Approximate Personal Name-Matching Through Finite-State Graphs

Carmen Galvez and Félix Moya-Anegón
Department of information Science, University of Granada, Campus Cartuja, Colegio Máximo, 18071 Granada, Spain. E-mail: {cgalvez, felix}@urg.es

This article shows how finite-state methods can be employed in a new and different task: the conflation of personal name variants in standard forms. In bibliographic databases and citation index systems, variant forms create problems of inaccuracy that affect information retrieval, the quality of information from databases, and the citation statistics used for the evaluation of scientists' work. A number of approximate string matching techniques have been developed to validate variant forms, based on similarity and equivalence relations. We classify the personal name variants as nonvalid and valid forms. In establishing an equivalence relation between valid variants and the standard form of its equivalence class, we defend the application of finite-state transducers. The process of variant identification requires the elaboration of: (a) binary matrices and (b) finite-state graphs. This procedure was tested on samples of author names from bibliographic records, selected from the *Library and Information Science Abstracts* and *Science Citation Index Expanded* databases. The evaluation involved calculating the measures of precision and recall, based on completeness and accuracy. The results demonstrate the usefulness of this approach, although it should be complemented with methods based on similarity relations for the recognition of spelling variants and misspellings.

Introduction

A number of different interrelated areas are involved in the identification of *Personal Names* (PNs), including *Natural Language Processing*, *Information Extraction*, and *Information Retrieval* (IR). Personal names are included in the category of *proper names*, defined as expressions or phrases that designate persons without providing conceptual information. Their primary semantic function is to establish a relationship of reference between the phrase and the object. The identity certainty of this relationship is determined in many cases by context. Some problems surrounding *homonyms* (i.e., different individuals may have the same name) are due precisely to the ambiguity in that relationship of reference. The context could be clarified by determining the intellectual connection between the PNs, or by using subject indicators.

The referential ambiguity of PNs is a major problem in IR and citation statistics. The disambiguation of authors whose names coincide and overlap in the citation indexing systems calls for methods that distinguish the contexts in which authors or individuals appear, such as cross-document co-reference resolution algorithms that use the vector space model to resolve ambiguities (Bagga & Baldwin, 1998; Gooi & Allan, 2004; Han, Giles, Zha, Li, & Tsioutsiouliklis, 2004; Wacholder, Ravin, & Choi, 1997), co-occurrence analysis and clustering techniques (Han, Zha, & Giles, 2005; Mann & Yarowsky, 2003; Pedersen, Purandare, & Kulkarni, 2005), probabilistic similarity metrics (Torvik, Weeber, Swanson, & Smalheiser, 2005), or co-citation analysis and visualization mapping algorithms through a *Kohonen network* (X. Lin, White, & Buzydlowski, 2003; McCain, 1990). Underlying identity uncertainty problems, however, are beyond the scope of the present article.

PNs can be considered object tags that may appear in many different forms, known as *variants*. A PN variant can be described as a text occurrence that is conceptually well related with the correct form, or canonical form, of a name. The recognition of the variant of these sequences would revolve around one of three procedures specified by Thompson and Dozier (1999): *name-recognition*, *name-matching*, and *name-searching*.

Name-recognition is the process by which a string of characters is identified as a name. This is done both in database indexing processes and at the time of retrieval, after parsing the user query. It is widely used to extract names from texts, as described in the Message Understanding Conferences (MUC-4, 1992; MUC-6, 1995), as part of information extraction. In MUC-6, the recognition of the named entity is considered a key element of extraction systems; entities include names of persons, organizations, or places as well as expressions of time or monetary expressions. In MUC-7 (Chinchor, 1997), the named entity recognition

Received January 29, 2007; accepted January 30, 2007

implies identifying and categorizing three subareas which are the recognition of time expressions, numerical expressions, and entity names—persons, organizations, and places. There are many studies dedicated to the specification of the formation rules for names (Baluja, Mittal, & Sukthankar, 2000; Bikel, Miller, Schwartz, & Weischedel, 1997; Coates-Stephens, 1993; Gaizauskas, Wakao, Humphreys, Cunningham, & Wilks, 1995; Navarro, Baeza-Yates, & Arcoverde, 2003; Paik, Liddy, Yu, & McKenna, 1993; Patman & Thompson, 2003; Ravin & Wacholder, 1996; Wacholder, Ravin, & Byrd, 1994).

Name-matching is the process through which it is determined whether two strings of characters previously recognized as names actually designate the same person—name-matching does not focus on the case of various individuals who have identical name labels. Within this processing, two situations can arise: (a) The matching is exact, in which case there is no problem; or (b) the matching is not exact, making it necessary to determine the origin of these variants and apply approximate string matching.

Name-searching designates the process through which a name is used as part of a query to retrieve information associated with that sequence in a database. Here, likewise, two problems can appear: (a) The names are not identified as such in the database registers or in the syntax of the query, in which case Name-recognition techniques are needed; and (b) the names are recognized in the database records and in the query, but it is not certain whether the recognized names designate the same person, and so Name-matching techniques must be used.

The setting of application of approximate matching techniques is not substantially different from that of IR itself. These techniques would be integrated in automatic spelling correction systems and automatic text transformation systems to track down correspondences between the correct or standard dictionary forms and the items included in the database records. A further application would be when matching query terms with their occurrences in prestored dictionaries or indexes. Once again, information retrieval through names entails two points of inaccuracies. First, the documents might contain one or more terms of the query but with different spellings, impeding successful retrieval. Second, the documents may not contain the specific terms used in the query but have equivalent terms, which would again give rise to a lack of retrieval of relevant documents. The reason behind both types of inaccuracy is that both the classic Boolean model and the vector space model only retrieve relevant documents if they contain exact query terms (Belkin & Croft, 1987).

Background

Name-matching is just one area of research in IR and Web search engines (Hayes, 1994; Hermansen, 1985; Navarro et al., 2003; Pfeiffer, Poersch, & Fuhr, 1996; Spink, Jansen, & Pedersen, 2004). The identification of PN variants is a recurring problem for the retrieval of information from online catalogs and bibliographic databases (Borgman & Siegfried, 1992; Bouchard & Pouyez, 1980; Bourne, 1977; Rogers & Willett, 1991; Ruiz-Perez, Delgado López-Cózar, E., & Jiménez-Contreras, 2002; Siegfried & Bernstein, 1991; Strunk, 1991; Tagliacozzo, Kochen, & Rosenberg, 1970; Tao & Cole, 1991; Taylor, 1984; Weintraub, 1991). In general, the techniques for identifying variants are included under the common heading of *authority work* (Auld, 1982; Taylor, 1989; Tillett, 1989). An authority file serves to establish the correspondences among all the permitted forms of a sequence; that is, among any equivalent forms within a particular bibliographic file. This is particularly important when different bibliographic databases use different authority conventions, requiring the use of a *joint authority file* (French, Powell, & Schulman, 2000).

In retrieving scientific literature and citation indexes, a variety of formats may be used to cite the same author (Cronin & Snyder, 1997; Garfield, 1979, 1983a, 1983b; Giles, Bollacker, & Lawrence, 1998; Moed & Vriens, 1989; Rice, Borgman, Bednarski, & Hart, 1989; Sher, Garfield, & Elias, 1966). This inconsistency is present in the databases of the *Science Citation Index Expanded* (*SCI-E*), *Social Science Citation Index* (*SSCI*) and *Arts & Humanities Citation Index* (*A&HCI*), produced by the Institute for Scientific Information (ISI) and now owned by the Thomson Corporation, the *CiteSeer Project* at the NEC Research Institute, or the product from Elsevier called *Scopus*. Because citation statistics are habitually used to assess the quality of scientists' work, worldwide, many researchers and policy makers depend on the citation statistics to help determine funding. Citation inaccuracies cause serious problems in citation statistics. Therefore, experts in bibliometric analysis should put extra effort into validating the accuracy of data used in their studies.

Many studies focus on automatic spelling correction (Accomazzi, Eichhorn, Kurtz, Grant, & Murray, 2000; Angell, Freund, & Willett, 1983; Blair, 1960; Cucerzan & Brill, 2004; Damerau & Mays, 1989; Davidson, 1962; Gadd, 1988; Kukich, 1992; Landau & Vishkin, 1986; Petersen, 1986; Pollock & Zamora, 1984; Rogers & Willett, 1991; Takahashi, Itoh, Amano, & Yamashita, 1990; Ullmann, 1977; Zobel & Dart, 1995). A number of studies have highlighted the context for correcting such errors and eliminating ambiguity (Damerau, 1964; Hull, 1992; Mays, Damerau, & Mercer, 1991; Riseman & Ehrich, 1971; Schulz & Mihov, 2002).

Some of the aforementioned problems can be resolved with the general methodology of spellcheckers, which verify errors by cross-reference to dictionaries with the correct forms of any given type of string. In this development, Salton (1989) distinguished: (a) a process of exact matching between variants and the word forms stored in dictionaries, requiring the prestorage of a list of the correct forms along with the variants. This method is not practical because it would correct only a small fraction of the variants—those previously included in the dictionary; and (b) a process of approximate matching to find the dictionary entries that are as close as possible to the variants, with no need to establish an exact correspondence. What is important about this second strategy is the definition of the procedure for selecting the correct form, generally based on the minimization of distances or on the maximization of the similarity between the dictionary entries and the variants.

Correction techniques are known to work poorly, however, when they attempt to verify the spelling of a PN precisely because the storage process would be unbearably long and involved in view of the huge diversity of name structures, owing to historical conventions and cultural characteristics (Borgman & Siegfried, 1992). All these factors make it very difficult to justify the application of any single method for the automatic processing of personal name strings. The interest in name-finding IR systems has led researchers to develop a sort of software generally called *Name Recognition Technology*. Nevertheless, the decades of investigation have not produced important advances in name-matching, and to date, no method affords a definitive solution. Moreover, the spelling-correction procedures just described neither detect nor correct different citation formats, which constitute valid variants. To solve this matter, we would need to develop methods that identify and group all the valid variants. The approximate matching would be based on the establishment of similarity and equivalence relations, among the different variant forms as well as with the standard form of its equivalence class.

Research Question—The Problem of Personal Name Variants

The main purposes of this article are (a) to establish a categorization of the fundamental methods for the identification of PN variants within the framework of approximate string matching techniques, and (b) to develop a procedure for the automatic recognition and standardization of valid variants. With these aims in mind, we begin with a classification of PN variants as:

- *Nonvalid variants*, defined here as the variation present among nonlegitimate variant forms and correct forms. These include any misspellings of a phonetic or typographic nature, involving deletions, insertions, or substitutions of characters in the strings, the incorrect use of capital letters, nicknames, abbreviations, transliteration problems, and errors of accentuation in the names from certain languages.
- *Valid variants*, defined here as the variation that is produced among legitimate variant forms and canonical forms. Among these variants would be different syntactic formats and structural errors, such as the lack of some components of a full name, the absence or use of punctuation marks, the use of initials, variation in the syntax of names and the order of sequence of author citation, or permuted string order, due to particular conventions of the database.

According to this classification, the problem of recognizing PN variants would be integrated within the techniques of *approximate string matching* (Hall & Dowling, 1980), used to solve two types of problems: (a) *similarity problems*, calculating coefficients of similarity between nonvalid variants and correct forms; and (b) *equivalency problems*, requiring the computation of equivalence relations between the valid variants and standardized forms. This latter type of relationship exists between names that appear different, yet are interchangeable in specific contexts without leading to a change in meaning.

Under the approach of equivalence problems, our objective is to identify equivalent names that refer to the same person. This means recognizing ambiguous name labels, or different syntactic formats, of the same name. We consider these PN valid variants or syntactic formats as patterns or regular expressions that will be generated in accordance with a Local Grammar (LG), thus implying the application of parsing techniques. They will be identified and conflated by means of *finite-state methods*. With this objective, we shall construct *binary matrices* according to the methodology put forth by Gross (1975), and *master graphs*, or finite-state graphs, in charge of interpreting the elements of the matrices, after the procedure proposed by Roche (1993). The binary matrices and the master graphs will be applied for a new purpose: the automatic conflation of all valid variants of these sequences into equivalence classes. The computer tool used for finite-state graph analysis is a parser developed by Silberztein (1993, 2000).

Approach to Personal Name Variants: Similarity Versus Equivalence Relations

The inaccuracies produced by PN variants, in the framework of approximate string matching, could be solved through statistical similarity measures or with the establishment of equivalence relations between variant forms and standard forms. The main difference is in their underlying mathematical concepts. According to Hall and Dowling (1980), we would have: "Given s in s, find all t in T such that $s \sim t$," a similarity relation for r, s, and t in s has the properties: reflexive ($s \sim s$), symmetric ($s \sim t = t \sim s$), but not necessarily transitive ($r \sim s$ and $s \sim t \neq r \sim t$). For example, the relationship of similarity between variants of a bibliographic author name taken from the database *Library and Information Science Abstracts* (*LISA*) could be as follows:

- Reflexive: *Melvon Ankeny ~ Melvon Ankeny*
- Symmetric: *Melvon Ankeny ~ Melvon L Ankenny = Melvon L Ankenny ~ Melvon Ankeny*
- Nontransitive: *Melvon Ankely ~ Melvon Ankeny* & *Melvon Ankeny ~ Melvon L Ankenny ≠ Melvon Ankely ~ Melvon L Ankenny*

Nevertheless, departing from the same premises, that is, "given s in S, find all t in T such that $s \approx t$," an equivalence relation for r, s, and t in S holds the properties: reflexive ($s \approx s$), symmetric ($s \approx t = t \approx s$), and transitive ($r \approx s$ and $s \approx t = r \approx t$). Accordingly, the relationship of equivalence between the names would be established as follows.

- Reflexive: *Melvon Ankeny ≈ Melvon Ankeny*
- Symmetric: *Melvon Ankeny ≈ Ankeny, Melvon L = Ankeny, Melvon L ≈ Melvon Ankeny*

- Transitive: *Ankeny, ML* ≈ *Melvon Ankeny* & *Melvon Ankeny* ≈ *Ankeny, Melvon L* = *Ankeny, ML* ≈ *Ankeny, Melvon L*

If we conceive name-matching as a problem of similarity relations, we would have to calculate different similarity coefficients that would either minimize or maximize the distance between correct variants and nonvalid variants. On the other hand, if we approach name-matching as a problem of equivalence relations, we would have to establish different classes of equivalency between standardized variants and valid variants.

When the identification of variants is carried out through exact matching, or the retrieval is done through the exact match between the query terms and the terms stored in the dictionary or in the database indexes, there would be no problem. The difficulty arises when the identification of the variants is done by means of approximate methods. A general taxonomy of these techniques is shown in Figure 1. In later sections, we describe how the different systems resolve the equivalency of variants using the proposed approaches.

We put forth that string-matching problems will involve both similarity and equivalence relations, and can be considered *hybrid problems* (Hall & Dowling, 1980). Similarity may be perceived among valid forms, which would influence the choice of the standard form. Under this approach, based on *co-occurrence analysis* (Xu & Croft, 1998), all variants would be replaced by the standard form. To bring variants together, clustering algorithms can be applied. That is, we could establish a relation of similarity as well as a relation of equivalence between the set of variant instances that are grouped into equivalence classes.

Similarity Relations

Approximate name-matching methods based on similarity relations are applied for the recognition of the nonvalid variants. The *Damerau–Levenshtein metric*, also known as edit distance, is a measure of string similarity defined as the minimal number of operations needed to transform one string into another (Damerau, 1964). The distance between the names *s* and *t* would be the number of edit operations that convert *s* into *t*. Assuming that most misspellings are single-character errors, as has been shown by different studies (Damerau, 1989; Petersen, 1986; Pollok & Zamora, 1983), the edit operations would consist of the insertion, deletion, or substitution of a single character, or the transposition of two characters, taking into account the cost of each operation. The Damerau–Levenshtein similarity coefficient between two names *s* and *t* is calculated iteratively using the function $f(i, j)$, which represents the minimum number of errors between two strings. If edit operations are produced, each one would have a cost , giving rise to the so-called *Levenshtein distance* (Levenshtein, 1965), expressed as follows:

$$\begin{aligned} f(0,0) &= 0 \\ f(i,j) &= min[f(i-1,j)+1, \\ &\qquad f(i,j-1)+1, \\ &\qquad f(i-1,j-1)+d(s_i,t_j)] \end{aligned}$$

where

$$d(s_i, t_j) = 0, \text{ if } s_i = t_j$$
$$d(s_i, t_j) = 1, \text{ if } s_i \neq t_j$$

Applying dynamic programming techniques to obtain the solution with the optimal value, the recursive relations of the Levenshtein algorithm count the minimum number of edit operations needed to transform *Name*1 into *Name*2. For example, if *s* were *Ankeny* and *t* were *Ankenny*, then $DL(s, j) = 1$ because it would suffice to omit one single character to transform *s* into *t*. The equivalence of the two names can be represented in a graph by means of the shortest path, in which the distance between the two PNs would be situated at the lower right corner of the matrix of Levenshtein distances (see Figure 2). To calculate the global distance, with which we would obtain the closest equivalence, we divide the distance calculated by the length of the longest string using this equation:

$$Global\ distance = \frac{DL(s,j)}{max(l_s, l_j)}$$

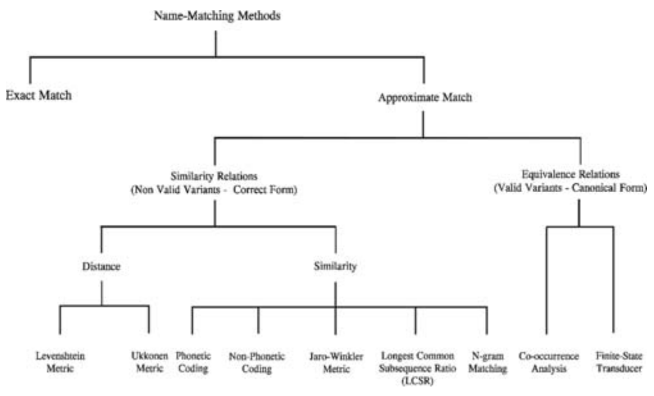
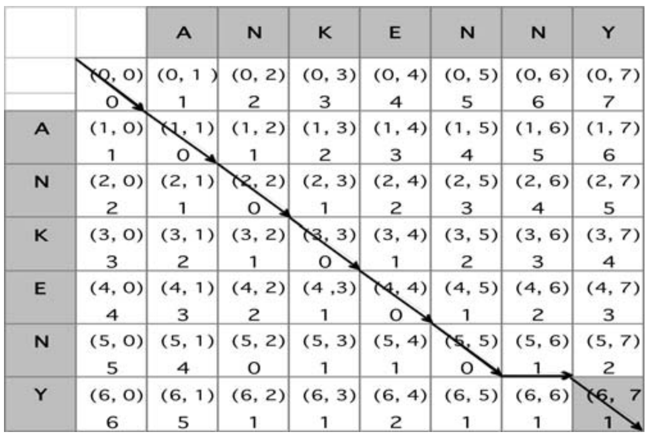


FIG. 1. Categorization of name-matching methods.



		A	N	K	E	N	N	Y
	(0, 0) 0	(0, 1) 1	(0, 2) 2	(0, 3) 3	(0, 4) 4	(0, 5) 5	(0, 6) 6	(0, 7) 7
A	(1, 0) 1	(1, 1) 0	(1, 2) 1	(1, 3) 2	(1, 4) 3	(1, 5) 4	(1, 6) 5	(1, 7) 6
N	(2, 0) 2	(2, 1) 1	(2, 2) 0	(2, 3) 1	(2, 4) 2	(2, 5) 3	(2, 6) 4	(2, 7) 5
K	(3, 0) 3	(3, 1) 2	(3, 2) 1	(3, 3) 0	(3, 4) 1	(3, 5) 2	(3, 6) 3	(3, 7) 4
E	(4, 0) 4	(4, 1) 3	(4, 2) 2	(4 ,3) 1	(4, 4) 0	(4, 5) 1	(4, 6) 2	(4, 7) 3
N	(5, 0) 5	(5, 1) 4	(5, 2) 0	(5, 3) 1	(5, 4) 1	(5, 5) 0	(5, 6) 1	(5, 7) 2
Y	(6, 0) 6	(6, 1) 5	(6, 2) 1	(6, 3) 1	(6, 4) 2	(6, 5) 1	(6, 6) 1	(6, 7 1

FIG. 2. Levenshtein distance between two variant forms of names.

Thus, we have:

$$Global\ distance = \frac{1}{max(6,7)} = 0.14$$

The distance between the two strings also could be calculated using the number of *n-grams*, as proposed by Ukkonen (1992). An n-gram of a name *s* is a substring of *s* with the length *n*. This unit is commonly used as a measure of similarity between strings. The Ukkonen proposal consists of calculating n-gram distance by means of the following equation:

$$n\text{-}gram\ distance\ (s, t) = \sum_{g \in G_s \cup G_t} |s[g] - t[g]|$$

where G_s and G_t are the sets of n-grams in *s* and *t*, *s*[*g*] is the number of occurrences of n-gram *g* in string *s*, and *t*[*g*] is the number of occurrences of n-gram *g* in string *t*.

If $n = 2$, the Ukkonen distance for the two PNs *s* = *Ankeny* and *t* = *Ankenny* would be calculated as indicated next:

$$G_{Ankeny} = \{An, nk, ke, en, ny\}$$
$$G_{Ankenny} = \{An, nk, ke, en, nn, ny\}$$
$$n\text{-}gram\ distance\ (Ankeny, Ankenny) = 1$$

Nonetheless, the simple recount of n-grams does not allow us to take into account string length as a factor (i.e., "Gold" has exactly the same n-grams in common with itself as with "Goldberg"), and moreover, it so happens that in the PN, the n-grams are not usually repeated. To solve this problem, Ukkonen (1992) suggested another measure of distance, q-grams.

$$q\text{-}gram\ distance\ (Name1, Name2) = |G_{Name1}| + |G_{Name2}| - 2|G_{Name1} \cap G_{Name2}|$$

If *Name*1 = *Gold* and *Name*2 = *Goldberg*, using the method *q-grams* with $q = 2$, the Ukkonen distance would be the following:

$$q\text{-}gram\ distance\ (Gold, Goldberg) = |3| + |7| - 2\,|3|$$
$$q\text{-}gram\ distance\ (Gold, Goldberg) = 4$$

Within the array of phonetic codings, we have the systems *Soundex* (Russell, 1918) and *Phonix* (Gadd, 1988, 1990). These methods are based on the assignment of the same key to those names that have a very similar pronunciation. They are used to simplify database searches when one knows only the sound of a name, but not its exact transcription. The Soundex algorithm, developed and patented by Russell (1918) reduces, particularly in the case of names in English, the surname to a code of four characters: The first character is an upper case letter, and the other three are digits. Knuth (1973) described this procedure by means of a function that consists of: (a) the conversion of characters to a phonetic code: 0 = (*a*, *e*, *h*, *i*, *o*, *u*, *w*, *y*), 1 = (*b*, *f*, *p*, *v*), 2 = (*c*, *g*, *j*, *k*, *q*, *s*, *x*, *z*), 3 = (*d*, *t*), 4 = (*l*), 5 = (*m*, *n*), 6 = (*r*), and (b) an algorithm that replaces all the characters except for the first with its corresponding phonetic code, eliminates all the appearances of the code 0 and any other adjacent repetition of codes, and returns only the first four characters of the resulting string.

Each name in the database is classified in one of these three ranks with respect to the query: identical, different but sharing the same code, and not related. A modification of the Soundex algorithm is the *Extended Soundex Algorithm*. In this extended system, the first letters are dealt with in the same manner as the rest of the letters; that is, the code is purely numerical, and this speeds up the database examination of names that sound alike. However, the flaw inherent in these two codifications is that they do not establish a ranking of the similar string; therefore, we have a variation of Soundex, the Phonix system, whose algorithm is more complex than that of its predecessor. Another code is *Metaphone* (Philips, 1990), which has an algorithm to phonetically codify words in English. The difference is that Metaphone only retains the consonants, and these are reduced to 16 (not digits, though there are exceptions such as 0 to represent the sound *TH*).

The Phonix code is based on the substitution of all the characters except the first with numerical values, with a slight variation—0 = (*a*, *e*, *h*, *i*, *o*, *u*, *w*, *y*), 1 = (*b*, *p*), 2 = (*c, g, j, k, q*), 3 = (*d*, *t*), 4 = (*l*), 5 = (*m*, *n*), 6 = (*r*), 7 = (*f*, *v*), 8 = (*s*, *x, z*)—and in the elimination of all the appearances of the value 0. The improvements introduced by Phonix are that it previously carries out some 163 transformations of groups of letters by standardizing the strings, and the final sounds are computed. Consequently, it is able to establish three ranks of similarity established by the words that coincide in their final sounds, in the prefixes of the final sounds, or that have different final sounds.

Taft (1970), in "Name sound techniques," reports on the code known as the *New York State Identification and Intelligence System* (NYSIIS), developed by the New York Division of Criminal Justice. The NYSIIS algorithm is much more complicated than the ones described earlier. It essentially consists of a code of up to six letters for each PN. For instance, the algorithm translates the first characters of a name, such as *MCA* ⇒ *MC* and *PH* ⇒ *F*; or the last characters of a PN, such as *EE*, *YE*, *IE* ⇒ *Y* and *NT*, *ND* ⇒ *D,* while all the vowel sounds are mapped to the letter *A*.

Noteworthy among the error-correction systems based on nonphonetic coding is *SPEEDCOP* (Spelling Error Detection-Correction Project) (Pollock, 1982; Pollock & Zamora, 1984), currently used in the databases of Chemical Abstracts Services. The nucleus of this system is an algorithm designed to correct misspellings that consist of a single error, and a dictionary where the correct forms are stored. Its method of nonphonetic coding captures the essence of the correct words as well as that of the misspellings, and a word is corrected when the compared codes collate very closely. The codes designated as *keys* can be of two types: "Skeleton keys" and "Omission keys" (Pollock & Zamora, 1984). The SPEEDCOP correction strategy is based on the creation of a key of similarity for each word of the dictionary and its classification in alphabetic order. The distance between the ordered keys constitutes the measure of string similarity.

Beyond the phonetic and nonphonetic codifying systems is the *Jaro–Winkler metric* (Jaro, 1989, 1995; Winkler, 1999), developed by the U.S. Census Bureau. This system is based on the number and order of characters common to two given words. Briefly, given the strings *Name*1 and *Name*2, let s' be the characters in *Name*1 that are the same in *Name*2, and let t' be the characters in *Name*2 that are the same in *Name*1. Then, a character a' in s' is "in common" with t', if the same character a' appears in the same order in *Name*2, and a character b' in t' is "in common" with s' if the character b' appears in the same order in *Name*1. A transposition for s' and t' is a position i such that $a'_i \neq b'_i$ (Hence, $T_{s',t'}$ is one half the number of transpositions for s' and t'.) The similarity measure would be found between values 0 and 1, with 1.0 = *perfect match*, 0.5 = *partial match*, and 0.0 = *completely different*. The Jaro–Winkler metric for *Name*1 and *Name*2 would be contained in the following equation:

$$Jaro{-}Winkler(Name1, Name2)$$
$$= \left(\frac{1}{3} \cdot \frac{|s'|}{|Name1|}\right) + \left(\frac{1}{3} \cdot \frac{|t'|}{|Name2|}\right) + \left(\frac{1}{3} \cdot \frac{|s'| - T_{s',t'}}{|s'|}\right)$$

where s' and t' = *Number of matching characters*, *Name*1 = *Length of Name*1, *Name*2 = *Length of Name*2, and $T_{s',t'}$ = *Number of transpositions*.

If we consider that *Name*1 = *Ankeny* and *Name*2 = *Ankenny*, the Jaro–Winkler metric would be:

$$Jaro(Ankeny, Ankenny)$$
$$= \left(\frac{1}{3} \cdot \frac{5}{6}\right) + \left(\frac{1}{3} \cdot \frac{5}{7}\right) + \left(\frac{1}{3} \cdot \frac{5 - 0.5}{5}\right)$$
$$Jaro(Ankeny, Ankenny) = 0.8$$

Another measure of the similarity of names, but this time based on approximate spellings, is the *Longest Common Subsequence Ratio* (LCSR), proposed by Melamed (1999). Two names would be related if they have a certain number of characters in common. The LCSR of the two names, *Name*1 = *Ankeny* and *Name*2 = *Ankeny*, is defined by the ratio of length (not necessarily continuous) of their *Longest Common Subsequence* (LCS) and the length of the longest name.

$$LCSR(Name1, Name2)$$
$$= \frac{length\ [LCS\ (Name1, Name2)]}{max\ [length\ (Name1), length\ (Name2)]}$$
$$LCSR(Ankeny, Ankenny) = \frac{6}{7} = 0.85$$

Finally, the measure of similarity based on n-grams uses a function of binary identity to compare orthographic characters by means of the *Dice coefficient* (Angell et al., 1983; Salton, 1989; Zamora, Pollock, & Zamora, 1981), or trigrams (Church, 1988). The similarity between two names, *Name*1 = *Ankeny* and *Name*2 = *Ankenny*, on the basis of the Dice coefficient, would be calculated using the following equation:

$$Dice(Name1, Name2)$$
$$= \frac{2|bigrams\ (Name1) \cap bigrams\ (Name2)|}{|bigrams\ (Name1)| + |bigrams\ (Name2)|}$$
$$Dice(Ankeny, Ankenny) = \frac{14}{15} = 0.93$$

In lieu of this coefficient, some other measure of the degree of association between two binary variables could be used, such as the Φ coefficient, t score, odds ratio, or log likelihood ratio.

Equivalence Relations

The approximate name-matching methods based on equivalence relations are applied for the recognition of the valid variants of a PN. An equivalence relation in a set S is a binary relationship between pairs of elements of S that feature the properties of reflexivity, symmetry, and transitivity. Thus, given an equivalence relation in S, the class of equivalence of an element x of the set S, $x \in S$, would consist of all those elements s of the set S, $s \in S$ that are related with the element x. To indicate the class of equivalence of x, the following annotation can be used:

$$cla[x] = \{s \in S; s \approx x\}$$

The relations of equivalence of strings are presented as "*Given s in S, find all t in T such that s ≈ t*." Consequently, the search for classes of equivalence between PNs could be defined as: "Find all the elements t in T which are in the same equivalence class as the search string s." The equivalence class would be characterized as a representative member of the class defined as the canonical form. Hence, the problem of the search for equivalent names could be reformulated, along the reasoning of Hall and Dowling (1980), as: "*Find all the elements t in T which are in the same canonical form as the search string s*."

We assume that the association of variant instances with canonical forms would be included within the general framework of the reduction of strings to standard forms. Therefore, it would be necessary to use term conflation methods; that is, methods for the standardization and unification of terms belonging to the same equivalence class. The techniques for term conflation most widely used involve stemming algorithms. We focus on the grouping of multiple variations of a PN that is to be mapped into a canonical form, or single class, by means of: (a) *equivalence class based on co-occurrence analysis* and (b) *equivalence class based on finite-state transducers* (FST).

Equivalence classes can be built using co-occurrence analysis, according to Croft and Xu (1995; Xu & Croft, 1998), who designed a corpus-filter to select word variants for stemming. Their hypothesis is that the word forms that should be conflated for a given corpus will co-occur in documents from that corpus. This proposition attempted to map multiple variations into a single class by means of the *Expected Mutual Information Measure* (Church & Hanks, 1990; van Rijsbergen, 1979). Arriving at equivalence class on the basis of statistical association involves checking word co-occurrence and similarity measures. After binary relations are established, the words are grouped in clusters of highly similar terms.

In a potential application of *Mutual Information* (MI) to the standardization of PNs, the joint probability of two PN components would compare with the probability that they appear independently. If two components c_1 and c_2 have the probabilities $P(c_1)$ and $P(c_2)$ of occurrence, then the MI similarity coefficient would be obtained by applying the following equation:

$$MI(c_1, c_2) = log_2 \frac{P(c_1, c_2)}{P(c_1)P(c_2)}$$

where $P(c_1, c_2)$ is the joint probability of two components (c_1, c_2) in the corpus, $P(c_1)$ is the probability of the independent occurrence of the component c_1 in the corpus, and $P(c_2)$ is the probability of the independent occurrence of the word c_2 in the corpus. Probabilities $P(c_1)$ and $P(c_2)$ are calculated by counting the frequency of occurrence of $f(c_1)$ and $f(c_2)$ in a corpus. The joint probability $P(c_1, c_2)$ is the count of the frequency with which c_1 is followed by c_2, within a parameter or set of components $f_c(c_1, c_2)$, normalized in corpus N:

$$MI(c_1, c_2) = log_2 \frac{Nf(c_1, c_2)}{f(c_1)f(c_2)}$$

Equivalence classes would comprise the PNs that share n-grams, by refining these classes with clustering techniques. Probabilistic modeling with n-grams would take into account the context where these PN components appear. The co-occurrence of n-grams for a PN component c_i would be the set of probabilities that the component is followed by another component, for each possible combination of c_i in the vocabulary of that language. The linking coefficient between n-grams also would be calculated by MI to obtain the association matrix of the conditioned probability of a component c_i being followed by other components.

In a unigram, bigram, or trigram bigram model, the probability for $c_1, c_2 \ldots c_n$ would be the product of $P(c_i)$, $P(c_i|c_{i-1})$, and $P(c_i|c_{i-2}\, c_{i-1})$. To calculate these probabilities, we would need the associated expected frequencies from the estimation parameters of $P(c_i) = f(c_i) / N$, $P(c_i|c_{i-1}) = f(c_{i-1}, c_i) / f(c_{i-1})$, and $P(c_i|c_{i-2}, c_{i-1}) = f(c_{i-2}, c_{i-1}, c_i) / f(c_{i-2}, c_{i-1})$. To construct the association matrix in the case of a bigram model, we would apply an adaptation of the MI similarity coefficient, with which the probability of the co-occurrence of component c_1 along with word c_{1-1} is obtained, as well as the independent probability of the occurrence of component c_1 and of word c_{1-1}:

$$MI(c_1, c_{1-1}) = log_2 \frac{P(c_1|c_{1-1})}{P(c_1)P(c_{1-1})}$$

Overall, a high score of $MI(c_1, c_2) > 0$, would indicate that there is a genuine association between two components, and so they are very likely to occur together. The lowest coefficient, $MI(c_1, c_2) < 0$, would show a complementary distribution. A low score of $MI(c_1, c_2) \approx 0$ would point to an irrelevant association. This result is represented as a coordinate on a vector, and then, on the basis of vector similarity measures, component clustering is effected.

The normalized similarity matrix would characterize the associations of PN components, from which they could be conflated into a single class, or the canonical form, by means of a clustering technique such as the hierarchical method of simple linking. There are a number of variants of clustering which can give different groupings depending on the agglomeration rule applied. There is therefore no single solution to this conflating process; rather, at any given stage in the fusing process, the most satisfactory PN for the grouping is obtained. Finally, the PN class is identified for each component cluster.

Nevertheless, the hypothetical application of statistical methods for PN conflation would have serious weaknesses. First, there would be a tendency to group PNs representing very different name structures because the number of frequencies is different than the similarity coefficient (Church & Hanks, 1990). Therefore, the joint probabilities are symmetric, $P(c_1, c_2) = P(c_2, c_1)$, and the similarity coefficient also is symmetric, yet the ratio of frequencies is asymmetric, $f(c_1, c_2) \neq f(c_2, c_1)$. For instance, $f(Melvon, Ankeny) \neq f(Ankeny, Melvon)$. A second drawback would be manifest with the use of initials, the use of punctuation marks, the permuted component order, abbreviated PNs, and, more importantly, the lack or nonappearance in many cases of some components of a full name. Consequently, the process of normalization using only the co-ccurrence of components can generate inaccuracies and statistically significant names that would be *compositionally* incorrect. All these factors suggest ineffective conflation performance if applied to PNs.

We defend the application of FST for the recognition and grouping of the different variants into an equivalence class that would be configured as the standard form. One application of FST is to establish a *Regular Relation* between input strings and output strings. Briefly, an FST, sometimes referred to as a *Mealy Machine* (Hopcroft & Ullman, 1979), is a system comprising a finite set of states and a function of transition, which defines the changes of the state. The transition function is tagged with a pair of symbols, which proceed respectively from an input alphabet and an output alphabet. This mechanism can be represented in the form of *finite-state graphs* or transition diagrams, or else as a *matrix* or *transition table*. The FST accept input strings and associate them with output strings.

Formally, an FST is referred to as a quintuple (Roche & Schabes, 1995) expressed as shown next:

$$FST = (\Sigma, Q, i, F, E)$$

where

Σ = *input and output alphabet*
Q = *number of states*
i = *initial state,* $i \in Q$
F = *final state,* $F \subseteq Q$
E = *number of transition relations,* $E \subseteq Q \times \Sigma \cup \{e\} \times \Sigma^* \times \Sigma$

Finite-state mechanisms are used to accept and unify variant instances and transform them into standard forms. The linguistic toolbox Intex 4.33 (Silberztein, 1993, 2000) will allow us to create finite-state graphs to later convert them into FSTs by means of a compiler. The finite-state graph (Figure 3) would represent the possible structures making up the variants of a name, and produce as output the structure selected as the standardized form of that PN. In this case, we have proposed the format of author citation used by the ISI, although we could have opted for any other particular format.

Once the finite-state graph is compiled in an FST, the application itself allows its transformation into a table or transition matrix (Table 1) where the following components are specified:

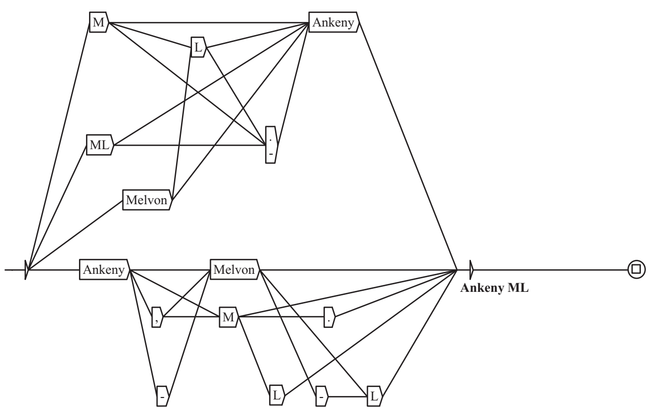


FIG. 3. Representation of a finite-state graph that conflates the name variants into a standardized format.

Number of alphabet symbols, or vocabulary, $\Sigma = 10$, that is, Σ = (*"L"*, *"M"*, *"ML"*, "*Melvon*", "*Ankeny*", ". ", "-", ", ", "<*E*>/*Ankeny ML*", "<*E*>"), where the symbol <*E*> represents the empty string.

Number of FST states, $Q = 17$

Initial state, $i = 0$.

Final state, $F = 1$.

Number of transitions between states, $E = 34$, where each transition is defined by a 3-tuple: *current state*, *symbol*, *outgoing state*.

TABLE 1. Result of the transformation of the finite-state graph into a transition table.

```
// Characteristics of the FST
#define NUMBER_OF_STATES 17 // states are numbered from 1 to NUMBER_OF_STATES
#define NUMBER_OF_SYMBOLS 10 // symbols are numbered from 0 to NUMBER_OF_SYMBOLS-1

#define NUMBER_OF_TRANSITIONS 34

// FST Alphabet/Vocabulary
static const char *symbols[NUMBER_OF_SYMBOLS]={
"L","M","ML","Melvon","Ankeny",".","-",",","<E>/Ankeny ML","<E>"};

// FST terminal states: 0=non terminal; 1=terminal
static const char terminal state[NUMBER_OF_STATES+1]={
'\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\0','\1','\0','\0'};

// FST transitions: each transition is a 3-tuple (current state, symbol, outgoing state)
static const int transitions[NUMBER_OF_TRANSITIONS*3]={
1,1,2,1,2,3,1,3,4,1,4,5,
2,0,6,2,4,7,2,5,8,2,6,8,
3,4,7,3,5,8,3,6,8,
4,4,7,
5,1,9,5,3,10,5,6,11,5,7,12,
6,4,7,6,5,8,6,6,8,
7,8,15,
8,4,7,
9,0,13,9,5,14,9,8,15,
10,0,16,10,6,17,10,8,15,
11,3,10,
12,1,9,12,3,10,
13,8,15,
14,8,15,
16,8,15,
17,0,16
};
```

TABLE 2. Language generated by the finite-state graph.

Ankeny M => Ankeny ML	Ankeny Melvon L => Ankeny ML
Ankeny Melvon => Ankeny ML	Ankeny M L => Ankeny ML
Melvon Ankeny => Ankeny ML	Ankeny M. => Ankeny ML
ML Ankeny => Ankeny ML	Ankeny Melvon - L => Ankeny ML
M Ankeny => Ankeny ML	Ankeny - Melvon L => Ankeny ML
M - Ankeny => Ankeny ML	Ankeny, Melvon L => Ankeny ML
M. Ankeny => Ankeny ML	Ankeny, M L => Ankeny ML
M L Ankeny => Ankeny ML	Ankeny, M. => Ankeny ML
ML - Ankeny => Ankeny ML	M L - Ankeny => Ankeny ML
ML. Ankeny => Ankeny ML	M L. Ankeny => Ankeny ML
Ankeny, M => Ankeny ML	Ankeny, Melvon - L => Ankeny ML
Ankeny, Melvon => Ankeny ML	Melvon L Ankeny => Ankeny ML
Ankeny - Melvon => Ankeny ML	Melvon L- Ankeny => Ankeny ML
Ankeny - Melvon - L => Ankeny ML	Melvon L. Ankeny => Ankeny ML

FST, within the framework of the approximate string matching techniques, is a technology applied to identify and group all possible valid variants of a PN. These variants would belong to the same equivalence class—characterized as a representative member of the class. The finite-state graph obtained generates and recognizes 28 variant formats of the PN "Melvon L. Ankeny" and conflates then into the canonical form "Ankeny ML" (Table 2).

The finite-state graphs could be used in IR systems to index together all the variants, and to locate all these variants in the texts of databases. This procedure therefore promises greater control of database index entries; however, its use within the IR systems environment would be inefficient, as it would require the hand-drawn representation of each equivalence class and the thousands of variants that these sorts of sequences might have. The next sections will describe a semiautomatic procedure that would allow us to recognize and unify the valid variants into standard forms.

Methodology

The automatic method for the identification and the conflation of the valid variants of a PN into equivalence classes that we propose entails three phases. The first step is the exhaustive and systematic analysis of the components of this type of noun phrase. These sequences will be generated by means of formation rules in a *task-oriented grammar*, according to the classification of Pereira (1997), defined as a set of units and rules designed for a specific application; however, under the approach we adopt, the formation rules are to be constructed in an indirect manner, through the automatic conversion of the elements stored previously in *binary matrices* (Gross, 1975, 1997). At the same time, this calls for the construction of a *master graph* (Roche, 1993, 1999), or finite-state graph, that will be in charge of interpreting the elements and the properties described in the matrices, giving rise to the indirect generation of a *grammar*.

In principle, the binary matrices, or lexicon-grammar matrices, are conceived as a mode of linguistic representation that does not separate grammar and lexicon to describe simple, elementary linguistic sequences such as PNs. This methodology was heavily influenced by the *transformational grammar* of Harris (1951), articulated essentially around systematic relationships between syntactic structures and lexical units. The actual attempt to apply binary matrices to the recognition of certain text expressions, however, began with the construction of master graphs (Roche, 1993, 1999; Senellart, 1998), giving rise to the first computational formalism able to generate automata and transducers from the data stored in a binary matrix. This notion, in turn, was inspired by the *Recursive Transitions Network* (Woods, 1970).

Formation Rules for PNs

An LG consists of rigorous and explicit specifications of particular structures, and is formally defined as a tuple of four elements (N, T, S, P) (Hopcroft & Ullman, 1979), where N would be an alphabet of nonterminal symbols, T would be an alphabet of terminal symbols, S would be an initial symbol belonging to the nonterminal alphabet, and P would be the set of formation rules built manually for the generation of the possible structures. As we consider the PNs to be phrases, the LGs would generate the structures of these sequences.

Before developing the LG, we must define the linguistic phenomena we wish to represent. The habitual procedure consists of devising a list of constructions or *specifications*. Another technique consists of departing from an assembly of model constructions, which may be real examples that appear in the sample, that serve to guide the development of the LG. We used this second procedure, taking a sample of PNs from the LISA and SCI-E databases. The collection need not be very large, as there are only so many legitimate structural forms of names. Thus, after a certain point, the larger the sample, the lesser the variations. Thus, the LG for PNs would be defined through the following components.

An alphabet of nonterminal symbols, *N*, or *variables*, made up of:

$$\left\{\begin{array}{l} INA1, NA1, INA2, NA2, Part, \\ INA1\_INA2\_INPart, INA1\_INA2\_INSUR1, \\ INSUR1, SUR1, SUR2, SUR3 \end{array}\right\}$$

whose annotation would be:

*INA*1	Initial of Name1
*NA*2	Name1
*INA*2	Initial of Name2
*NA*2	Name2
Part	Particle
*INA*1_*INA*2_*INPart*	Initial of Name1_Initial of Name2_Initial of Particle
*INA*1_*INA*2_*INPart*_*INSUR1*	Initial of Name1_Initial of Name2_Initial Particle_Initial of Surname1
*INSUR*1	Initial of Surname1
*SUR*1	Surname1
*SUR*2	Surname2
*SUR*3	Surname3
PUNC	Punctuation marks

An alphabet of terminal symbols, *T*, with a *constant* value, that are grouped inside a dictionary or lexicon, made up of:

$$\left\{\begin{array}{l} \text{“-”, “, ”, “.”}, A, B, C, D, E, F, G, H, I, J, K, L, M, \\ N, M, N, O, P, K, R, S, T, V, W, Abate, Abbott \\ Abrahamsson, Adams, \ldots \end{array}\right\}$$

An initial symbol *S*, or initial variable of the LG, defined as *PN* (Personal Name) Name Formation rules, *P*, for the generation of the corresponding structures, some of which might be the following:

$PN \rightarrow NA1\ SUR1$
$PN \rightarrow INA1\ INA2\ SUR1$
$PN \rightarrow NA1\ NA2\ SUR1\ SUR2$
$PN \rightarrow INA1\ SUR1\ PUNC(\text{-})\ SUR2$
$PN \rightarrow NA1\ NA2\ Part\ SUR1$
$PN \rightarrow INA1\_INA2\_INPart\ SUR1$
$PN \rightarrow SUR1\ PUNC\ (,)\ NA1$
$PN \rightarrow NA1\ SUR1\ PUNC\ (\text{-})\ SUR2$
$PN \rightarrow NA1\ SUR1\ SUR2\ PUNC\ (\text{-})\ SUR3$

There are different types of grammars, depending on the formation rules. Chomsky (1957) classified the grammars in four major groups—known as the *hierarchy of Chomsky*—for which each group includes the next one ($G3 \subseteq G2 \subseteq G1 \subseteq G0$). The grammars of one type generate the strings corresponding to one type of language type, $L(G)$. The generative power of each grammar is determined by the type of strings of a language that it is capable of generating, *Type* 3 ⊂ *Type* 2 ⊂ *Type* 1 ⊂ *Type* 0. Thus, Regular Grammars, *G*3, generate *Type* 3 languages; that is, they are the ones with the most limited generative power. Even so, this type of grammar is expressive enough to generate the possible structures of PNs. Table 3 shows an extract of a simple grammar constructed for a name.

TABLE 3. Extract of a Local Grammar for a Personal Name.

Formation rules	Nonterminal symbols (variables)	Terminal symbols (constants)
NA1 SUR1	PN	Melvon
NA1 INA2 SUR1	NA1	M
SUR1 NA1	NA2	L
SUR1 INA1	INA1	Ankeny
SUR1 INA1 INA2	INA2	
SUR1 NA1-INA2	SUR1	

Given that the grammars provide the rules used in the generation of the language strings, a connection is established between the classes of languages generated by certain types of grammar and the classes of languages recognizable by certain machines. Regular Languages can be characterized by *Regular Grammars* and are recognized through Finite-State Automata (FSA)—an equivalence existing between the Regular Grammar and the FSA. Basically, an automata or machine is an abstract mechanism that carries out a series of operations on entry strings and determines whether a given string belongs to one language or another. In this way, the automata are classified according to the type of language that they recognize. An FSA is made up of states and tagged arcs. Each arc represents a transition between two states, there being two main kinds of states: initial states and final states. In the FSA, the states are represented by means of circles or boxes, and the arcs by means of arrows indicating the direction of the transition.

We assume that the possible structures of the PN belong to Regular Languages, *Type* 3, generated by an LG or Regular Grammar, *G*3, and accepted by finite-state machines. Under this understanding, the grammars that characterize this type of noun phrase will be built indirectly, by means of a semi-automatic procedure that can convert binary matrices into LGs, according to the methodology that we will proceed to describe.

Binary Matrix for PNs

The nonterminal and the terminal components of the LG are represented together in a binary matrix, using a spreadsheet application, according to the methodology conceived by Gross (1975, 1997), which establishes the behavior of each one of the elements that integrate the PN structures. The format of the matrix has the following specifications:

The columns contain *variables*; that is, the parts or PN zone (e.g., *INA*1, *NA*1, *INA*2) and the names of the properties (e.g., *Part*).

The intersection of a PN zone (*column*) and a lexical entry (*row*) contains a sequence of *constants* (i.e., “M,” “L,” “Melvon,” “Ankeny”) or the symbol $<E>$, which is used to represent the empty string. The rows corresponding to the lexical entries can never be empty, and the data must be homogeneous.

The intersection of a property (*column*) and the property-coding (*row*) contains the symbol (+) if the lexical entry admits the property or the symbol (−) if the lexical entry is not admitted. Likewise, the rows corresponding to the properties cannot be empty, and the data must be homogeneous.

The binary matrices constitute a uniform organization for the representation of the PN components, which proves very efficient for the later computational processing. For the description of the structures of names, we developed a 464 × 11 matrix for LISA and a 210 × 11 matrix for SCI-E. Table 4 shows an extract of one of the specialized matrices. Nonetheless, these tables are simply a representational resource, not an automatic mechanism to be applied to the texts with the objective of recognizing certain sequences.

Master Graph for PNs

To use the binary matrix, a master graph must be built to for interpreting the information codified therein. The automatic procedure for transforming the matrices into finite-state mechanisms, which can indeed be used for texts, was first described by Roche (1993), though for a different purpose than the one we suggest. The procedure consists mainly of the construction of a master graph, or finite-state graph, which contains references to the cells of the binary matrices.

The master graph is related with the contents of the matrix through referential variables *@A, @B, @C, @D, . . .* where the variable *@A* refers to the contents of the first column, and variable *@B* refers to the contents of the second column. In turn, the master graph can be represented in

TABLE 4. Excerpt of a binary matrix for Personal Names.

A	B	C	D	E	F	G	H	I	J	K
INA1	NA1	INA2	NA2	Part	INA1_ INA2_ INPart	INA1_INA2_ INPart_ INSUR1	INSUR1	SUR1	SUR2	SUR3
M	Marie	A	<E>	–	MA	MAA	A	Abate	<E>	<E>
J	John	P	<E>	–	JP	JPA	A	Abbott	<E>	<E>
B	Bernadine	E	<E>	–	BE	BEA	A	Abbott	Hoduski	<E>
S	Sixten	<E>	<E>	–	<E>	SA	A	Abrahamsson	<E>	<E>
A	Audrey	M	<E>	–	AM	AMA	A	Adams	<E>	<E>
D	<E>	M	<E>	–	DM	DMA	A	Adams	<E>	<E>
M	Michael	Q	<E>	–	MQ	MQA	A	Adams	<E>	<E>
P	Peter	M	<E>	–	PM	PMA	A	Adams	<E>	<E>
G	George	W	<E>	–	GW	GWA	A	Adamson	<E>	<E>
M	<E>	R	<E>	–	MR	MRA	A	Aderibigbe	<E>	<E>
M	Mary	L	M	Mary	L	Lynn	–	ML	MLA	A	Ainsworth	<E>	<E>
M	Michael	J	<E>	–	MJ	MJA	A	Aldrich	<E>	<E>
B	Bettie	<E>	<E>	–	<E>	BA	A	Alexander	Steiger	<E>
F	Frank	H	<E>	–	FH	FHA	A	Allen	<E>	<E>
M	Melvon	L	<E>	–	ML	MLA	A	Ankeny	<E>	<E>
P	Pauline	<E>	<E>	–	PM	PMA	A	Atherton	Cochrane	<E>
S	Steven	D	<E>	–	SD	SDA	A	Atkinson	<E>	<E>
E	Ethel	<E>	<E>	–	<E>	EA	A	Auster	<E>	<E>
J	Joyce	EB	<E>	–	JEB	JEBB	B	Backus	<E>	<E>
S	Shelley	A	<E>	–	SA	SAP	B	Bader	<E>	<E>
E	Edward	W	<E>	–	EW	EWB	B	Badger	<E>	<E>
M	Mitchell	M	<E>	–	MM	MMB	B	Badler	<E>	<E>
R	Ricardo	<E>	<E>	–	<E>	RB	B	Baeza	Yates	<E>
S	Sharon	L	<E>	–	SL	SLB	B	Baker	<E>	<E>
C	Carol	M	<E>	–	CM	CMB	B	Bamford	<E>	<E>
A	Andrea	S	<E>	–	AS	ASB	B	Banchik	<E>	<E>
G	Gordon	S	<E>	–	GS	GSB	B	Banholzer	<E>	<E>
D	Derek	H	<E>	–	DH	DHB	B	Barlow	<E>	<E>
J	John	M	<E>	–	JM	JMB	B	Barnard	<E>	<E>
J	Joyce	M	<E>	–	JM	JMB	B	Barrie	<E>	<E>
J	Jean	P	Pierre	–	JP	JPB	B	Barthelemy	<E>	<E>
K	Kenneth	H	<E>	–	KH	KHB	B	Baser	<E>	<E>
R	Rashid	L	<E>	–	RL	RLB	B	Bashshur	<E>	<E>
M	Marcia	J	<E>	–	MJ	MJB	B	Bates	<E>	<E>
D	David	<E>	<E>	–	<E>	DB	B	Bawden	<E>	<E>
M	Mark	P	<E>	–	MP	MPB	B	Bayer	<E>	<E>
P	Pierre	M	Marie	–	PM	PMB	B	Belbenoit	Avich	<E>
N	Nicholas	J	<E>	–	NJ	NJB	B	Belkin	<E>	<E>

the form of an *Enhanced Finite-State Transducer* (EFST), defined as an FST that contains internal variables used during the parsing to order or classify the parts of the sequences recognized. The internal variables of the EFST are introduced as tagged parentheses around the corresponding sequences and are identified by the symbol (*$*). The use of EFST variables allows us to order and realize permutations or insertions in the recognized sequences. Basically, the components of a PN are distributed into the categories: *first*, *middle*, and *last name* (Ravin & Wacholder, 1996). Thus, the variables that we propose for the ordering of the PN sequences would be the following: *$F* (*First Name*), *$M* (*Middle Name*), and *$L* (*Last Name*).

The semiautomatic process that converts binary matrices (with nonterminal and terminal symbols) into grammars is based on the fact that for each row of the binary matrix, the corresponding LG is generated by means of a path in the master graph. First, if the variable refers to a lexical entry, the variable is replaced by the lexical element. Second, if the variable refers to the code (+), the path is maintained whereas if it refers to the code (−), the path is abandoned or rejected. The punctuation marks, as an additional LG element, are introduced directly into the master graph.

In Figure 4, we show a finite-state graph in charge of recognizing certain sequences of a PN, and relate them with those elements of the matrix that are to form part of the standardized string. The standard form that we have selected as representative of the equivalence class is defined by the format for author citation used by the ISI; that is, *SUR*1 *INA*1 *INA*2.

With this procedure, not only do we achieve the semiautomatic construction of the grammar that will generate combinations of the thousands of possible variant instances stored in the matrices but we also automate the construction of the FST in charge of recognizing them and conflating them into canonical forms. The extensive finite-state graph elaborated for the automatic construction of this LG is presented in Figure 5.

With this procedure, we manage to build a grammar capable of describing some 1,860 rules, corresponding to the number of possible structures of PNs. Nevertheless, the application of this approach has a limitation: The finite-state graph cannot be used to generate name structures but only to recognize them. For instance, the rule *PN* → *SUR*1 *INA*1 *INA*2 that would describe the PN "Ankeny ML" according to

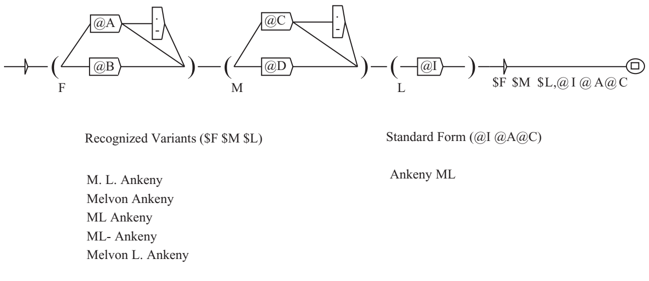


FIG. 4. Excerpt of a finite-state graph for Personal Names.

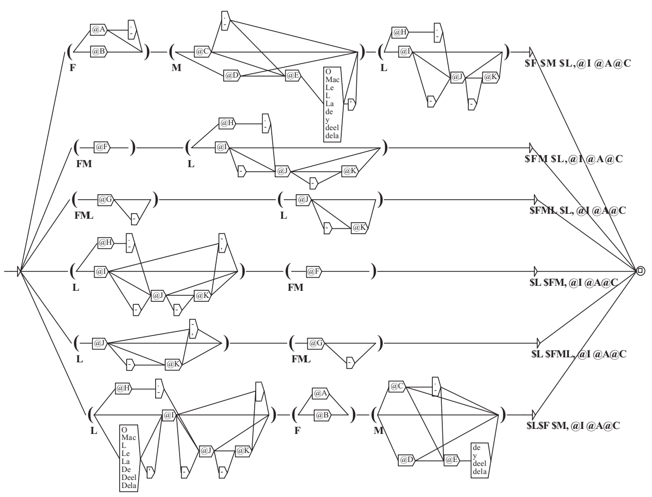


FIG. 5. Master graph for Personal Names.

the data of the matrix cannot be generated for this formalism, and therefore the finite-state graph would be restricted to identify the expression produced for this rule. On the other hand, because our aim is to recognize and conflate the structures into canonical forms, the master graph is configured as an FST graph, whose outputs associate the recognized sequences with those elements of the matrix considered to be the components of the standard form.

Evaluation Procedure

In this section, we evaluate the appropriateness of binary matrices and of the master graph for the identification of the PN variants. This procedure was tested on two samples of author names from bibliographic records, selected from a test list of 580 PNs, after removing the duplicates, from 1,382 records randomly selected from a search in LISA 1969–2004/2008; and a test list of 234 PNs, after removing the duplicates, extracted from 420 records selected from a search in SCI-E. All these records were retrieved through the Web of Science, with at least one author affiliation belonging to the research address "University of Granada, Department of Comp Science & Artificial Intelligence" in the field *Addresses*. The set of references obtained was imported to a bibliographic management system, ProCite database (Version 5.0), to automatically generate a list of variants that could be quantified in the evaluation. Before attempting analysis, the list was put through a series of transformations so that it could be processed in the text-file format and segmented it into sentences.

The next step was to apply the master graph to the occurrences of the selected variants. The fact that we have limited our consideration to a general sample found in LISA and to a specific sample inside SCI-E does not bias the results, as these records show great variability of name structures. The evaluation involved calculating the measures of precision and recall, based on completeness and accuracy, of effects on conflation performance but not actual retrieval. *Precision* could be redefined as the ratio of valid variants conflated from among the total variants identified by the finite-state graph. *Recall* would indicate the proportion of terms that are conflated with respect to a set of sequences of evaluation. We shall redefine it as correct variants conflated over total possible variants susceptible of normalization. One measure of performance that takes into account both recall and precision, the *F measure* (van Rijsbergen, 1979), stands as the harmonic mean of recall and precision (as compared to the arithmetic mean), which exhibits the desirable property of being highest when both recall and precision are high. Calculation entails the following equations:

$$Precision\ (P) = \frac{Number\ of\ Correct\ Variants\ Normalized}{Total\ Number\ of\ Variants\ Identified}$$

$$Recall\ (R) = \frac{Number\ of\ Correct\ Variants\ Normalized}{Total\ Number\ of\ Possible\ Variants}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Additionally, to evaluate the compression factor of fractional reductions in index size by means of conflation methods, we apply an adaptation of the *Index Compression Factor* (ICF) (Frakes & Fox, 2003), calculated using the equation:

$$ICF = \frac{(Total\ Number\ of\ Possible\ Variants - Number\ of\ Variants\ Conflated)}{Total\ Number\ of\ Possible\ Variants}$$

An extract of the data obtained is given in Table 5. The items in bold contain some type of misspelling or abbreviation. These sequences are included within the nonvalid variants, and therefore are not normalized by the method we have described.

Results and Discussion

We shall now expound and analyze the results of applying the master graph to the lists of author indexes. As shown in Table 6, the variants are identified with a precision of 99% in LISA and 98% in SCI-E. Results in both cases were very good, with *F* scores exceeding 98 and 95%, respectively. This very high precision index can be explained by the fact that the application was geared to a specific task, and all the possible PN structures were previously and exhaustively specified. That is, the information to be identified by the master graph was predetermined in the binary matrix. A further explanation of computational efficacy could be invoked: PN formation rules are straightforward, and for this reason, the finite-state mechanisms recognize the corresponding structures without any great difficulty.

The problems of *overanalysis* or structural ambiguity (when the construction can be analyzed in a number of ways, all of them correct), present in most systems that recognize patterns in practical situations, do not occur here because the master graph is closely guided by the data stored in the binary matrices. Even so, erroneous analysis occurred in 1% of the cases in LISA and in 2% in SCI-E because the master graph processes certain nonvalid variants as if they were valid (e.g., for a nonvalid variant such as “Aldrich Mike,” only the component “Aldrich” is recognized, giving rise to this margin of error in precision).

In the recall phase of the experiment, the model recognized 98% of the variants in LISA, with *F* scores of 98% as well. This very high percentage is due to the master graph’s capacity for generating and analyzing a great number of PN variants; however, if note that it was designed to identify some 1,860 structural variants of a single PN, recall should have been complete. The *underanalysis*, or lack of recall produced when the grammatical formalisms can detect only combinations specified in the matrices, stems from spelling errors. In LISA, the system did not manage to recognize 2% of the PN variants; the sequences with misspelling (e.g., “Marchioni Gary” for “Marchionini Gary”) could not be analyzed by the master graph.

In SCI-E, the variants are identified with a recall of 92%, meaning the master graph was not able to recognize 8% of PNs. Results in this case are relatively good, with *F* scores under 95%. The inaccuracies are caused, aside from misspelling, by some compound Spanish surnames that may appear with a variety of errors, such as the first surname run in with the second surname (e.g., “Gomezskarmeta AF” for “Gomez-Skarmeta AF”), the particle or articles of the surnames mistakenly united to the first or second surname (e.g., “Delablanca NP” for “De La Blanca NP”), and the abridgment of some surnames (e.g., “Fdez-Valdivia J” for “Fernandez-Valdivia J”). As all these sequences with errors and inexactitudes have no correspondence to the terminal symbols stored in the matrices, such problems cannot be avoided under this procedure.

Moreover, the application of the master graph achieves an ICF of 23% of the variants in LISA, meaning that if the total possible variants of the sample were 580, with this procedure we would achieve a reduction to 447. In SCI-E, the ICF arrives at 12%, meaning that if the total possible variants of the sample were 234, with this method they would be reduced to 205. The fact that the procedure we present manages to reduce the possible variants to this extent can be considered a satisfactory result.

Aside from the problems originated by the errors, an additional weakness of this approach is that we will not know exactly what constructions might appear in a collection, and yet we must mention a priori the type of PN structures to be processed before beginning to develop the model. The options are therefore either to devise a list of the constructions that we propose us to process or to limit the task to real examples taken from the domain of the application. The size of the collection does not necessarily have to be very large because the proportion of name formats diminishes as the sample increases. In any case, the master graph should be capable of processing at least those structures that appear in the specifications or in the sample.

TABLE 5. Excerpt of data obtained from the application of the master graph in a selection of author names found in LISA.

Possible Variants (LISA)	Variants Identified	Valid Variants Normalized	Valid Variants Conflated
1. Abate Marie A	1. Abate Marie A, Abate MA	1. Abate Marie A, Abate MA	1. Abate MA
2. Abbott John P	2. Abbott John P, Abbott JP	2. Abbott John P, Abbott JP	2. Abbott JP
3. Abrahamsson S	3. Abrahamsson S, Abrahamsson S	3. Abrahamsson S, Abrahamsson S	3. Abrahamsson S
4. Abrahamsson Sixten	4. Abrahamsson Sixten, Abrahamsson S	4. Abrahamsson Sixten, Abrahamsson S	4. Adams AM
5. Adams Audrey M	5. Adams Audrey M, Adams AM	5. Adams Audrey M, Adams AM	5. Adams DM
6. Adams D M	6. Adams D M, Adams DM	6. Adams D M, Adams DM	6. Adams MQ
7. Adams Michael Q	7. Adams Michael Q, Adams MQ	7. Adams Michael Q, Adams MQ	7. Adams PM
8. Adams Peter M	8. Adams Peter M, Adams PM	8. Adams Peter M, Adams PM	8. Adamson GW
9. Adamson George W	9. Adamson George W, Adamson GW	9. Adamson George W, Adamson GW	9. Aderibigbe MR
10. Aderibigbe M R	10. Aderibigbe M R, Aderibigbe MR	10. Aderibigbe M R, Aderibigbe MR	10. Ainsworth ML
11. Ainsworth Mary Lynn	11. Ainsworth Mary Lynn, Ainsworth ML	11. Ainsworth Mary Lynn, Ainsworth ML	11. Aldrich MJ
12. Aldrich M	12. Aldrich , Aldrich MJ	12. Aldrich M , Aldrich MJ	12. Alexander B
13. Aldrich M J	13. Aldrich M , Aldrich MJ	13. Aldrich M J, Aldrich MJ	13. Allen FH
14. Aldrich Michael	14. Aldrich M J, Aldrich MJ	14. Aldrich Michael , Aldrich MJ	14. Ankeny ML
15. Aldrich Michael J	15. Aldrich Michael , Aldrich MJ	15. Aldrich Michael J, Aldrich MJ	15. Atherton P
16. Aldrich Mike	16. Aldrich Michael J, Aldrich MJ	16. Alexander Steiger Bettie , Alexander B	16. Atkinson SD
17. Alexander Steiger Bettie	17. Alexander Steiger Bettie , Alexander B	17. Allen F H, Allen FH	17. Auster E
18. Allen F H	18. Allen F H, Allen FH	18. Allen Frank H, Allen FH	18. Backus JEB
19. Allen Frank H	19. Allen Frank H, Allen FH	19. Ankeny Melvon, Ankeny ML	19. Bader SA
20. Ankenny Melvon L	20. Ankeny Melvon, Ankeny ML	20. Atherton Cochrane Pauline, Atherton P	20. Badger EW
21. Ankeny Melvon	21. Atherton Cochrane Pauline, Atherton P	21. Atkinson S, Atkinson SD	21. Badler MM
22. Atherton Cochrane Pauline	22. Atkinson , Atkinson SD	22. Atkinson Steven D, Atkinson SD	22. Baker SL
23. Atkinson S	23. Atkinson S, Atkinson SD	23. Auster E, Auster E	23. Bamford CM
24. Atkinson Steve	24. Atkinson Steven D, Atkinson SD	24. Auster Ethel, Auster E	24. Banchik AS
25. Atkinson Steven D	25. Auster E, Auster E	25. Backus Joyce, Backus JEB	25. Banholzer GS
26. Auster E	26. Auster Ethel, Auster E	26. Bader Shelley A, Bader SA	26. Barlow DH
27. Auster Ethel	27. Backus Joyce, Backus JEB	27. Badger Edward W, Badger EW	27. Barnard JM
28. Backus Joyce E B	28. Bader Shelley A, Bader SA	28. Badler Mitchell M, Badler MM	28. Barrie JM
29. Bader Shelley A	29. Badger Edward W, Badger EW	29. Baker S, Baker SL	29. Barthelemy JP
30. Badger Edward W	30. Badler Mitchell M, Badler MM	30. Baker Sharon L, Baker SL	30. Baser KH
31. Badler Mitchell M	31. Baker S, Baker SL	31. Bamford Carol M, Bamford CM	31. Bashshur RL
32. Baker S	32. Baker Sharon L, Baker SL	32. Banchik Andrea S, Banchik AS	32. Bates MJ
33. Baker Sharon L	33. Bamford Carol M, Bamford CM	33. Banholzer Gordon S, Banholzer GS	33. Bawden D
34. Bamford Carol M	34. Banchik Andrea S, Banchik AS	34. Barlow D H, Barlow DH	34. Bayer MP
35. Banchik Andrea S	35. Banholzer Gordon S, Banholzer GS	35. Barlow Derek H, Barlow DH	35. Belkin NJ
36. Banholzer Gordon S	36. Barlow D H, Barlow DH	36. Barnard J M, Barnard JM	
37. Barlow D H	37. Barlow Derek H, Barlow DH	37. Barnard John M, Barnard JM	
38. Barlow Derek H	38. Barnard J M, Barnard JM	38. Barrie Joyce M, Barrie JM	
39. Barnard J M	39. Barnard John M, Barnard JM	39. Barthelemy Jean Pierre, Barthelemy JP	
40. Barnard John M	40. Barrie Joyce M, Barrie JM	40. Baser Kenneth H, Baser KH	
41. Barrie Joyce M	41. Barthelemy Jean Pierre, Barthelemy JP	41. Bashshur Rashid L, Bashshur RL	
42. Barthelemy Jean Pierre	42. Baser Kenneth H, Baser KH	42. Bates M J, Bates MJ	
43. Baser Kenneth H	43. Bashshur Rashid L, Bashshur RL	43. Bates Marcia, Bates MJ	
44. Bashshur Rashid L	44. Bates M J, Bates MJ	44. Bates Marcia J, Bates MJ	
45. Bates M J	45. Bates Marcia, Bates MJ	45. Bawden D, Bawden D	
46. Bates Marcia	46. Bates Marcia J, Bates MJ	46. Bawden David, Bawden D	
47. Bates Marcia J	47. Bawden D, Bawden D	47. Bayer Mark P, Bayer MP	
48. Bawden D	48. Bawden David, Bawden D	48. Belkin N J, Belkin NJ	
49. Bawden David	49. Bayer Mark P, Bayer MP	49. Belkin Nicholas, Belkin NJ	
50. Bayer Mark P	50. Belkin N J, Belkin NJ		
51. Belkin N J	51. Belkin Nicholas, Belkin NJ		
52. Belkin Nicholas			

A further weakness to be acknowledged is that this procedure can correctly identify only the constructions stored in each table due to the fact that the FST parser recognizes all the entries that are reflected in the matrix. A fault would appear if this model were applied to constructions not stored previously in the binary matrices, though some components might coincide with the data associated to tables. Hence, the reference automata would erroneously check some of these features, producing failures in recognition. In a hypothetical application of the parser to a dynamic collection, then, we would have to build a new binary matrix, or update it constantly.

TABLE 6. Precision, recall, F measure, and ICF.

	LISA	SCI-E
Possible Variants	580	234
Non-valid Variants	10	17
Variants Identified	574	220
Valid Variants Normalized	570	217
Valid Variants Conflated	447	205
Precision	0.99	0.98
Recall	0.98	0.92
F_1	0.98	0.95
ICF	0.23	0.12

Conclusions

The development of systems that identify and unify the variants of bibliographic author names is clearly a requirement for IR effectiveness. Within the retrieval of scientific literature in citation indexing systems, this recognition becomes a dire necessity for quality information proceeding from databases. Because many researchers depend on citation statistics, the development of tools for normalizing the data used in bibliometric analysis is a critical issue. In this article, we have proposed a procedure that would solve some of these inaccuracies. From the experiments performed and put forth here, several conclusions can be drawn. First, the precision of the master graph for the analysis and recognition of the variants is very high, though slightly hampered by a problem of overanalysis owing to the fact that some strings contain errors. Second, the recall result of this procedure also is very high, yet there is a problem of underanalysis, likewise resulting from strings with errors. Third, we need to look for complementary solutions involving similarity measures in view of the fact that the margin of error would have been greater if we had tested this technique on another type of collection not as restricted as author indexes from databases.

Finally, we suggest that this descriptive method provides a theoretical model that describes PNs in a systematic form. Name structures may present varied formats, identified here as a combination of variables in which any constants could be inserted. The main strength of this approach is that it allows us to describe, represent, and identify this type of construction in a uniform way, with scientific rigor. Furthermore, the parser based on FSTs conflates the PN variants into equivalence classes, well adapted to many other specific applications such as digital libraries, information extraction, and bibliometrics.

Acknowledgments

The article is an extended version of a chapter in a Ph.D. thesis by C. Galvez at the Department of Information Science, University of Granada, under the supervision of Dr. F. Moya-Anegón. We thank two anonymous referees for their valuable comments and suggestions to enhance this work.

References

Accomazzi, A., Eichhorn, G., Kurtz, M.J., Grant, C.S., & Murray, S.S. (2000). The NASA astrophysics data system: Architecture. Astronomy and Astrophysics Supplement Series, 143(1), 41–59.

Angell, R.C., Freund, G.E., & Willett, P. (1983). Automatic spelling correction using a trigram similarity measure. Information Processing & Management, 19(4), 255–261.

Auld, L. (1982). Authority control: An eighty-year review. Library Resources and Technical Services, 26, 319–330.

Bagga, A., & Baldwin, B. (1998). Entity-based cross-document co-referencing using the vector space model. Proceedings of the 17th International Conference on Computational Linguistics (pp. 79–85). Montreal: ACL.

Baluja, S., Mittal, V., & Sukthankar, R. (2000). Applying machine learning for high performance name-entity extraction. Computational Intelligence, 16(4), 586–595.

Belkin, N.J., & Croft, W.B. (1987). Retrieval techniques. Annual Review of Information Science and Technology, 22, 109–145.

Bikel, D.M., Miller, S., Schwartz, R., & Weischedel, R. (1997). Nymble: A high-performance learning name-finder. Proceedings of the 5th Conference on Applied Natural Language Processing (pp. 194–201). San Francisco: Kaufmann.

Blair, C.R. (1960). A program for correcting spelling errors. Information and Control, 3, 60–67.

Borgman, C.L., & Siegfried, S.L. (1992). Getty's synoname and its cousins: A survey of applications of personal name-matching algorithms. Journal of the American Society for Information Science, 43(7), 459–476.

Bouchard, G., & Pouyez, C. (1980). Name variations and computerized record linkage. Historical Methods, 13, 119–125.

Bourne, C.P. (1977). Frequency and impact of spelling errors in bibliographic data bases. Information Processing & Management, 13(1), 1–12.

Chinchor, N. (1997). Named entity task definition (Version 3.5). Proceedings of the 7th Message Understanding Conference. Fairfax, VA: Morgan Kaufmann.

Chomsky, N. (1957). Syntactic structures. The Hague, The Netherlands: Mouton.

Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text (pp. 136–143). Second Conference on Applied Natural Language Processing. Austin, TX: ACL.

Church, K., & Hanks, P. (1990). Word association norms, mutual information and lexicography. Computational Linguistics, 16, 22–29.

Croft, W.B., & Xu, J. (1995). Corpus-specific stemming using word form co-occurrence. Proceedings for the 4th annual Symposium on Document Analysis and Information Retrieval (pp. 147–159). Las Vegas, Nevada.

Cronin, B., & Snyder, H.W. (1997). Comparative citation ranking of authors in monographic and journal literature: A study of sociology. Journal of Documentation, 53(3), 263–273.

Coates-Stephens, S. (1993). The analysis and acquisition of proper names for the understanding of free text. Computers and the Humanities, 26(5–6), 441–456.

Cucerzan, S., & Brill, E. (2004). Spelling correction as an iterative process that exploits the collective knowledge of web users. In D. Lin, & D. Wu (Eds.), Proceedings of EMNLP 2004 (pp. 293–300). Barcelona: Association for Computational Linguistics.

Damerau, F.J. (1964). A technique for computer detection and correction of spelling errors. Communications of the ACM, 7(4), 171–176.

Damerau, F.J., & Mays, E. (1989). An examination of undetected typing errors. Information Processing & Management, 25(6), 659–664.

Davidson, L. (1962). Retrieval of misspelled names in an airlines passenger record system. Communications of the ACM, 5(3), 169–171.

Frakes, W.B., & Fox, C.J. (2003). Strength and similarity of affix removal stemming algorithms. ACM SIGIR Forum, 37(1), 26–30.

French, J.C., Powell, A.L., & Schulman, E. (2000). Using clustering strategies for creating authority files. Journal of the American Society for Information Science and Technology, 51(8), 774–786.

Gadd, T.N. (1988). Fisching for werds: Phonetic retrieval of written text in information systems. Program: Automated Library and Information Science, 22(3), 222–237.

Gadd, T.N. (1990). PHONIX: The algorithm. Program: Automated Library and Information Science, 24(4), 363–366.

Gaizauskas, R., Wakao, T., Humphreys, K., Cunningham, H., & Wilks, Y. (1995). University of Sheffield: Description of the LaSIE system as used for MUC-6. Proceedings of the 6th Message Understanding Conference (pp. 207–220). Columbia, MD: NIST, Kaufmann.

Garfield, E. (1979). Citation indexing—Its theory and application in science, technology, and humanities. New York: Wiley.

Garfield, E. (1983a). Idiosyncrasies and errors, or the terrible things journals do to us. Current Contents, 2, 5–11.

Garfield, E. (1983b). Quality control at ISI. Current Contents, 19, 5–12.

Giles, C.L., Bollacker, K., & Lawrence, S. (1998). CiteSeer: An automatic citation indexing system. In I. Witten, R. Akscyn, & F.M. Shipman III (Eds.), Digital libraries 98—The 3rd ACM Conference on Digital Libraries (pp. 89–98). New York: ACM Press.

Gooi, C.H., & Allan, J. (2004). Cross-document coreference on a large scale corpus. In S. Dumais, D. Marcu, & S. Roukos (Eds.), Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (pp. 9–16). Boston, MA: ACL.

Gross, M. (1975). Méthodes en syntaxe. Paris: Hermann.

Gross, M. (1997). The construction of local grammars. In E. Roche & Y. Schabes (Eds.), Finite-state language processing (pp. 329–352). Cambridge, MA: MIT Press.

Hall, P.A.V., & Dowling, G.R. (1980). Approximate string matching. Computing Surveys, 12(4), 381–402.

Han, H., Giles, L., Zha, H., Li, C., & Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries (pp. 296–305). Tucson, AZ: ACM.

Han, H., Zha, H., & Giles, C.L. (2005). Name disambiguation in author citations using a K-way spectral clustering method. Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries. Denver, CO: ACM.

Harris, Z.S. (1951). Methods in structural linguistics. Chicago: University of Chicago Press.

Hayes, P. (1994). NameFinder: Software that finds names in text. Proceedings of 4th RIAO Conference of Computer Assisted Information Searching on the Internet (pp. 762–774), New York: Rockefeller University.

Hermansen, J. (1985). Automatic name searching in large data bases of international names. Unpublished doctoral dissertation, Washington, DC: Georgetown University, Department of Linguistics.

Hopcroft, J.E., & Ullman, J.D. (1979). Introduction to automata theory, languages, and computation. Reading, MA: Addison-Wesley.

Hull, J.J. (1992). A hidden Markov model for language syntax in text recognition. Proceedings of the 11th IAPR International Conference on Pattern Recognition (pp. 124–127). The Hague, The Netherlands: IEEE Computer Society Press.

Jaro, M.A. (1989). Advances in record-linkage methodology as applied to matching the 1985 Census of Tampa, Florida. Journal of the American Statistical Association, 89(406), 414–420.

Jaro, M.A. (1995). Probabilistic linkage of large public health data files. Statistics in Medicine, 14(5–7), 491–498.

Knuth, D. (1973). The art of computer programming. Reading, MA: Addison-Wesley.

Kukich, K. (1992). Techniques for automatically correcting words in texts. ACM Computing Surveys, 24(4), 377–439.

Landau, G.M., & Vishkin, U. (1986). Efficient string matching with k mismatches. Theoretical Computer Science, 43, 239–249.

Levenshtein, V.I. (1965). Binary codes capable of correcting spurious insertions and deletions of ones. Problems of Information Transmission, 1(1), 8–17.

Lin, X., White, H.D., & Buzydlowski, J. (2003). Real-time author co-citation mapping for online searching. Information Processing and Management, 39(5), 689–706.

Mann, G., & Yarowsky, D. (2003). Unsupervised personal name disambiguation. In W. Daelemans & M. Osborne (Eds.), Proceedings of CoNLL-2003 (pp. 33–40). Edmonton, Canada: ACL.

Mays, E., Damerau, F.J., & Mercer, R.L. (1991). Context based spelling correction. Information Processing & Management, 27(5), 517–522.

McCain, K.W. (1990). Mapping authors in intellectual space: A technical overview. Journal of the American Society for Information Science, 41(6), 433–443.

Melamed, I.D. (1999). Bitext maps and alignment via pattern recognition. Computational Linguistics, 25(1), 107–130.

Moed, H.F., & Vriens, M. (1989). Possible inaccuracies occurring in citation analysis. Journal of Information Science, 15(2), 95–107.

MUC-4. (1992). Proceedings of the 4th Message Understanding Conference. McLean, VA: Kaufmann.

MUC-6. (1995). Proceedings of the 6th Message Understanding Conference. Columbia, MD: Kaufmann.

MUC-7. (1997). Proceedings of the 7th Message Understanding Conference. Fairfax, VA: Kaufmann.

Navarro, G., Baeza-Yates, R., & Arcoverde, J.M.A. (2003). Matchsimile: A flexible approximate matching tool for searching proper names. Journal of the American Society for Information Science and Technology, 54(1), 3–15.

Paik, W., Liddy, E.D., Yu, E., & McKenna, M. (1993). Categorizing and standardizing proper nouns for efficient information retrieval. In B. Boguraev & J. Pustejovsky (Eds.), Corpus processing for lexical acquisition (pp. 44–54). Cambridge, MA: MIT Press.

Patman, F., & Thompson, P. (2003). Names: A new frontier in text mining. NSF/NIJ Symposium on Intelligence and Security Informatics, Lecture Notes in Computer Science (pp. 1–3). Berlin: Springer-Verlag.

Pedersen, T., Purandare, A., & Kulkarni, A. (2005). Name discrimination by clustering similar contexts. Proceedings of the 6th International Conference and Computational Linguistics (pp. 226–237). Mexico City, Mexico: ACL.

Pereira, F. (1997). Sentence modeling and parsing. In R.A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, & V. Zue (Eds.), Survey of the state of the art in human language technology (pp. 130–140). Cambridge, MA: Cambridge University Press.

Petersen, J.L. (1986). A note on undetected typing errors. Communications of the ACM, 29(7), 633–637.

Pfeiffer, U., Poersch, T., & Fuhr, N. (1996). Retrieval effectiveness of proper name search methods. Information Processing & Management, 32(6), 667–679.

Philips, L. (1990). Handing on the metaphone. Computer Language, 7(12), 39–43.

Pollock, J.J. (1982). Spelling error detection and correction by computer: Some notes and a bibliography. Journal of Documentation, 38(4), 282–291.

Pollock, J.J., & Zamora, A. (1983). Collection and characterization spelling errors in scientific and scholarly text. Journal of the American Society for Information Science, 34(1), 51–58.

Pollock, J.J., & Zamora, A. (1984). Automatic spelling correction in scientific and scholarly text. Communications of the ACM, 27(4), 358–368.

Ravin, Y., & Wacholder, N. (1996). Extracting names from natural-language text. Research Report RC 20338, IBM Corporation.

Rice, R.E., Borgman, C.L., Bednarski, D., & Hart, P.J. (1989). Journal-to-journal citation data: Issues of validity and reliability. Scientometrics, 15(3–4), 257–282.

Riseman, E.M., & Ehrich, R.W. (1971). Contextual word recognition using binary digrams. IEEE Transactions on Computers, C-20, 397–403.

Roche, E. (1993). Analyse syntaxique transformationelle du français par transducteurs et lexique-grammaire. Unpublished doctoral dissertation, Université Paris.

Roche, E. (1999). Finite state transducers: Parsing free and frozen sentences. In A. Kornai (Ed.), Extended finite state models of language (pp. 108–120). Cambridge, UK: Cambridge University Press.

Roche, E., & Schabes, Y. (1995). Deterministic part-of-speech tagging with finite state transducers. Computational Linguistics, 21(2), 227–253.

Rogers, H.J., & Willett, P. (1991). Searching for historical word forms in text databases using spelling-correction methods: Reverse error and phonetic coding methods. Journal of Documentation, 47(4), 333–353.

Ruiz-Perez, R., Delgado López-Cózar, E., & Jiménez-Contreras, E. (2002). Spanish personal name variations in national and international biomedical databases: Implications for information retrieval and bibliometric studies. Journal of the Medical Library Association, 90(4), 411–430.

Russell, R.C. (1918). United States Patent No. 1261167. Washington, DC: U.S. Patent Office.

Salton, G. (1989). Automatic text processing: The transformation, analysis and retrieval of information by computer. Reading, MA: Addison-Wesley.

Schulz, K.U., & Mihov, S. (2002). Fast string correction with Levenshtein-automata. International Journal of Document Analysis and Recognition, 5(1), 67–85.

Senellart, J. (1998). Locating noun phrases with finite state transducers. Proceedings of the 17th International Conference on Computational Linguistics (pp. 1212–1219). Montreal: COLING.

Sher, I.H., Garfield, E., & Elias, A.W. (1966). Control and elimination of errors in ISI services. Journal of Chemical Documentation, 6(3), 132–135.

Siegfried, S., & Bernstein, J. (1991). Synoname: The Getty's new approach to pattern matching for personal names. Computers and the Humanities, 25(4), 211–226.

Silberztein, M. (1993). Dictionnaires électroniques et analyse automatique de textes: Le systëme INTEX. Paris: Masson.

Silberztein, M. (2000). INTEX: An FST Toolbox. Theoretical Computer Science, 231(1), 33–46.

Spink, A., Jansen, B.J., & Pedersen, J. (2004). Searching for people on Web search engines. Journal of Documentation, 60(3), 266–278.

Strunk, K. (1991). Control of personal names. Cataloging & Classification Quarterly, 14(2), 63–79.

Taft, R.L. (1970). Name search techniques (Special Report No. 1). Albany, NJ: Bureau of Systems Development, New York State Identification and Intelligence Systems.

Tagliacozzo, R., Kochen, M., & Rosenberg, L. (1970). Orthographic error patterns of author names in catalog searches. Journal of Library Automation, 3, 93–101.

Takahashi, H., Itoh, N., Amano, T., & Yamashita, A. (1990). A spelling correction method and its application to an OCR system. Pattern Recognition, 23(3–4), 363–377.

Tao, H., & Cole, C. (1991). Wade-Giles or Hanyu-Pinyin: Practical issues in the transliteration of Chinese titles and proper names. Cataloging & Classification Quarterly, 12(2), 105–124.

Taylor, A.G. (1984). Authority files in online catalogs: An investigation of their value. Cataloging & Classification Quarterly, 4(3), 1–17.

Taylor, A.G. (1989). Research and theoretical considerations in authority control. In B.B. Tillett (Ed.), Authority control in the online environment: Considerations and practices (pp. 29–56). New York: Haworth.

Thompson, P., & Dozier, C.C. (1999). Name recognition and retrieval performance. In T. Strzalkowski (Ed.), Natural language information retrieval (pp. 25–74). Dordrecht, The Netherlands: Kluwer.

Tillett, B.B. (1989). Considerations for authority control in the online environment. In B.B. Tillett (Ed.), Authority control in the online environment: Considerations and practices (pp. 1–11). New York: Haworth.

Torvik, V.I., Weeber, M., Swanson, D.R., & Smalheiser, N.R. (2005). A probabilistic similarity metric for Medline records: A model for author name disambiguation. Journal of the American Society for Information Science and Technology, 56(2), 140–158.

Ukkonen, E. (1992). Approximate string matching with Q-grams and maximal matches. Theoretical Computer Science, 92(1), 191–212.

Ullmann, J.R. (1977). A binary N-gram technique for automatic correction of substitution, deletion, insertion and reversal errors. The Computer Journal, 20(2), 141–147.

van Rijsbergen, C.J. (1979). Information retrieval. London: Butterworths.

Wacholder, N., Ravin, Y., & Choi, M. (1997). Disambiguation of proper names in text. Proceedings of the 5th Conference on Applied Natural Language Processing (pp. 202–208). Washington, DC: ACL.

Wacholder, N., Ravin, Y., & Byrd, R.J. (1994). Retrieving information from full text using linguistic knowledge. Proceedings of the 15th National Online Meeting (pp. 441–447). Yorktown Heights, NJ: IBM T.J. Watson Research Center.

Weintraub, T. (1991). Personal name variations: Implications for authority control in computerized catalogs. Library Resources and Technical Services, 35, 217–228.

Winkler, W.E. (1999). The state of record linkage and current research problems. Research Report No. RR99/04. Washington, DC: U.S. Bureau of the Census, Statistical Research Division.

Woods, W.A. (1970). Transition network grammars for natural language analysis. Communications of the ACM, 13(10), 391–606.

Xu, J., & Croft, B. (1998). Corpus-based stemming using co-occurrence of word variants. ACM Transactions on Information Systems, 16(1), 61–81.

Zamora, E.M., Pollock, J., & Zamora, A. (1981). The use of trigrams analysis for spelling error detection. Information Processing and Management, 17(6), 305–316.

Zobel, J., & Dart, P. (1995). Finding approximate matches in large lexicons. Software Practice and Experience, 25(3), 331–345.