Fuzzy Matching in SAS: A Tutorial for Data Integration

Authored by

Mohammed looti

November 14, 2025

The Necessity of Fuzzy Matching in Modern Data Integration

In the sphere of modern data integration and comprehensive analytical processes, practitioners frequently encounter a pervasive challenge: merging or comparing disparate datasets where the primary identifying keys, such as customer names, addresses, or specialized product codes, fail to align perfectly. This discrepancy often stems from unavoidable real-world issues, including simple manual data entry errors, subtle variations in spelling, or the deployment of inconsistent naming conventions across different source systems. When the crucial requirement of exact string matches cannot be met, an exceptionally powerful statistical technique known as fuzzy matching becomes absolutely indispensable for successfully identifying and linking records that are similar, yet not strictly identical.

Fuzzy matching, sometimes referred to as approximate string matching, empowers data professionals to establish connections between records based solely on their measured string similarity rather than demanding strict binary equality. This capability is paramount for essential data quality tasks, including rigorous data deduplication, accurate record linkage across institutional boundaries, and efficiently merging data derived from heterogeneous sources. By utilizing this technique, analysts build a far more comprehensive and accurate foundation for subsequent analysis. Failure to implement effective fuzzy matching protocols means valuable business insights can be irrevocably lost, and the overall quality of the data can be severely compromised by unlinked or duplicated entries.

Within the robust SAS programming environment, the execution of sophisticated fuzzy matching routines is simplified and highly efficient through the application of specialized functions. Two of the most effective tools available for accurately quantifying the similarity between two strings are the **SOUNDEX** and **COMPGED** functions. These functions provide reliable, mathematical mechanisms to assess how closely two strings resemble one another, thereby facilitating the automated "matching" of entries that are similar in sound or spelling, even if they contain errors. The remainder of this guide will explore a detailed, practical example demonstrating the combined power of these two functions within a typical SAS data processing workflow.

Deep Dive into the SOUNDEX Function for Phonetic Matching

The **SOUNDEX** function, native to SAS, is based on a classic phonetic algorithm specifically engineered to encode names and words according to their English pronunciation. The fundamental goal of **SOUNDEX** is to systematically group together names that sound identical when spoken aloud, despite having variations in spelling. This is particularly valuable for resolving potential matches resulting from common typographical errors or alternative spellings that produce distinct string values but share phonetic equivalence.

When the **SOUNDEX** function is applied to a source string, it generates a standardized four-

character code. This code is constructed such that the initial character is always the first letter of the original string, while the subsequent three characters are digits that represent the remaining consonants based on a predefined phonetic grouping system. For instance, common variations of a surname, such as "Smith" and "Smyth," are highly likely to yield the exact same **SOUNDEX** code. This shared encoding serves as a rapid, preliminary mechanism for performing a broad-stroke comparison of strings by focusing on how they sound, rather than their exact character sequence.

While remarkably effective for identifying phonetic similarities, it is important to acknowledge **SOUNDEX**'s inherent limitations. Because its primary focus is auditory similarity, it may not successfully capture all types of string variations. These missed variations include transpositions of letters that do not significantly alter the pronunciation, or minor misspellings that are visually very close but phonetically distinct. Consequently, **SOUNDEX** is most robust and reliable when used in conjunction with other textual string similarity metrics, such as those based on edit distance, to achieve a more comprehensive and highly accurate fuzzy matching solution.

Implementing Textual Comparison with the COMPGED Function

The COMPGED function in SAS complements the phonetic capabilities of **SOUNDEX** by offering a much more granular, character-level approach to measuring string similarity. It achieves this by calculating the edit distance between two strings. The edit distance is a powerful metric that quantifies the minimum number of single-character editing operations--specifically insertions, deletions, or substitutions--required to transform the first string into the second. Fundamentally, a lower resulting edit distance score unequivocally signifies a higher degree of similarity between the two analyzed strings.

The inherent versatility of the COMPGED function lies in its ability to implement various edit distance algorithms, most notably including the widely recognized Levenshtein distance. By default, COMPGED returns a generalized edit distance score, which is a numerical value directly representing the dissimilarity between the inputs. When this function is incorporated into a fuzzy matching routine, this score is evaluated against a carefully selected, predefined threshold: if the calculated score is less than or equal to this threshold, the two strings are confidently flagged as a match.

The most effective methodology for robust record linkage involves the synergistic combination of **SOUNDEX** and COMPGED, establishing a powerful, two-stage matching workflow. **SOUNDEX** is utilized first to quickly and efficiently identify phonetically similar candidates, thereby narrowing the pool of potential matches significantly. Subsequently, COMPGED performs a precise, fine-grained measurement of the textual closeness among these pre-selected candidates. This combined approach allows for both broad phonetic screening and detailed character-level comparison,

making it the preferred method for highly complex fuzzy matching tasks within enterprise SAS installations.

Practical Example: Preparing Disparate Basketball Data

To concretely illustrate the practical application of fuzzy matching in the SAS environment, we will examine a common scenario involving basketball team statistics. Assume we possess two distinct datasets, each containing crucial information about team names, but originating from entirely different source systems or having been manually entered by various personnel. These differences inevitably introduce inconsistencies that prevent a standard, direct, exact-match join operation.

Our initial dataset, designated `data1`, contains the team names alongside the corresponding points scored by various players. We consider this to be our primary or reference source for team information. The following SAS code block demonstrates the precise syntax required to create and populate this initial dataset:

```
/*create first dataset*/
data data1;
input team $ points;
datalines;
Mavs 19
Nets 22
Kings 34
Warriors 19
Magic 32
;
run;
/*view dataset*/
proc print data=data1;
```

Obs	team	points
1	Mavs	19
2	Nets	22
3	Kings	34
4	Warriors	19
5	Magic	32

Next, we define our secondary dataset, `data2`, which holds team names and associated assists. Critically, this second dataset is intentionally populated with common variations and misspellings of the team names found in `data1`. This simulates a highly realistic, messy data scenario where inconsistencies render direct matching infeasible. The subsequent SAS code constructs this challenging secondary dataset:

```
/*create second dataset*/
data data2;
input team $ assists;
datalines;
Netts 8
Majick 7
Keengs 8
Warriors 12
Mavs 4
;
run;
/*view dataset*/
proc print data=data2;
```

Obs	team	assists
1	Netts	8
2	Majick	7
3	Keengs	8
4	Warriors	12
5	Mavs	4

A visual inspection of the two input datasets confirms that while many team names in `data2` are clearly intended to match entries in `data1`, they are not textually identical. For example, "Nets" is rendered as "Netts," "Magic" as "Majick," and "Kings" as "Keengs." This exact situation--where conceptual similarity exists but textual equality fails--is the precise type of data quality challenge that fuzzy matching techniques are uniquely designed to solve, enabling us to accurately combine these records despite their underlying discrepancies.

The SAS Implementation: Combining SOUNDEX and COMPGED

To successfully join `data1` and `data2` based on the detected similarity of team names, we must

deploy a sophisticated SAS syntax that orchestrates the combined strength of both the **SOUNDEX** and **COMPGED** functions. This combined methodology ensures that the matching process considers both phonetic resemblance and the physical textual proximity between the strings, thereby guaranteeing the most robust and accurate matches possible. The underlying process involves an exhaustive iterative comparison, calculating distance metrics, and applying a defined similarity threshold.

The core logic of this fuzzy matching routine is executed within a `DATA` step that systematically iterates through every single record in `data1` and compares it against every single record present in `data2`. For each team name encountered in `data1`, we first generate its standardized **SOUNDEX** code (achieved via the statement `tmp1=soundex(team)`). Subsequently, an internal `DO` loop reads through every record in `data2` using the `SET data2(rename=(team=team2)) POINT=i NOBS=nobs;` statement. The `POINT=i` option grants random access to the observations, and `NOBS=nobs` stores the total count of observations in `data2`, ensuring the loop covers all possible comparison records.

For the team name extracted from `data2` (which is temporarily renamed to `team2`), its corresponding **SOUNDEX** code is also generated (`tmp2=soundex(team2)`). The pivotal step then follows: `dif=compged(tmp1,tmp2);`. Here, the COMPGED function calculates the generalized edit distance, but critically, it does so between the two **SOUNDEX** codes, not the original team names. Comparing the phonetic encodings provides an initial, highly effective layer of phonetic matching, significantly improving the efficiency and accuracy of matching phonetically similar strings that have textual discrepancies.

A final conditional statement, `if dif<=50 then do;`, applies the crucial similarity threshold. If the calculated edit distance (`dif`) between the **SOUNDEX** codes is 50 or below, the records are deemed a successful match. It must be noted that the threshold value of 50 is selected arbitrarily for this tutorial example; in production environments, this value must be rigorously calibrated based on the inherent characteristics of the data and the desired precision level of the matching operation. Upon a successful match, the temporary variables created for the comparison (`i`, `tmp1`, `tmp2`, `dif`) are dropped, and the successfully linked record is written out to the new merged dataset, `data3`.

/*use fuzzy matching to merge datasets based on similar team names*/
data **data3;**
set **data1;**
tmp1=soundex**(team); /*encode team names from data1*/**
do **i=1** to **nobs;**
set **data2(rename=(team=team2)) point=i nobs=nobs;**
tmp2=soundex**(team2); /*encode team names from data2*/**

```
dif=compged(tmp1,tmp2); /*determine similarity between team names*/
if dif<=50 then do;
drop i tmp1 tmp2 dif; /*drop unnecessary variables*/
output;
end;
end;
run;

/*view resulting dataset*/
proc print data=data3;
```

Interpreting the Results of the Fuzzy Match

Upon successful execution of the SAS code dedicated to fuzzy matching, the resulting output dataset, `data3`, provides concrete evidence of the efficiency and linking power achieved by combining the **SOUNDEX** and COMPGED functions. This newly merged dataset has successfully consolidated records from the initial `data1` and `data2` sources, accurately linking team names despite the spelling variations we introduced.

Obs	team	points	team2	assists
1	Mavs	19	Mavs	4
2	Nets	22	Netts	8
3	Kings	34	Keengs	8
4	Warriors	19	Warriors	12
5	Magic	32	Majick	7

By examining the structure of `data3`, we can confirm the accurate pairings: the entry "Nets" from `data1` has been correctly matched with "Netts" from `data2`, "Magic" with "Majick," and "Kings" with "Keengs." This positive outcome highlights precisely how utilizing phonetic encoding via **SOUNDEX** in conjunction with the precise edit distance calculation by COMPGED effectively overcomes the challenges posed by imperfect data. The process successfully identifies and unifies records that would have remained separate and unusable had we relied solely on a traditional exact-match join operation.

A defining element in securing these accurate fuzzy matches is the judicious selection and calibration of the similarity threshold (the condition `dif<=50` in our specific example). The ability to

adjust this numerical value grants the user fine-grained control over the strictness of the required match. A lower threshold will necessitate a much higher degree of similarity between records, which often results in fewer but potentially more precise matches. Conversely, setting a higher, more permissive threshold will capture a greater volume of potential matches but simultaneously elevates the risk of generating false positives (incorrect linkages). Therefore, determining the optimal threshold for any given project is a vital exercise that requires both empirical experimentation and deep domain knowledge related to the specific fuzzy matching task at hand.

Best Practices for Optimized SAS Fuzzy Matching

While the combination of **SOUNDEX** and COMPGED provides an excellent starting point for fuzzy matching within SAS, adhering to established best practices can dramatically boost both the accuracy and computational efficiency of your processes. A critical initial step is comprehensive data preprocessing. Analysts should standardize all input data by converting strings to a consistent case (all uppercase or all lowercase), meticulously removing extraneous punctuation, and trimming any leading or trailing whitespace. These preparatory steps significantly reduce noise and improve the consistency of comparisons, which in turn leads to far more reliable match scores and fewer false negatives.

Another essential consideration revolves around the selection and subsequent calibration of appropriate similarity metrics and thresholds. Depending entirely on the characteristics of your source data and the expected types of errors (e.g., phonetic vs. transpositional), you might choose to heavily prioritize phonetic similarity (via **SOUNDEX**), purely textual edit distance (via COMPGED), or a balanced combination of both. The threshold used for defining the maximum acceptable edit distance must be determined empirically, often involving a manual audit of sample matches to achieve the desired equilibrium between high recall (ensuring all true matches are found) and high precision (minimizing the occurrence of false matches). Iterative refinement of these key parameters is almost always necessary to achieve production-ready quality.

For organizations dealing with very large datasets, the brute-force methodology demonstrated in our tutorial example--comparing every record in one dataset against every record in the second using nested loops--can be prohibitively resource-intensive and slow. To optimize performance and ensure scalability, advanced data partitioning strategies like blocking or indexing should be considered. Blocking involves strategically dividing the datasets into smaller, more manageable subsets based on an initial, rough matching key (e.g., using only the first letter of a word or the **SOUNDEX** code itself). This technique drastically reduces the total number of pairwise comparisons required, fundamentally making the fuzzy matching process highly scalable for real-world enterprise data volumes.

Expanding SAS Skills: Further Resources

Beyond mastering the specific complexities of fuzzy matching, the SAS system provides an expansive toolkit of functions, procedures, and methodologies essential for comprehensive data manipulation and sophisticated statistical analysis. Achieving proficiency in these core tools is mandatory for any data professional whose workflow relies heavily on the SAS environment.

To continuously enhance your capabilities in data preparation, transformation, and statistical modeling within the SAS ecosystem, exploring targeted, authoritative tutorials is highly recommended. These resources typically cover a broad spectrum of common and advanced tasks, ranging from basic data hygiene operations to complex regression analysis, ensuring you are fully equipped to address diverse analytical challenges efficiently.

The following tutorials explain how to perform other common tasks in SAS:

Tutorial 1: Performing Advanced Data Cleaning Techniques in SAS

Tutorial 2: Using PROC SQL for Efficient Data Joins

Tutorial 3: An Introduction to Time Series Analysis using SAS/ETS