Set**: $V = \{A, O, S\}$. A person name is transformed into a variant by applying a sequence of one or more operations from *V*; the name format changes as a result of each operation. Figure 1 illustrates the process.

$$\begin{tikzcd}
\text{M. Joe Penn} \arrow[r, "A"] & \text{M. J. Penn} \arrow[r, "O"] & \text{M. Penn} \arrow[r, "S"] & \text{Penn M.} \\
(I, N, N) & (I, I, N) & (I, N) & (N, I)
\end{tikzcd}$$

Figure 1. Transforming the name format.

With appropriate transformations, two names (*A, B*) can be transformed into the same name format; we call the final format the **Matching Format** *(M)*. We define the transformations applied to *A* and *B* in this process the **Transformation Path** (*TP*):

$$TP_{AB} = (TP_{A\Rightarrow M}, TP_{B\Rightarrow M}, M),$$

$$TP_{A\Rightarrow M} = (v_{a1} \ldots v_{ai} \ldots v_{am}) \quad v_{ai} \in V$$

$$TP_{B\Rightarrow M} = (v_{b1} .. v_{bj} \ldots v_{bn}) \quad v_{bj} \in V.$$

Furthermore, the similarity between two transformed names can be calculated; we call this the **Matching Similarity** and denote it $Sim(N_A,N_B|TP)$. From the definition, it is clear that there can be multiple transformation paths between a pair of names. In Figure 2, the names "Michael John Pfeng" and "M. Joe Penn" reach the matching format through multiple paths. Table 1 shows the corresponding transformation paths; the matching similarity is calculated with the Jaro metric [14].

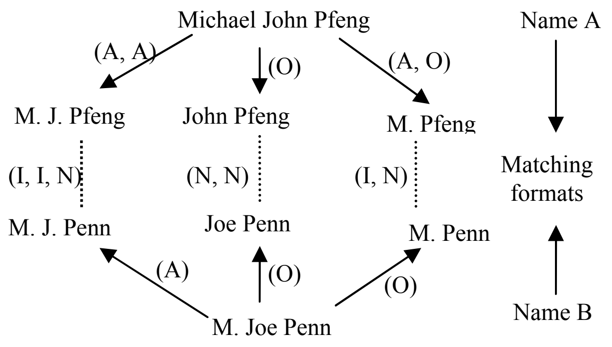


Figure 2. The Name transformation paths.

Table 1. Possible transformation paths for matching.

NameA	NameB	Transformation Path	Similarity
M J Pfeng	M J Penn	*((A,A), (A), (I,N,N))*	0.884
John Pfeng	Joe Penn	*((O), (O), (N,N))*	0.763
M Pfeng	M Penn	*((A,O), (O), (I,N))*	0.849
……	……	……	……

Which path is optimal? We next present a method to estimate that.

2.2 Selecting Path using Graph Theory

In Figure 2, the name variants and transformation paths compose a directed graph, which suggests that a good path can be selected using Dijkstra's algorithm [8]. We prefer the shortest path; when there are multiple shortest paths, we prefer the one with maximum string similarity between the final versions of the two names. Assigning a weight (w_i) to each transformation operation $v_i \in (A, O, S)$, we formulate path selection as:

$$\text{Step1: } TP^{*} = \arg\min \sum_{v_i \in TP} w_i(v_i)$$

$$\text{Step2: } TP^{\wedge} = \arg\max Sim(N_A, N_B \mid TP^{*}).$$

This method is based on the assumption that excessive transformation should be avoided when a pair of names is being compared. We call this method STP.

3. SUPERVISED NAME MATCHING

The STP method calculates the similarity of a person name pair after selecting the best name transformation path. However, some name pairs that should not match nonetheless achieve high similarity after a sequence of name transformations. We therefore should also integrate supervised learning into our matching model.

3.1 Learning a Weighted Similarity Function

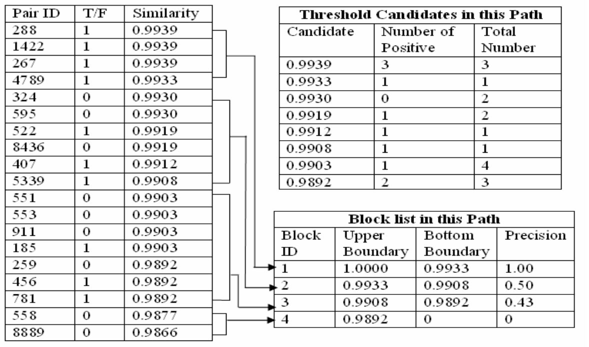
Some transformation paths are simply less reasonable than others. In order to estimate this, each transformation path is assigned a weight *W*(*TP*). We now define a new similarity measure that takes transformation path weights into account:

$$Sim^{*}(N_i, N_j \mid TP) = W(TP) * Sim(N_i, N_j \mid TP).$$

Let $W_1, \ldots W_n$ denote the respective weights for the transformation paths $TP_1, \ldots TP_n$ under consideration. To compute optimal weights $\{W_i\}$, we formulate the following optimization problem:

$$\{W_i\}^{opt} = \arg\max_{\{W_i\}} (F\_Value(\{W_i\})),$$

where F_Value($\{W_i\}$) reflects how well our system outputs match up with the ground truth when using $\{W_i\}$ as transformation path weights. The F-measure [5] (the harmonic mean of precision and recall) is often used to evaluate matching quality; maximizing the F-measure on the training data seeks to achieve similarly good results on the evaluation data. There are two main steps in our algorithm: the first is partitioning name pairs into blocks according to their *TP* and similarities so that we can operate on them in batches; the second is a multi-way merge sort so that we only have to calculate F_Value *N(block)* times to detect the best F-measure in training, where *N(block)* is the number of blocks.



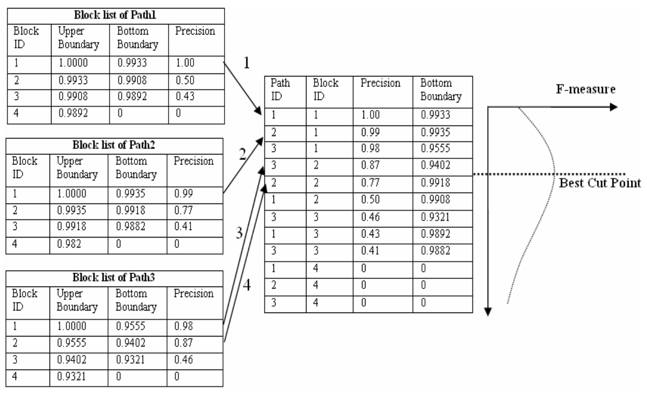
Pair ID	T/F	Similarity
288	1	0.9939
1422	1	0.9939
267	1	0.9939
4789	1	0.9933
324	0	0.9930
595	0	0.9930
522	1	0.9919
8436	0	0.9919
407	1	0.9912
5339	1	0.9908
551	0	0.9903
553	0	0.9903
911	0	0.9903
185	1	0.9903
259	0	0.9892
456	1	0.9892
781	1	0.9892
558	0	0.9877
8889	0	0.9866

Threshold Candidates in this Path		
Candidate	Number of Positive	Total Number
0.9939	3	3
0.9933	1	1
0.9930	0	2
0.9919	1	2
0.9912	1	1
0.9908	1	1
0.9903	1	4
0.9892	2	3

Block list in this Path			
Block ID	Upper Boundary	Bottom Boundary	Precision
1	1.0000	0.9933	1.00
2	0.9933	0.9908	0.50
3	0.9908	0.9892	0.43
4	0.9892	0	0

Figure 3. Partitioning name pairs with same TP into blocks.

Figure 3 illustrates the process of partitioning name pairs into blocks. First, we group all name pairs with the same transformation path TP (and hence the same similarity, computed as in Table 1) and rank the name pairs in descending similarity order. We then define *block precision* as the number of positive name pairs in a block divided by the total number of pairs in the block, and we set a boundary between blocks whenever adding a new set of name pairs (i.e., pairs with the same TP) would reduce the *block precision*.



Block list of Path1			
Block ID	Upper Boundary	Bottom Boundary	Precision
1	1.0000	0.9933	1.00
2	0.9933	0.9908	0.50
3	0.9908	0.9892	0.43
4	0.9892	0	0

Block list of Path2			
Block ID	Upper Boundary	Bottom Boundary	Precision
1	1.0000	0.9935	0.99
2	0.9935	0.9918	0.77
3	0.9918	0.9882	0.41
4	0.982	0	0

Block list of Path3			
Block ID	Upper Boundary	Bottom Boundary	Precision
1	1.0000	0.9555	0.98
2	0.9555	0.9402	0.87
3	0.9402	0.9321	0.46
4	0.9321	0	0

Path ID	Block ID	Precision	Bottom Boundary
1	1	1.00	0.9933
2	1	0.99	0.9935
3	1	0.98	0.9555
3	2	0.87	0.9402
2	2	0.77	0.9918
1	2	0.50	0.9908
3	3	0.46	0.9321
1	3	0.43	0.9892
3	3	0.41	0.9882
1	4	0	0
2	4	0	0
3	4	0	0

Figure 4. Multi-way merge sorting to calculate the F-measure.

Figure 4 shows the multi-way merge sorting process: there are three block lists for three name transformation paths; these blocks are merged into the result list in descending order of *block precision*; the F-measure (now calculated on pairs, not blocks) typically increases to a unimodal peak and then decreases as the size of result list grows. We search exhaustively to locate the maximum F-measure, and define the corresponding similarity value as the cut point, which we call TH_{best}. For each transformation path (TP_i), we label the corresponding similarity (as calculated for Table 1) B_i and we estimate a reasonable weight for that path as: $W_i = TH_{best} / B_i$. We call this method STP+WT.

3.2 Learning a Decision Rule

Given a similarity value, we must decide whether two name strings refer to the same person. We train a Support Vector Machine (SVM) [7] to learn a decision rule. Our feature set includes the number of initials and non-initials in the matching format, the number of each type of transformation in the transformation path, and the matching similarity. Table 2 shows a decision table that illustrates this process.

Table 2. Decision table for judging the name pairs.

		Attributes	P_1	P_2	P_3	P_i
Conditions	$TP_{A=>M}$	A	1	0	0	…
		O	0	1	1	…
		S	0	0	0	…
	$TP_{B=>M}$	A	0	0	0	…
		O	0	1	1	…
		S	0	0	0	…
	Matching Format	I	1	0	0	…
		N	2	2	2	…
	Similarity	Similarity	0.93	0.89	0.78	…
	Positive (+1)		√			…
	Negative (-1)			√	√	…

In our SVM, we use a radial basis function; the kernel and penalty parameters are tuned following the instructions for libsvm [3]. We label the method STP+SVM for convenience.

4. EXPERIMENTS

4.1 Datasets and Evaluation Metrics

We evaluate on datasets from three sources. The first is the CiteSeer dataset containing citations from four areas in machine learning, originally created by Giles et al. [10]. It has 2,892 references to 1,165 authors. The second is significantly larger: Arxiv (HEP) contains papers from high-energy physics used in KDD Cup 2003.[1] It has 58,515 references to 9,200 authors. The third is from Reuther; it was built upon the Digital Bibliography & Library Project (DBLP) [13]. There are three subsets defined for Reuther's dataset; we use the largest, sub03, which contains 58,399 references to 21,688 authors. Author names were hand-labeled for co-reference in all three datasets. Table 3 summarizes the dataset characteristics. In our experiments, we evaluate matching quality using the F-measure.

Table 3. Comparison of the three name datasets

	CiteSeer	arXiv	DBLP
Unique name strings	1,636	12,732	21,688
Coreferent name string pairs	798	4,923	2,020
Average characters per name	10.30	12.17	13.57
Average tokens per name	2.43	2.41	2.50

4.2 Comparing Unsupervised Techniques

In previous studies, good results have been reported for the Soft-TFIDF (STF), Jaro-Winkler (JW), recursive JW (LJW), Monge-Elkan (ME), and recursive ME (LME) string similarity metrics, so we adopt those as baselines. We implemented each using the Second String Project.[2] Training an SVM adds an additional source of variation, so for these initial experiments we report post-hoc optimal F-measure values based on ranking all candidates in decreasing order of similarity and then sweeping across all possible rank cutoffs to find an optimal (global) rank cutoff for each measure. Pairs above the rank cutoff are classified as matches. As Table 4 shows, for datasets with more unique name strings the task is noticeably harder. Ranging from the most difficult to least we have: DBLP > Arxiv > CiteSeer. STP achieves the highest post-hoc optimal F-measure for all three datasets. The post-hoc optimal F-measure for the largest dataset, DBLP, is still undesirably low (~0.45), although STP did achieve a 25% relative improvement over the next best similarity measure (STF). This motivates our focus on supervised techniques.

Table 4. Best F-measure on three datasets with six methods.

	STP	LJW	JW	LME	ME	STF
CiteSeer	**0.9544**	0.8567	0.4962	0.8581	0.7132	0.7982
Arxiv	**0.8684**	0.6352	0.3329	0.4252	0.6259	0.4333
DBLP	**0.4484**	0.3381	0.3297	0.2934	0.2399	0.3607

4.3 Comparing Unsupervised with Supervised

For our experiments with supervised methods, we use a variant of three-fold cross-validation in which we split each dataset into three subsets, choosing one for training and the other two for evaluation. This process is repeated three times, with each third being used as training data exactly once. The three resulting F-measures are averaged to produce the result plotted in Figure 5. CiteSeer is too small for cross-validation, so here we report

[1] http://www.cs.cornell.edu/projects/kddcup/

[2] http://secondstring.sourceforge.net/

results only for Arxiv and DBLP. For STP and STP+WT we report post-hoc optimal F-measures; for STP+SVM we report the F-measure for the decision rule learned by the SVM.

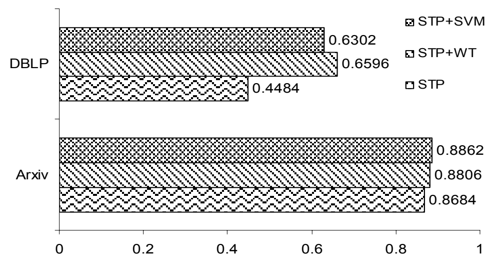


Figure 5. Best F-measure on two larger datasets using STP, STP+WT and STP+SVM methods.

In Figure 5, both supervised methods enhance the F-measure substantially (at least 45%) for DBLP, but far less improvement (about 1.5%) is evident for Arxiv. One possible explanation is that DBLP has a larger number of ambiguous name pairs than Arxiv, and thus perhaps has more room for improvement. DBLP is also considerably larger than Arxiv, which results in far more training data being available. The STP+SVM technique achieves results that are reassuringly close to the best post-hoc optimal F-measure with the STP+WT technique; the decision rule learned on the training data by our SVM is clearly quite good.

4.4 Implementation and Efficiency

Efficient name matching system is important since the number of name pairs can be very large. We build an index to retrieve similar tokens using the Flamingo package, which then finds approximately matching strings efficiently [2]. We also build a token-name inverted index to rapidly associate tokens with the names that contain that token. Figure 6 illustrates this process.

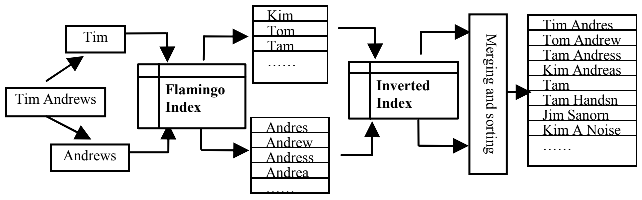


Figure 6. Pruning name pairs with two indices.

It takes only two minutes to build the two indices, and then three minutes to find all pairs that merit further processing. All name pairs that share one or more similar tokens are processed. For the DBLP dataset, there are 1,592,244 such pairs, which is just 0.7% of the total number of possible name pairs.

5. Conclusion and Future Work

We have introduced the concept of name transformation paths for person name matching. An unsupervised method (STP) and two supervised methods (STP+WT, STP+SVM) are proposed that leverage this idea to improve matching accuracy. Experiment results on three datasets show that our methods perform well. Of course, many names from outside the English speaking world are also often present in English texts and in the future we plan to extend our approach to model transformations that are appropriate for those names as well.

6. ACKNOWLEDGMENTS

The authors would like to thank the dataset providers. This work was completed during a visit by the first author to the UMIACS CLIP lab at the University of Maryland that was supported by National Natural Science Foundation of China Grant 70671007.

7. REFERENCES

[1] Arehart, M. D. and Miller, K. J. 2008. A Ground Truth Dataset for Matching Culturally Diverse Romanized Person Names. Language Resources and Evaluation Conference.

[2] Behm,A., Ji,S., Li, C and Lu,J. 2009. Space-constrained gram-based indexing for efficient approximate string search. International Conference on Data Engineering.

[3] Chang, C.C. and Lin C.J. 2001. LIBSVM: a library for support vector machines. Software available at http://csie.ntu.edu.tw/~cjlin/libsvm.

[4] Christen, P. 2006. A Comparison of Personal Name Matching: Techniques and Practical Issues. Technical report TR-CS-06-02, Australian National University, Canberra.

[5] Christen, P. and Goiser, K. 2006. Quality and complexity measures for data linkage and deduplication. In F. Guillet and H. Hamilton, editors, Quality Measures in Data Mining.

[6] Cohen, W. W., Ravikumar P., and Fienberg S. 2003. A Comparison of String Metrics for Matching Names and Records. KDD Workshop on Data Cleaning and Object Consolidation.

[7] Cristianini, N. and Shawe T. J. 2000. An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press.

[8] Dijkstra, E. W. 1959. A note on two problems in connection with graphs. Numerische Mathematik, 1 (1959), S. 269–271.

[9] Galvez, C. and Moya-Anegon, F. 2007. Approximate personal name-matching through finite-state graphs. JASIST, 58(13)1–17.

[10] Giles, C. L., Bollacker, K. and Lawrence, S. 1998. CiteSeer: An automatic citation indexing system. ACM Conference on Digital Libraries.

[11] Monge, A. and Elkan, C. 1997. An efficient domain-independent algorithm for detecting approximately duplicate database records. SIGMOD 1997 workshop on data mining and knowledge discovery.

[12] Piskorski, J., Wieloch, K., Pikula, M. and Sydow, M. 2008. Towards Person Name Matching for Inflective Languages. WWW 2008 Workshop NLP Challenges in the Information Explosion Era.

[13] Reuther, P. 2006. Personal name matching: New test collections and a social network based approach, Computer Science Technical Report 06-01, University of Trier.

[14] Winkler, W. E. 1999. The state of record linkage and current research problems. Statistics of Income Division, Internal Revenue Service Publication R99/04.