Computer-Based Name Searches

Alexander Beider

Independent scholar, Paris, France
Email: albeider@yahoo.fr

The paper discusses general features of several systems of phonetic matching (American Soundex, Daitch-Mokotoff Soundex, Metaphone, Double Metaphone) that allow to match various name spellings by assigning numerical or alphabetical values to various letters and letter combinations. One of the most important drawbacks of these systems consists in a large number of "false positives" they generate. In other words, they consider matching numerous names that are totally unrelated. The paper presents the principles used in Beider-Morse Phonetic Matching that was elaborated as an alternative to the Soundex approaches. It has an ambition to obtain a significantly smaller number of "false positives" without introducing a considerable number of "false negatives". These methods are complementary to various algorithms of fuzzy string searching known in the computer sciences. Finally, the approach of using "synonyms" can provide helpful information that is not retrieved by other methods.

I. Introduction

When searching names, one can often face the same major problem: several names written in different ways can actually represent variant spellings of the same name. This problem is particularly acute for countries there numerous immigrants live. For example, various Americans named Schwarz, Schwartz, Schvarz, Schvartz, Schwarc, Shvarts, Shwarz, Shwartz, Shvarz, Shvartz, Svarc, Svartz, Szwarc, Szwartz, Szwarz, Szvartz, Szvarz, Sjvarts, Chvarts, Chvartz, Chvarz, Chwartz, and even Svart, can, in principle, be relatives. For all of them, their family names can be of the same origin, derived from the German adjective *schwarz* 'black' or the Yiddish word cognate to it. Many forms from this list can be pronounced by their bearers in US in the same way. However, even if two of these forms are pronounced distinctly for American families, it is still possible that in the European countries from where these names originated they were pronounced identically. Indeed, in German, Schwarz and Schwartz are just variant spellings of the same name.

When migrants move from one country to another, even if in both countries local languages use Roman characters, their surnames can undergo several kinds of changes. Names can be re-transcribed/transliterated taking into account their pronunciation and following the rules of local orthography. This way

in France the name in question could be transcribed as Chvarts. Similarly, Shvarts, Sjvarts, Svarc, Švarc, Szwarc, and Şvarț represent its transcriptions in English, Dutch, Hungarian, Czech, Polish, and Romanian, respectively. Since in many countries people may be aware about the existence of the German adjective *schwarz*, hybrid spellings can also start to be used such as Shvartz (English *Shv-*, but German *-tz*), Chwartz (French *Ch-*, but German *-wartz*), Schwarc (German *Sch-*, but Polish *–c*) etc. Diacritic signs unknown in the graphic system of the new area can be easily dropped and consequently in English-speaking countries Czech Švarc and Romanian Şvarț turn into Svarc and Svart, respectively. The most significant changes occur if one passes from one alphabet to another. For example, the same name as the above can be spelled Шварць (before 1918) or Шварц (after 1918) in Russian, *שוורץ* or *שבארץ* in Hebrew, *שוואַרץ* in Yiddish, Σβαρτς in Greek, and *شفارتس* in Arabic. Not all of the above graphic forms result from the re-transcription of names of immigrants. Certain graphic forms in question were valid already for their first bearers when scribes in the original countries recorded names adopted by local families of either German Christian or Jewish origin. To make things more complicated, in several cultures names can vary depending on the gender of the person, the grammatical case and/or be different in singular and plural forms. These peculiarities are particularly common in Slavic languages. Compare the following Russian forms: Пушкин (Pushkin, masculine nominative singular); Пушкины (Pushkiny, masculine nominative plural); Пушкина (Pushkina, either feminine nominative or masculine genitive, both singular). In the 19th-century Poland, men, married women and unmarried girls / women all had different forms of names. For example, the father was Szewc, the mother appeared in sources as Szewcowa, and their daughter was called Szewcówna.

Several methods can be used for searching of different spelling forms that can, in theory, correspond to the same original name. Evidently, the most common approach is just manual / visual. However, its results are heavily dependent on linguistic skills of the researcher. The procedure of manual search for various possible variants can also be extremely time-consuming. For these reasons, during the last 100 years several methods of automatic searches were suggested. Their importance and usefulness grew dramatically during the last decades with the development of personal computers and Internet.

One can distinguish several independent basic approaches in automatic searching. For example, one can make a search for all forms having the same sets of characters. Someone who wants to search for Washington, can look for all names starting with *Wash-*, and/or ending in *–gton*, and/or having the cluster *–shing-,* and/or including such letters or letter combinations as “w”, “sh”, and “t”. The main disadvantage of this method consists in the necessity for the user to make his/her choices of the letters to include or to exclude. This feature places this category of searches close to the manual ones. Other methods make everything automatically. The user needs only to enter the name to be searched.

This paper discusses automatic methods of name searching. They can be classified into three major groups. The first one deals with the phonetics and searches for equivalents among forms that are written differently but pronounced identically or close to each other. It is covered in Chapter 1. The second one—discussed in Chapter 2—deals with graphic errors. The third group covered in Chapter 3 deals with synonyms. These groups are complementary. In some applications, the use of all of three of them can be helpful.

II. Soundex and Other Phonetic Methods

a. Terminology

The term *soundex* comes from the contraction of two words: *sound* and *index*. As its name indicates, the purpose of making a *soundex* consists of sorting names not according to their spelling (as in a classical index using the alphabetical order of characters) but according to their pronunciation. Simply put, *soundex* is an encoding of a name such that names that sound the same will get the same encoding. A search application based on *soundex* will look for matches of the *soundex* code rather than matches of the name itself, thereby finding all names that sound like the name being sought.

If two names matching according to a *soundex* method actually cannot correspond to the same original name, one say that this pair of names are "false positives" for this method. Alternatively, if two names not matching according to a method, in reality correspond to the same original name, this pair is called "false negatives". These two terms serve in main functional criteria that are generally used to measure a performance of various *soundex* methods. The smaller is the number of pairs of names that represent "false positives" and "false negatives", the better is the method using for matching.

b. American Soundex

The first *soundex* system was patented in 1918 by Robert Russell. Its first applications appeared during the 1930s when a variation of Russell's method, called the *American Soundex Code* was designed and used by the US *Census Bureau* to facilitate name searches in the census. This amended method suggests the following coding rules:

(1) The initial letter is retained without any change.

(2) All letters corresponding to vowels or semi-vowels ("a", "e", "i", "o", "u", "y", and "w"), as well as the letter "h" are ignored.

(3) The remaining consonants are replaced with figures representing their conventional *soundex* codes:

1 for "b", "p", "v", and "f"
2 for "c", "g", "j", "k", "q", "s", "x", and "z"
3 for "d" and "t"

4 for “l”
5 for “m” and “n”
6 for “r”

(4) If two or more consecutive letters have the same *soundex* code, one of them is ignored.

(5) Every name is coded as a string of four characters, one letter and three figures. Consequently, if there are fewer than three encoded figures, the balance is filled with zeroes. On the other hand, once the length of four is achieved, the remaining part of the name is truncated and all following characters are ignored.

Looking to the above encoding rules, one can easily realize that this method is primarily based on the pronunciation. Indeed, the digit 1 corresponds to close sounds: in general phonetics, the sounds /b/ and /p/ are both dentals, while /v/ and /f/ are labiodentals. Both /m/ and /n/ (coded 5) are nasal consonants. The only difference between /d/ and /t/ (both coded 3) consists in the fact that the former is voiced, while the latter is unvoiced. In a number of languages, “h” is not pronounced. Vowels change most often from one dialect to another or even in the individual pronunciation. For this reason, ignoring them during the coding can allow finding related names. As an example, the *American Soundex* code for Schwarzenegger is S625: “S” kept, “c” ignored because it follows “S” that has the same code, “h”, “w”, and “a” are always ignored, “r” gives 6, “z” is coded 2, “e” is ignored, “n” corresponds to 5, other letters are ignored because the total length of the code reached already its limit of four characters. If the name was misspelled as Shwarzeneg(g)er, the code would still be S625, so any search application based on American Soundex would still find the match in spite of that misspelling. Smith, Smit, Schmidt, Schmitt, Schmied, Schmit, Shmit, and Smet correspond all to S530. Miller, Müller, Mailer, Moller, Muler, and Mular are all M460. The code M600 is valid for Mayer, Meyer, Maier, Meier, and Meir. Both Braun and Brown are coded B650.

Using the *American Soundex* in a search application has several major drawbacks. Firstly, it classification of consonants in based on English orthography and does not take into account peculiarities of other languages that also use Latin characters. For example, if the English “w” is a sign for a semi-vowel and for this reason its ignoring can be considered justified, in Polish and German this letter is pronounced exactly as English “v”. Consequently, it would be more appropriate to code “w” following two possible ways (and not just one of them): ignoring it or assigning to it the code 1. This approach would find equivalence between Ivanov, Iwanow, Ivanow, and Iwanoff. Without this amendment, the corresponding codes are I151, I500, I150, and I510, respectively, that is, none of these variant spellings of the same name matches to others. These examples are “false negatives” for the *American Soundex*. A second large group of “false negatives” appear because of the individual processing of any letter. Yet, in many languages sounds are designated not only by single letters, but by their combinations as well. For

example, English “ts” sounds exactly as German “z”. For this reason, if Schwarzenegger is misspelled as Schwartsenegger, the *American Soundex* code would be S632 instead of S625 and a search application based on the *American Soundex* would not find the match with that misspelling. The number of “false positives” generated by this method is also huge. The limitation of the code length to four characters brings together (under the code P362) such different surnames as Peters, Peterson, Petersman, Petersmark, Petesmeyer, Petersteig, and Peterstrom. The exclusion of vowels provokes numerous incorrect matches as, for example, Shander and Schneider (S535), Baron and Braun (B650), Kuhn and Knee (K500). Generally speaking, several major rules of this algorithm appear to be totally conventional and therefore not very appropriate:

- Why is the length limited to 4 (and not, for example, to 5 or 6)?
- Why is the first character of the code a letter, while all others are numbers?
- For what reason one distinguishes initial vowels, but not the internal or the final ones?

c. Daitch-Mokotoff Soundex

In 1985, Randy Daitch and Gary Mokotoff suggested a series of significant amendments to the *American Soundex code* that resulted in a major improvement of its capabilities. The resulting system received the name of the *Daitch-Mokotoff Soundex* (in this paper, DMS). Its main characteristics are:

- Changes of the structure
    (1) Any DMS code has only figures (including a code for the initial letter)
    (2) Its total length equals to 6 instead of 4
- Exactly as in the *American Soundex*:
    (1) For short names, remainder is filled with zeroes
    (2) Consecutive letters with the same soundex code are coded as a single letter
- Most important changes are the following ones:
    (1) Several letter combinations are coded together if they correspond to one sound
    (2) Coding of letters can differ depending on the context: one code can be applicable at the start of a name (initial position), another code can be used before a vowel (“prevocalic position”), and a third one in other positions
    (3) If letter(s) can correspond to more than one sound, they aree coded in several alternate ways.

During the last decades, the authors of this system made slight changes to their exact coding rules. The version that was applicable in 2008 was characterized by the following conventions:

- 0 for initial vowels “a”, “e”, “i”, “o”, and “u” as well as for all combinations at the start of a name between these vowels, “y” and “j” except for “eu” (examples: “ai”, “au”, “ei”, “ey”, “ej”, “oe”, and “uo”)
- 1 for “eu”, “ij”, iy”, “yj”, “yy” in all positions; “y”, “ia”, “ie”, “io”, “iu”, “ya”, “ye”, “yo”, and “yu” in initial or prevocalic positions; all non-initial combinations of two letters that have 0 in the initial position
- No code: for “a”, “e”, “i”, “o”, “u”, and “y” in contexts not mentioned till now; “h” when non-initial and not prevocalic
- 2 for a number of letter combinations at the start of a word such as “st”, “sc”, “schts(c)h”, “schtch”, “sh(t)sch”, “shtsh”, “s(c)ht”, “schd”, “st(s)ch”, “strz”, “strs”, “stsh”, “szcz”, “szcs”, “szt”, “shd”, “s(z)d”, “zdz(h)”, “zhdzh”, and “z(h)d”
- 3 for “d”, “t”, and “th”
- 4 for “s”, “z” and numerous combinations that include these letters (examples: “cs”, “cz”, csz”, “ds”, “dsh”, “dz”, “sz”, “sch”, “tch”, “ts”, and “tz”). The letter combinations that have 2 in the initial position are coded 43 if they end in “d” or “t”; others are coded 4
- 5 for “g”, “k”, “q”, “kh”, “h” (initial or prevocalic), initial “x”. However, in other positions, the same letter “x” is coded 54
- 6 for “m” and “n”
- 7 for “b, “p”, “v”, “f”, “w”, “ph”, and internal prevocalic “au”
- 8 for “l”
- 9 for “r”.

Among the letters or letter combinations that allow for multiple coding are: “c” and “ch” (4 and 5), “ck” (5 and 45), “j” (1 and 4), “rs” and “rz” (94 and 4).

As it can be seen from the above table, coding rules of DMS are significantly more sophisticated than those used in the *American Soundex*, the latter algorithm being rather simplistic. However, in principle, looking into the table listing all DMS rules, users can construct the DMS code of the name of their interest without computer. A quick look into the list of letter combinations treated in a special way shows that—in addition to English—it mainly takes into account orthography of German, Polish, and English transcriptions of names originally spelled in various Slavic languages. This property is not fortuitous. Both authors of DMS are American Jewish genealogists and their optimization of *American Soundex* was primarily designed to make it suitable for treating names of Ashkenazic Jews that are often based on Slavic languages or German. Jewish migrants from Eastern Europe could use one spelling in their country of origin and another spelling in their new countries. This situation is plausible for several

reasons. First, surnames from the Russian Empire originally spelled using the Cyrillic alphabet, were respelled in Latin characters in the Americas, Western Europe, Australia, and South Africa. Secondly, even within the same country, numerous spelling variations existed for the same family names. For example, the name of the same Polish Jew might appear in Polish civil records as Grynsztajn, Grinsztajn, Grünstein, Grynstein, Grynstain, Grynsztayn, Grinszteyn, Grinstain, and other variants. In DMS, all these spellings correspond to the same code: 596436.

Nevertheless, nothing in this algorithm is specifically Jewish. The same approach is perfectly applicable to all non-Jewish names from the countries where Ashkenazic Jews lived. For example, under DMS, Austrian/German name Schwarzenegger has two codes, namely 474659 and 479465. The spelling Shwarzenegger has the same two codes, while another incorrect spelling, Schwartsenegger, has the code of 479465, which is one of the two codes for the correct spelling. So a search application based on DMS would find the match with either of these misspellings. Moreover, since numerous graphic conventions are shared by various cultures that use Latin characters, with or without diacritic signs, it appears that DMS is suitable for certain languages and countries for which it was not primarily designed. Generally speaking, one can say that languages are better than others adapted for DMS if they have a small number of (1) letters whose pronunciation depend on the context, (2) special letters or diacritic signs. It is much easier to amend DMS to fix the second issue than the first one. Indeed, by introducing into the coding tables the appropriate codes for special characters the second series of issues can be fixed. For example, to pronounce Turkish correctly, it suffices knowing the phonetic values of all 27 letters of its Latin-based alphabet created in 1928. The same letter is pronounced the same way in any word and it does not combine with other letters to denote different single sounds. Basically, the same is true for Czech. Its alphabet includes only one letter combination, “ch”. DMS possesses already a special rule for it because the same combination (and same pronunciation) is present in Polish and German too. As a result, if one introduces into DMS rules an appropriate coding for Turkish and Czech special characters, both these languages immediately become perfectly suitable for DMS. On the other hand, this system is much less suitable for French. Indeed, numerous French letters can be not pronounced at all. It is the case of the final “s” in Dumas, “lt” in Renault, “d” in Monod, and “t” in Petit. Some internal letters can also be mute as “p” in Compte and “s” in Miromesnil. DMS is not designed to cover such cases.

It goes without saying that according to all criteria, DMS is a more powerful tool for name searching applications than its predecessor, the *American Soundex*. The number of pairs that represent “false negatives” for DMS is small. Its design is flexible and allows introducing additional details (namely, special coding for some particular letter combinations) to exclude “false negatives”. Its primarily limitation lies in the fact that many unrelated surnames that never could

represent alternate spellings of the same name receive the same numerical value. As a result, when searching for a specific name, the researcher usually must search through a large set of names that include numerous irrelevant items. For individual searches, this feature is minor, albeit annoying at times. To illustrate the problem of such "false positives" consider several pairs of Polish Jewish surnames, in which the numerical DMS value of both elements is identical, while the surnames are obviously independent: Żaba and Sowa (470000), Ptakowicz and Witkiewicz (735740), Agnes and Eugieniusz (056400), Akcyg [German Ackzig] and Ochocki (054500), Augienfisz [Augenfisch] and Okuniewicz (056740), Drzewienko and Szybniak (476500), Szajnwar [Scheinwahr] and Sznaper [Schnap(p)er] (467900). The inclusion of numerous unrelated names within the same DMS code precludes use of this method for the automatic processing of large lists of names for whose elements one wants to discover equivalents in another list.[1]

d. Metaphone and Double Metaphone

Another attempt of ameliorating *American Soundex* was realized in 1990 when Lawrence Philipps published a new phonetic algorithm that he called *Metaphone* (in this paper, PM). This method, totally independent from DMS, allows indexing words (including names) according to their English pronunciation. It possesses two major changes in comparison to its predecessors that make it much more suitable in comparison to them for analyzing English words and names. Firstly, the code length varies depending of the length of the original word. No limitation exists for the number of characters present in a code. This contrasts to the fixed length of four and six characters used in two previous systems. Secondly, in any position of the code one can find one of the 16 possible characters that correspond to various consonants. It can be reminded that in any position of the *American Soundex* code (except for the first one) only six different figures (from 1 to 6) are possible, while DMS makes use of all ten possible figures (from 0 to 9). This feature shows that PM makes more subtle distinction between English sounds. The 16 characters for consonants in PM are: 0 for "th"; B for "b"; F for "f", "ph", and "v"; H for certain "h" (initial, intervocalic, after consonants other than "c", "g", "p", "s", and "t"); J for "j", "dge", "dgi", "dgy", "ge", "gi", gy" (the last three if not double "gg"); K for "c" (if not before "e", "i", and "y"), "ck", "gg", "g" (if not before "e", "i", and "y"), "k", and "q"; L for "l"; M for "m" and final "mb"; N for "n"; P for "p"; R for "r"; S for "s"; "ce", "ci, "cy" and "z"; T for "d" and "t"; W for "w" followed by a vowel and the initial "wh"; X for "ch", "cia", "sia", "sh", "sio", "tch", "tia", "tio"; Y for "y" followed by a vowel. The letter "x" gives the code S at the start of a word and KS in other positions. In the following

[1] One cannot, for example, use DMS to make an automatic match between surnames included in the Ellis Island Passenger Lists and those that appear in different other large genealogical sources prepared by the representatives of the Mormon Church.

combinations of two letters present at the start of a word the first letter is dropped: “ae”, “gn”, “kn”, “pn”, and “wr”. The letter “g” is dropped in the following combinations: (1) “gh” when not at the end or before a vowel; (2) “gn” and “ned” at the end of a word. Similarly to the *soundex* algorithms, the second letter of doubled letters is ignored (in PM, except for “c”). The vowels “A”, “E”, “I”, “O”, “U” are kept in the initial position. In other contexts they are dropped.

Several choices made by Philipps in his algorithm were questionable. Firstly, it is unclear why five vowels are all differentiated at the start of the name but ignored in other positions. This seems to be a feature partly inherited from the *American Soundex* rather than a characteristic that correctly describes the English phonetics. In principle, in many contexts English distinguishes between the pronunciation of the same vowel letter in open and closed syllables. Initial position is just a particular case of open syllable. Secondly, one can observe that contrary to its predecessors, PM assigns different codes to the letters “b” and “p”, but uses one code for both “d” and “t” and another code for both “g” and “k”. Yet, in English (and many other languages) the phonetic relation between the sounds /b/ and /p/ is exactly the same as that between /d/ and /t/ or /g/ and /k/. In all these cases we are dealing with pairs of stops in which one is voiced and another is its unvoiced equivalent. Differences in coding of “i” and “y” produced a number of “false negatives” (mainly when dealing with diphthongs) that were unknown in previous *soundex* systems. Among the examples are: Eye (code EY) and Eie (code E).

The fact that PM makes an appropriate phonetic analysis for English and its general character (it is applicable for words in general and not only for names) contributed to a large use of this algorithm in various English spell-checking applications, that is, in a domain for which DMS is of no use. It is not a surprise that its example was followed by the creation of similar algorithms applicable to other languages: *Spanish Metaphone, Brazilian Metaphone* etc. All of them deal with only one language at the same time. They can be primarily used for spell-checking purposes. For searching names, they are applicable only if names are spelled according to the orthography of the same language. In a context of names of different origins, PM and its equivalents for languages other than English are clearly inappropriate.

In 2000, Philipps suggested a new version of his algorithm called *Double Metaphone* (in this paper, PDM). Several incoherent elements of the original algorithm were withdrawn including the difference between “b” and “p” (both are now coded “P”) and the coding for initial vowels (all are now coded “A”). The coding rules for the letter “w” changed: it is either coded “F” or assimilated to vowels (and therefore either dropped or coded “A”). The resulting code distinguishes 13 different consonants and one (initial) generic vowel. Philipps also took into account a few irregularities of the English spelling of words or names that are of French, German, Slavic, and some other origins. For all the above

reasons, PDM is generally considered to be an ameliorated version of PM. In a purist approach, this consideration is incorrect. The two algorithms are partly complementary and can be used in different contexts. PDM represents a good alternative to PM for searches of names in a multi-language context, that is, in a context in which PM (and *American Soundex* too) was not really usable. On the other hand, PDM is less appropriate for general English lexicon than PM. Here the only (but major) disadvantage of PDM in comparison to PM consists in the limitation of the resulting code to four characters returning to the limitation present in the *American Soundex*. The most appropriate tool for processing English words would be PM amended to withdraw its incoherent elements but without any truncation of long words.

The name of the new algorithm includes the word "Double" because, when using PDM, a name can receive two different codes, "primary" and "secondary". For example, encoding the name Smith yields a primary code of SM0 and a secondary code of XMT, while the name Schmidt yields a primary code of XMT and a secondary code of SMT. One can observe that both have the code XMT in common. Consequently, a name searching application can find a match between these names. To take into account the possibility of non-English origins, the algorithm distinguishes several dozens of contexts.

PM, with a small number of major general amendments mentioned above, but without the rules for foreign languages, is clearly better adapted for searching English names than other algorithms mentioned until now (including PDM). For searching names that can be of various origins, it makes sense to consider PDM and DMS as competing algorithms. Both of them ignore internal and final vowels. In the initial position, DMS is more sensitive: it can uses two different codes (0 or 1), while PDM turn all of them into "A". According to this feature, PDM has a bigger number of "false positives" than DMS (and is, therefore, worse than it). Compare Yar and Ore, for which DMS has different codes (190000 and 090000, respectively), while PDM uses the same code AR. For consonants, PDM has 13 signs (letters) in contrast to 8 (figures) present in DMS. This corresponds to the following distinctions present in PDM but absent from DMS: M versus N (both coded 6 in DMS), P versus F (7 in DMS), K versus H (5 in DMS), 0 (English "th") versus T (3 in DMS), as well as X (English "ch" and "sh") versus S (4 in DMS). Due to the introduction of these distinctions, certain names that match according to DMS do not match for PDM. In numerous cases, this approach eliminates "false positives". Compare: (1) Moon (code MN) and Nine (code NN), while the DMS code is 660000 for both of them; (2) Pine (code PN) and Fine (code FN), with the common code 760000 in DMS. However, in a few cases it also introduces new "false negatives" that are not present in DMS. For example Grinberg (KRNP) and Grimberg (KRMP) both have the DMS code of 596795. In many languages, these names are pronounced quite close to each other, while in Spanish the pronunciation is identical. Sas (SS) and Schasch (XX) both have the

DMS code 440000. Actually, the former name can represent the Hungarian spelling of the latter (that is spelled according to the rules of the German orthography). For non-English names, the processing of the letter "w" is more appropriate in DMS than in PDM. Indeed, in German and Polish this letter has the same sound as English "v". Consequently, it is more appropriate to assign to both "v" and "w" the same code (at least, as an alternate coding) as it is done within DMS (code 7). Without such a rule, we can obtain such inappropriate situation as Dvorak (TFRK) and Dworak (TRK) that do not match in PDM and are, therefore, "false negatives". Yet, both have the DMS code 379500. The limitation of PDM to four characters also puts DMS (with its six digits) into a favorable position. Indeed, the bigger is the number of characters coded, the smaller is the number of "false positives" an algorithm generates. Another advantage of DMS consists in the possibility of having more than two different codes for the same names, while for PDM two represents the maximum number of allowed different codes. Generally speaking, the bigger is the number of possible alternate codes in an algorithm, the smaller is its number of "false negatives".

Taking all the above arguments into account, one can say that globally speaking the *Daitch-Mokotoff Soundex* is a better tool to search non-English names than *Double Metaphone*.

e. Beider-Morse Phonetic Matching

Beider-Morse Phonetic Matching (BMPM) was designed by Alexander Beider (France) and Stephen P. Morse (U.S.). Beider dealt with the linguistic part of this method and Morse with the computer aspects and all technical issues. Major algorithmic decisions are due to common efforts of both authors.[2]

As other tools mentioned above, BMPM is designed to be used as a programming tool. Similarly to DMS and PDM, a user would enter a name on a form, that name would be transmitted to a server running the phonetic engine that would generate the BMPM codes, and these codes would then be compared to the BMPM codes that were previously generated for all the names in a specific database. The steps of this comparison are described in the following sections. Major characteristics of BMPM are:

---

[2] The initial work on this algorithm was based on the article by Beider "Some Issues in Ashkenazic Name Searches" (*Avotaynu: The International Review of Jewish Genealogy*. Vol. XXIII, Number 1, Spring 2007, pp.3–13) in which major limitations of DMS were discussed and a number of amendments were suggested. The main principles of BMPM are exposed in the paper "Beider-Morse Phonetic Matching: An Alternative to Soundex with Fewer False Hits". *Avotaynu: The International Review of Jewish Genealogy*. Vol. XXIV, Number 2, summer 2008, pp.12-18, signed by both authors of this algorithm.

- The procedure starts with the step (unknown in other algorithms) during which the program is trying to identify the language in which the name is spelled.
- The length of the resulting code has no limitations (only *Metaphone* has the same feature)
- The number of alternate codes can be significantly bigger than in DMS (not to speak about other algorithms)
- Numerous coding rules are formulated for specific contexts: beginning or end of a name, following and/or preceding some particular letters. For this feature, some similarities can be found in PDM, while DMS allows only for two particular contexts: beginning of a name and the position before any vowel.

Step 1. Identifying the Language

The spelling of a name can include some letters or letter combinations that allow identifying the language unambiguously. Some examples are:

- The presence of some special characters can be decisive: “ś”, “ľ”, and “ż” are Polish, “ğ” is Turkish, “š”,“č”, “ň”, and “ř” are Czech, “ţ” is Romanian, “ã” is Portuguese, “ñ” is Spanish
- “tsch”, final “mann” and “witz” are specifically German
- final and initial “cs” and “zs” are Hungarian
- “cz”, “cy”, initial “rz”, final “cki” are typically Polish.

More often, several languages can be responsible for a letter or a letter combination. For example, “ö” can be German, Hungarian, or Turkish, final “ck” can be either German or English, “sz” can be either Polish or Hungarian. Sometimes it can be easier to name the language or the languages in which certain letters can never occur. For example, “y” and “k” are not present in Romanian, “v” cannot be Polish, the string “kie” can be neither French, nor Spanish.

BMPM includes several hundreds of rules for determining the language. Their processing yields one or several languages that could, in principle, be responsible for the spelling entered by the user. These languages represent a subset of the initial list for which a particular version of BMPM is set up. Currently, the version called “Generic” is designed for working with the following languages: English, German, Dutch, Czech, Polish, Russian (in Cyrillic characters or transliterated in English), Hungarian, Romanian, French, Spanish, Catalan, Portuguese, Italian, Turkish, Greek (in Greek characters or Latin transliteration), Hebrew, and Arabic. One option of the BMPM engine allows for making a subset of this list and process name(s) according to this subset only. For example, if only Polish and German languages are left in the above list, the analysis will be applicable for processing a large majority of names known in Poland. Two

particular subsets exist as predefined versions. The first is called “Ashkenazic” and covers English, German, Polish, Russian, Hungarian, Romanian, French, the Argentinean version of Spanish, and Hebrew. The second version, “Sephardic”, covers French, Spanish, Catalan, Portuguese, Italian, and Hebrew. The existence of these two subsets is not fortuitous. It is directly due to the involvement of one of the designers of this algorithm in Jewish onomastics and another in Jewish genealogy. These two versions were actually designed before the “Generic” one. However, exactly as DMS, the entire system has nothing specifically Jewish. For example, the “Ashkenazic” version is perfectly applicable for names of Gentiles from the Eastern and Central Europe, while the “Sephardic” version fits to the onomastics of Christians from Romance West European countries. The “Generic” version is neutral: it is applicable to names belonging to all cultures mentioned above.

The option of making a subset from the initial global list of languages is of particular interest when all names present in a database to be coded are known to be written in one or several possible particular languages. In this case, numerous rules present in BMPM for other languages become not applicable. This approach allows diminishing dramatically the number of “false positives”.

Step 2: Calculating the Exact Phonetic Value

In a number of languages, forms of surnames used by women are different from those used by men. For example, it would be Jan Suchy but Maria Sucha. And the wife of Mr. Novikov would be called Mrs. Novikova. This occurs, among others, in Slavic tongues (including Polish and Russian used in the above examples), Lithuanian and Latvian. Since the name under analysis can, in principle, be feminine, this step starts with replacing feminine endings with the masculine ones. After the name has been “defeminized”, the phonetic engine tries to identify the *exact phonetic value* of all letters of the name, and transcribe them into a conventional phonetic alphabet. Since in principle the number of different sounds is huge, it was decided to restrict the phonetic alphabet used in BMPM to those sounds that are shared by several languages covered by the algorithm. For example, the difference between Polish “y” and “i” was deliberately ignored because there is no way to express it in non-Slavic languages. Also ignored was the difference between two sounds expressed in German by “ch”, those present in words “ach” and “ich”. For the same reasons, numerous vowels found in French and English (but absent from Latin and therefore having no characters to express them unambiguously) do not figure in the phonetic alphabet of BMPM, but instead were replaced with closest equivalents found in Germanic and Slavic languages. The retained list appears in the table below.

	Example		Example
a	Like in *part*	b	Like in *boy*
d	Like in *dog*	e	Like in *set*
f	Like in *flag*	g	Like in *dog*
h	Like in *hand*	i	Like in *Nice* (the city), or *ee* as in *fleet*
j	Like *y* in *yes,* equivalent to German *j*	k	Like in *king*
l	Like in *lamp*	m	Like in *man*
n	Like in *neck*	o	Like in *port*
p	Like in *pot*	r	Like in *ring*
s	Like in *star*	t	Like in *tent*
u	Like in *flu*, or *oo* as in *good*	v	Like in *vase*
w	Like in *wax*	x	Like *ch* in *loch*; equivalent to German *ch*
z	Like in *zoo*	s	Like *s* in *sure*, or *sh* in *shop*
q	Like *ü* in German *Müller*	z	Like *z* in *azure*; equivalent to French *j*
y	Like *ö* in German *Löwe*		

Generally, the conventionally chosen signs for sounds are the same as those used by *International Phonetic Alphabet* (IPA). The only exceptions are S, Z, Q, and Y whose IPA's equivalents are ʃ, ʒ, y, and ø, respectively. The BMPM choices were dictated by the wish to be limited to standard Latin characters present on any keyboard using the Roman alphabet. As it can be seen from the table above, the total list covers 27 different sounds. This number is significantly bigger than the distinctions done in other methods.

The transcription of the name into the characters found in the above table (a better term for it would be *mapping*) depends of the result of Step 1. Either Step 1 determined a unique language, or it determined a set of possible languages. If only one possible language was left after Step 1 the phonetic engine transcribes the spelling to the phonetic alphabet using rules specific to that language. In BMPM, every language possesses its own set of rules for this mapping (less than 40 for Romanian, about 80 for German and more than 130 for Polish). For example, if the language is German, then some of the rules are: “sch” maps into the “S”; “s” at the start of the word and “s” present between two vowels becomes “z”, “w” becomes “v”. For certain languages, some letters can be read in several ways. In these cases, the phonetic engine assigns them two (or more) elements from the phonetic alphabet. For example, Polish “a” normally corresponds to phonetic “a”. In some cases, however, this letter can result from Polish “ą” in which the diacritic sign (comma under the “a”) was lost. In this example, the phonetic value would be either “om” (before “b” or “p”) or “on” (before other consonants). Generally speaking, one can say that the language-specific rules of BMPM are in many aspects similar to those applied by the *Metaphone*-like algorithms for the corresponding languages with one significant change: BMPM allows for multiple coding.

If Step 1 resulted in more than one possible language, the phonetic engine processes the name using "generic" rules. To adequately support the languages of the current version of BMPM, about 300 generic rules are present. There are two types of such generic rules – ones that are language independent and ones that apply only to certain languages. An example of a language-independent generic rule is the rule for final "tz" – it can be pronounced only as English "ts". Such language-independent generic rules are applied regardless of which languages are present in the output of Step 1. Other generic rules might be applicable, however, to specific languages only. The output of Step 1 would determine whether or not these language-specific generic rules would be applied. For example, "ch" can be mapped (using the signs of the conventional phonetic alphabet) to "x" in Polish or German, "S" in French, or the diphthong "tS" in English or Spanish. If during Step 1 we learn that English, Spanish, and French are not possible, only the Polish/German language-specific rule will be applied, causing the "ch" to be mapped to "x".

Once the name is processed by either the generic rules or the language-specific rules, the phonetic engine applies to the resulting string of phonetic characters a series of phonetic rules that are common to many languages. As an example, consider the rule known in linguistic literature as *final devoicing*. It applies to many European languages, such as German, several Slavic tongues including Russian and Polish, and some dialects of Yiddish. Final devoicing states that at the end of the word the voiced consonants are pronounced as their unvoiced counterparts – that is, "b" is pronounced as "p"; "v" as "f"; "d" as "t" etc. The phonetic engine takes this peculiarity of speech into account and keeps in the final position only the unvoiced consonants. For example, Maslov gives Maslof. Another rule, also applied by the phonetic engine, is that of *regressive assimilation*, whereby a consonant acquires characteristics of the consonant that follows it.

At the end of Step 2 the initial surname is transformed by the phonetic engine into one or several strings of characters that are conventionally called the *exact phonetic value*.

Step 3: Calculating the Approximate Phonetic Value

After the rules mentioned in Step 2 are applied, there is an option for the phonetic engine to apply a series of additional rules. These rules take into account the fact that some sounds can be interchangeable in some specific contexts that are more complex than the contexts considered in Step 2 (beginning/end of word or before/after a letter). For example, the Russian unstressed "o" is pronounced as "a". As a result, Mostov and Mastov sound alike because the first syllable is unstressed. On the other hand, there is no interchangeability in the stressed position: Panin and Ponin sound differently. Since in many languages automatic determination of the stress position is impossible, "a" and "o" are considered to

be *approximately* interchangeable. Other rules allow for phonetic proximity of a pair of sounds resulting in their partial confusion. For example, “n” before “b” sounds close to “m”. Grinberg becomes *approximately* equivalent to Grimberg for many languages. Since in Spanish this equivalence is total, in Argentina Grinberg and Grimberg are *exactly* equivalent. Just as in Step 2, the *approximate* rules applied here can be either language-specific or generic, depending of the results of Step 1.

At the end of this step the initial surname is transformed by the phonetic engine into one or several strings of characters called the *approximate phonetic value*. Here the approach has some structural similarity with the matching obtained in PDM between two names for which one of the pair of codes (“primary” or “secondary”) are identical. However, this analogy should not be overestimated: the number of BMPM codes for the same name can be very large, while in PDM it is limited to only two.

Step 4: Calculating the Hebrew Phonetic Value

All previous steps can in principle be applied to various cultures. This optional step, on the other hand, is specifically Jewish. The main aim of this step consists in taking into account the fact that the initial name as written in Latin or Cyrillic characters can be the result of a transliteration from Hebrew. Since some vowels do not appear in Hebrew spelling and the sounds of other vowels and certain consonants are ambiguous, a transliteration of the same name from Hebrew to Latin characters made by different people can yield different results. For example, פסטר can yield Fester, Faster, Paster, Pastar, Pester, Fasater, Psater etc., בין can correspond to surnames that were spelled in German as Bien, Bin, Bühn, Bün and Bein, פרימס can be Frimes or Primas. This step is designed to fix the issues related to the transliteration from Hebrew. To accomplish this, the phonetic engine takes the results of Step 2 and applies a series of additional rules that allow for the ambiguity of certain sounds when dealing with the Hebrew spelling.

At the end of this step, the initial surname is transformed by the phonetic engine into one or several conventional strings of phonetic characters called the *Hebrew phonetic value*. Surnames whose Hebrew spelling is the same have the identical *Hebrew phonetic value*. Some examples are Bader and Beder; Brak, Berak and Barak; Bober, Buber and Bubar; Brauner, Bronner, and Bruner; Mandel and Mendel; Thaler and Teller; Zipper and Ziffer.

Step 5: Searching for Matches

The matching of individual name to names present in specific electronic lists proceeds in the following way:

If the one of the *exact phonetic values* of this name and a name from the list are identical, the match is called *exact*. These two names are phonetically equivalent.

If one of the *approximate phonetic values* of this name and a name from the list are identical, the match is *approximate*. These two names can be (or not be) phonetically equivalent.

If one of the *Hebrew phonetic values* of this name and a name from the list are identical, we have a *Hebrew* match. These two names can be phonetically equivalent only if at least one of them was originally spelled in Hebrew. If the user knows that neither of them was spelled in Hebrew or results from the transliteration from Hebrew, the *Hebrew match* is of no importance and can be simply ignored.

Comparison between BMPM and DMS

A context in which DMS seems to be more appropriate than BMPM is when the original form of the name (which is the form as it appears in the list to be searched through) is not known and all that is known is the Anglicized form of the name used today. Here are some examples:

(1) Various names starting with Silver – such as Silverberg and Silverstein—in which Silver came from the original German Silber. The change is not just phonetic, it is partly semantic: the German word for “silver” was replaced with its English equivalent. For DMS, both Silver and Silber have the same code 487900. For BMPM, these two names do not match.

(2) Names having English “stone” instead of German “stein” such as Rotstone or Redstone instead of Rotstein. The DMS value for all three of them is the same (943600), though the pronunciation of these words is significantly different and for this reason BMPM will not find any match here. (The situation is different in the case of “green” for "grün" and “field” for “feld”: they do match in BMPM too because here the match is phonetic as well.)

(3) Tartatsky/Tartatzky/Tartacki becoming Tartasky in US (DMS has 393450 for all these names). Again, phonetically speaking, Tartatsky and Tartasky are not equivalent and for that reason BMPM does not consider them as matches. In this example as well as in the two others given above, DMS can find some Anglicized fits because the adaptation of sounds from one language to another often changes them to sounds that are different, but still close (and consequently their DMS can be identical), while both English and German belong to the same linguistic group of Germanic languages. Under these conditions, semantic adaptations of surnames can produce forms that are close phonetically too.

(4) Konstantinovsky shortens his name to Constantine. Here the DMS code does not change (564363) just because it is limited to six digits.

Outside of this very limited context of certain Anglicized forms (that are “false negatives” for BMPM but not for DMS), BMPM has numerous advantages over

DMS. Taking into account the number of rules present in each method (a few dozens in DMS in contrast to several thousands in BMPM) this is not really a surprise.

On the one hand, numerous “false negatives” for DMS are processed correctly by BMPM. Several examples dealing with Ashkenazic surnames appear below making references to peculiarities of BMPM:

(1) Triphthongs are *approximately* equivalent to diphthongs: Altmayr matches to Altmayer, Heym to Heyem, Kajm to Kaiem.

(2) Forms with “h” between vowels or at the beginning of the word are *approximately* equivalent to those in which “h” was lost: Johanes and Joanes, Halperin and Alperin.

(3) The letter combinations “inm” and “jnm” are *approximately* equivalent to “im” and “jm”: Weinman(n) and Weiman(n), Fajnman and Fajman.

(4) “sc” before a vowel is not equivalent to “s” or “sch”, it can be *exactly* equivalent to “sk”: Boscowitz and Boskowitz, Muscat and Muskat.

(5) When one sound expressed in the BMPM conventional phonetic alphabet by the signs “S” (English “sh”), “Z” (French “j”), “s” and “z” is followed by another sound from the same group, it can be dropped due to the phenomenon of the *regressive assimilation* mentioned above. As a result, the following names match *exactly*: Hirschstein and Hirstein, Ovruchsky and Ovrutsky.

(6) The sound “d” disappears if it is followed by the sound “t” or a diphthong that starts with “t” (such as that expressed by “ch” as in English “check”). Consequently, names in the following pairs match *exactly*: Gladtke and Glatcke, Goldzweig and Golzweig, Kurlandchik and Kurlanchik.

(7) Several transliterations into English of Cyrillic vowels followed by “e” are *exactly* equivalent: “ae”, “aye”, “aie” and “aje” (all for Cyrillic “ae”); “oe”, “oye”, “oie” and “oje” (all for Cyrillic "oe") etc. Examples: Faer, Fajer, Faier and Fayer (Cyrillic Фаер), Meer, Mejer, Meier and Meyer (Cyrillic Меер). In DMS, the forms with “ae” and “oe” do not match to “aye/aie/aje” and “oye/oie/oje”, respectively.

(8) Initial "Rh" is *exactly* equivalent to "R": Rhau and Rau, Rhein and Rain.

Evidently, some of these drawbacks of DMS can be easily eliminated by introducing new rules (for example, the last one). For others, the general logic of the DMS prevents such pairs from matching.

On the other hand (and even more important), the number of “false positives” in DMS is dramatically larger than in BMPM. This is mainly due to the following major characteristics: (a) truncation of original names by DMS due to the code length limitation; (b) DMS ignores all internal vowels; (c) DMS distinguishes a significantly smaller number of sounds, (d) BMPM analyses the contexts in which a particular letter is present in a significantly more nuanced way.

One example based on calculations made by Steve Morse can illustrate the situation. Searching through the Ellis Island Passenger Lists data base for

Eisenhower, DMS finds 27 differently spelled names. 21 of them are “false positives” including such forms as Asimakofrulos, Assinacoperde, Eichenhofer, and Ochenhofer. BMPM finds 26 matching names of which only 2 matches—Allinayer and Gasnier (both *approximate*)—are “false positives”. If one takes advantage of the flexibility of the implementation set up of BMPM and withdraw Spanish from the list of potential languages, Allinayer will immediately disappear. It takes into account the possibility of “ll” in Argentina to correspond to the sound expressed in French by the letter “j”. Gasnier disappears if one excludes Russian (in its transliterated version) from the list of possible source languages. Indeed, this “false positive” is generated because the Russian letter “Г” (transliterated as “G”) can, in theory, stay for the original sound /h/ that can in turn be not pronounced in some contexts. The number of “false negatives” for Eisenhower is zero for BMPM and 19 for DMS.

If for searches made by users for one particular name, one can overcome some of the aforementioned difficulties of using DMS (generally, the user can at least easily tell “false positives” from the correct matches), for automatic searches made by applications trying to match names from several large lists only BMPM is applicable.

For any implementation of BMPM, two important choices should be done. Firstly, one needs to choose the exact subset of languages. If all of these languages are present in one of the predefined versions, the “Ashkenazic” or the “Sephardic” ones, it is better to select the subset in question from one of them rather than from the “Generic” version. Such approach allows diminishing the number of “false positives”. Secondly, one needs to choose between the “exact” and/or “approximate” matches. The “exact” match has a smaller number of “false positives”. Yet, the “approximate” match has a smaller number of “false negatives”.

III. Fuzzy String Searching

a. Problem

The phonetic ambiguities of some letters discussed in the previous chapter do not cover all identification issues. In many cases, when dealing with printed or online sources, one faces secondary (or even tertiary) spellings that result from misinterpretation of the primary source. These mistakes can be classified into two broad categories. The first category encompasses all typographic errors. When a researcher prepares large extracts from a source, he/she may unintentionally introduce some incorrect characters. Statistically speaking, one most often finds inversions of the order of two letters, omissions of a character, introductions of an extra character, or substitutions between characters. In modern times, the last error is often related to the typing of a character that is next to the right one on the keyboard. The second category

covers misinterpretations made by persons who worked with handwritten materials. Such errors may have been present already in the first (printed or electronic) source made public. The presence of such items demonstrates without ambiguity that the persons responsible for the printing based their publication on original handwritten lists in which some characters were not written clearly. Every alphabet possesses its own set of letters that look similar and therefore are easy to be taken on for another. For Roman alphabet, it is the case of: *n, u* and *v*; *e* and *c*; *f, t* and *l*; *F, T, L,* and *J*; *h* and *k*; *v* and *r*; *r, z,* and *s*; *r* and *i*; *m, rn, w*, *in,* and *ni*; *W* and *M*; *a* and *o*; *a* and *u*; *B, R,* and *K*; *g* and *y*. In Russian handwriting confusion can be easily made between:

• Low case letters: *а* (a) and *о* (o); *г* (g) and *ч* (ch); *ш* (sh) and *м* (m); *ш* (sh) and *т* (t); *н* (n), *и* (i), and *й* (j); *ф* (f) and *ер* (er); *ц* (ts) and *у* (u); *ю* (yu) and *іо* (io; used only before 1918, modern *ё*); ъ (*hard sign*), ь (*soft sign*) and ѣ (e; this letter, called *yat'* in Russian, was abolished in 1918); *к* (k) and *х* (kh)

• Both low and upper case letters: *ш* (sh) and *щ* (shch); *м* (m) and *т* (t); *б* (b) and *в* (v); *к* (k) and *н* (n)

• Only upper case letters: Г (G) and Т (T).

Examples of the same kind can be suggested for Hebrew, Arabic, and other alphabets. These rules can be easily taken into account by software programs that allow searching through online databases based on original documents written in the corresponding alphabets.

To illustrate ambiguities of the Latin spelling one can take as an example one of the most important available databases of genealogical interest, the Ellis Island Passenger Lists. This electronic source is, unfortunately, particularly affected by misinterpretations. The original Roman alphabet records were done by American clerks who often had terrible handwriting. To complicate matters, the interpretation of these records, dealing with people of various origins coming to U.S., was done by volunteer members of the LDS (Mormon) Church. In many cases, it is clear that these individuals were totally unfamiliar with the world of Germanic and Slavic names. As a result, one finds thousands of incorrect forms. In addition to the general rules of letter substitution given above, several other general rules simplifying this automatic search can be formulated. During the period that Ellis Island was in use (1892-1924), names of immigrants from Eastern Europe were never spelled according to the rules of English orthography. The spelling was mostly German. Beginning in the 1920s, Polish spelling starts to be common too. As a result, typical English forms mainly correspond to misinterpretations. Here are two examples: the surnames appearing as Shipski and Shongin in the database are actually spelled Slupski and Strongin on the

original passenger lists.[3] The letter "v" is rare in German and absent from Polish. Consequently if it is present in a surname, then we likely have a misinterpretation. For example, Disvak, Gilvin, Vikel and Vowagruaska actually are spelled Diwak, Gildin, Nickel and Nowogrudska, respectively.

The above rules unfortunately, are far from being exhaustive. In numerous cases, the misinterpretation is particularly difficult to detect automatically without looking at the picture of the actual record. As a result of errors, some names become almost totally unrecognizable. The list below dealing with surnames of Jews from Grodno province illustrates this statement. The first forms are those from the Ellis Island Data Base. The second forms are those that expert users can read themselves by looking directly to the pictures of the original documents available on the same site:

Agnoski < Agurski, Atowski < Stawski, Belsusky < Belansky, Besches < Bosiches, Bewanowski < Baranowski, Bivinstein < Burnstein, Buleuski < Butenski, Cheten < Cholow, Dazrowsky < Lazarowsky, Efwimski < Efroimski, Findzel < Finckel, Gerlin < Perlin, Gumoritz < Gumnitz, Hominski < Slonimski, Hransowicz < Abramowicz, Hameschin < Manuschkin, Hezkerrazy < Aszkenazy, Kalpson < Halpern, Kenstein < Orenstein, Kinlang < Rimland, Kirstof < Kustof, Knochncz < Kuschner, Koffenauer < Hoffmann, Kowcikin < Kiweikin, Kranek < Krawetz, Kresner < Kremer, Lekvoiski < Zakroiski, Rukvi < Rubin, Sambaum < Barnbaum, Schipin < Schifrin, Schkowitz < Selikowitz, Stukow < Stukar, Swochitzky < Suschitzky, Tomerkutz < Pomerantz, Tun < Torn, Turber < Farber, Walforn < Wolfow, Wowezyla < Wowczyk, Wulles < Müller, Zabendive < Babendure, Zakfus < Zalzfas, and Zumgas < Grüngas.

As the discussion demonstrates, to facilitate the solution of problems related to the possible misinterpretation of handwritten characters, for every element of online searchable databases it is appropriate to identify whether or not the original source was handwritten. If it was, the alphabet used in the original source should be explicitly identified: Roman, Cyrillic, Hebrew, Arabic etc. Awareness about a specific language (English, German, Polish, Russian etc.) used in the original spelling would be even more helpful. This information is of paramount importance when analyzing possible graphic misinterpretations.

---

[3] English spelling is relevant only for passengers coming to Northern America from England. Some of them bear surnames that clearly originated in Eastern Europe. These families likely correspond to those making already a second migration. The first one was that from the Russian Empire to England. For this reason, these individuals often have Anglicized given names.

b. Computer Solutions

To address the issues discussed above, several techniques exist in computer sciences. Their general name is *approximate string matching*. Often, the same methods are colloquially referred to as *fuzzy string searching*. They all have the same aim in common: finding strings of characters that match between them approximately rather than exactly. To turn one string into another, different from it, several operations can be performed:

- Insertion: Renaut → Renau***l***t
- Deletion: Peugeo***l***t → Peugeot
- Substitution: ***S***itroen → ***C***itroen
- Transposition (of two adjacent characters): P***ro***sche → P***or***sche.

The closeness of a match between two strings (that is, a distance between them) is measured in terms of the number of operations necessary to obtain the exact fit. The smaller is the distance, the better is the match. The most commonly used measures are the following ones:

1. *Levenshtein distance* counts for a total number of insertions, deletions, or substitutions needed to change one string to another. For example, in the above examples, the distance is 1. Yet, the distance between *Rinaut* and *Renault* is 2 because two steps are needed: (1) insertion of *l*, (2) substitution of *i* by *e*.
2. *Damerau-Levenshtein distance* represents an evolution of the previous measure: it also authorizes transpositions. For example, the distance between *Prosch* and *Porsche* is 2: (1) transposition *ro* → *or*, (2) insertion of the final *e*.
3. *Hamming distance* counts the number of substitutions needed between two strings of equal length. For example, the distance between Peu***j***eo***d*** and Peu***g***eo***t*** is 2.
4. Length of the *Longest common subsequence* corresponds to the number of insertions or deletions.

The second of these approaches is the most appropriate method for detecting and correcting misspellings and for this reason it is often used in spell-checking applications. Evidently, it is also applicable for processing typographic errors and similar misspellings of names. However, it is not adapted for dealing with misinterpretations of the originally handwritten materials. Here, it would be more appropriate to use a generalized approach based on the *Levenshtein distance* because the issue in question has nothing to do with transpositions. The generalization in question consists in assigning different weights to various operations and letters that participate in them. If we conventionally designate by the Greek symbol λ the absence of a letter, all three operations taken into account when calculating the *Levenshtein distance* can be reduced to only one: substitution. An insertion of a letter can be processed as a substitution of λ by this

letter. A deletion of a letter can be processed as a substitution of this letter by λ. Using these designations, one can set up weights for different operations:

- w(a, b) = weight of the substitution of “a” by “b”
- w(λ, b) = weight of the insertion of “b”
- w(a, λ) = weight of the deletion of “a”.

Two strings are considered matching if and only if the distance between them is smaller than one predefined threshold. Several computer programs designed for measuring the *Levenshtein distance*—including the *Wagner-Fischer algorithm*—allow for such kind of flexible approach.

In our particular context, it is logical to assign smaller weights for insertions and deletions of letters that are can consist in only one stroke (*l, j, c, r, I*) and significantly bigger weights for graphically elaborate letters such as *a, z, w, m*, *g, q, x,* or *R*. For certain letters from the last list, the weight can be defined sufficiently big in order to exclude the possibility of their insertion or deletion as totally implausible. According to the list of letters easily interchangeable given in the previous section, a number of particular substitutions should receive small weights. For example, it is the case for the values of w(n, u), w(u, v), w(e, c), w(r, i), and w(g, q). On the other hand, the values of certain other substitutions can be much bigger and even larger than the threshold defined for considering names as matching: w(q, m), w(g, z) etc. Such approach allows taking into account numerous misinterpretations of the original handwritten text. However, it does not cover substitutions between groups of adjacent letters such as “m” taken for “rn” or “in”. In these examples, the classical *Levenshtein distance* between the strings in question is 2. Yet, it is intuitively clear that here the distance should be 1. As a result, supplementary generalization of the existing programming algorithms is needed to take into account this kind of logic.

IV. Synonyms

The methods described in the above chapters are applicable for any kind of names. The only specificity of surnames in comparison to given names or toponyms mentioned till now consisted in the “defeminizing” procedure present in *Beider-Morse Phonetic Matching*. Yet, given names and toponyms can possess additional important features that are generally irrelevant in the domain of surnames. Indeed, someone called Robert can also be referred to by one of hypocoristic forms of the same name such as Bob, Bobby, Rob, Robby, Bert etc. This English tradition finds its equivalents in many other cultures: compare Spanish José and its derived forms such as Pepe, Chepe, Pepito, Chepito, Pito, and Pepín, or Polish Jan and its hypocoristic forms such as Janek, Janko, Jasiek, Jaś, and Jaśko. To obtain a match between two given names that both correspond to the same base form, methods described in the previous chapters are of little help.

An application aimed to provide this kind of matching should include information of a totally different kind. Firstly, it should contain the explicit list of all base forms for which derived variants can, in principle, be found. This list can serve as a kind of "common denominator" for matching given names between them. Secondly, it should possess data allowing identifying base forms from their derived variants. Here one approach could consist in the introduction of certain general morphological rules describing the way hypocoristic forms are obtained such as truncation, addition of particular suffixes (whose exhaustive list should be in this case entered into the application), and possible stem changes concomitant to one of the two above processes. However, this approach is unlikely to cover all cases (compare Pepe that has no letter in common with José). Moreover, for every culture the list of possible hypocoristic forms is known. For these reasons, it can be much more appropriate to simply include into the application a comprehensive list of derived forms. These names can be considered to be "synonyms" for the base name as well as between them. When dealing with given names of immigrants in various sources of genealogical interest, one can need to enter into the application an additional set of "synonyms" that represents correspondences between base forms belonging to the original country and those of the new country. For example, in U.S. a Polish immigrant called Jan at his birth can "translate" his name into John, while a Spaniard named José can become or be referred to as Joseph. A third set of correspondences is specific to Jewish men. In different contexts directly related to Judaism (call for the Torah reading in a synagogue, tombstone inscriptions, texts written in Hebrew) a religious Jew is usually referred to under a name from the category of *shemot ha-qodesh.* These "sacred names" are either post-biblical of Hebrew or Aramaic origin, or biblical. In the vernacular life, the same individual can be known under a totally different name, one of the *kinnuim* typical for the community in question. Correspondences between *shemot ha-qodesh* and *kinnuim* are not formally fixed, but some of them are standard to the Jewish tradition for many centuries. Yiddish Leyb mainly corresponds either to Arie or Judah, Wolf is an equivalent to both Zev and Benjamin, both Issachar and Dov are *shemot ha-qodesh* for Ber.

An introduction of the explicit list of "synonyms" is also appropriate for matching certain toponyms. For example, this is needed to recognize the equivalence between the Netherlands and Holland. Names of the same areas or towns in different languages whose equivalence cannot be found by the phonetic matching should also be treated as synonyms. Examples: U.S.A. and Etats-Unis, Aix-la-Chapelle and Aachen, Munich and München, London and Londres, Gothenburg and Göteborg. Milan(o) and Mailand. In the genealogical context, when dealing with toponyms one can also face another problem unknown for surnames or given names. Two different sources can mention two toponyms of which one covers a geographic area that represents only a part of the region covered by another toponym. For example: a small town and the center of the

same district, a town and a province/state/region, a province and a country. A "clever" application should be able to match such toponyms. For realizing appropriate matching in this context, hierarchical lists of toponyms should be introduced within the application.

For surnames, the introduction of synonyms is significantly less useful. Its only (rather limited) application can deal with certain common changes of names of immigrants in which the original name is translated into the language of the new country. This way, German Zimmerman, Adler, and Schwarz can become in English-speaking countries Carpenter, Eagle, and Black, respectively, while a Hungarian called Szabó can translate his surname to Taylor.

V. Conclusion

In certain computer programs designed for name searches in large online data bases, the methods discussed in this paper are considered to be alternative. A user can run a search based on some kind of phonetic matching, or use one of the methods of fuzzy string searching, or ask the computer to make a research within the list of available synonyms. For individual searches, this simple approach is not a problem. The user can make all these searches for the name of his/her interest one by one and judge visually about the pertinence of the displayed matches. However, even in these cases some matches will be lost. For matching between large lists, this approach is inappropriate. To sort a large amount of information in the most efficient way, the procedure of determining the best fits can be based on an approach that combines various methods. Indeed, the three groups of algorithms discussed in this paper are complementary to each other. Consider an example. Someone is searching in genealogical data bases for his ancestor known in U.S. as Margaret Covalsky and born in the Lublin area. Methods of phonetic matching can be helpful for realizing that references to the surnames Kowalska, Kovalskaya, Kowalski, Cowalski, Kawalski, Kavalsky, and Ковальская can all be perfectly relevant for this search. Methods of fuzzy string searching used together with phonetic matching can allow identifying potential additional candidates to relatives such as Conalsky, Kuwalska, Kouvalski, Kowaiski, and Кобальская. If the application includes comprehensive lists of synonyms for given names, we can also match references to various forms derived from Małgorzata, the Polish equivalent to Margaret, such as Gosia and Małgosia. Here the simultaneous use of the phonetic matching can be helpful for realizing that Gosza is a variant of Gosia, while Russian Малгоржата and Малгожата are just alternate transliterations of Małgorzata. On the other hand, fuzzy string searching can retrieve such misspellings as Goria, Gona (both for Gosia) and Maigorzata. If hierarchical lists of toponyms are built within the searching programming tool, a user can understand the relevance of the references not only to Lublin, but also to some towns inside Lublin province (for example, Szczebrzeszyn), to "Poland" as a

whole, and even to “Russia” (in the sense of the Russian Empire). Phonetic matching can be helpful to successfully go through many dozens of possible spellings of the toponym of Szczebrzeszyn in English or German transcriptions, some misinterpretations of which can be revealed by the fuzzy string searching methods applied to the data bases constructed using handwritten materials.

Many methods described in this paper are relatively new. Some of them are based on the phonetic analysis and therefore can be considered as belonging to the science of linguistics. Others are based on mathematical algorithms. All of them are primarily designed for computer applications. These modern algorithms using modern software (some of which are still awaiting to be designed) can be helpful for genealogical searches allowing us to discover fascinating information about the time when (some of) our ancestors were living in countries different from those we are living now, were speaking languages having nothing to do with our native idioms and were bearing given names that today are uncommon or even not found at all…

BIOGRAPHY

Alexander Beider was born in Moscow in 1963. Since 1990, he lives with his family in Paris, France. PhD in applied mathematics (1989, Moscow Physico-Technical Institute) and Jewish studies (2000, Paris, Sorbonne), he is the author of a series of reference books on Jewish onomastics and papers about the origins of Yiddish.