

String Similarity Evaluation Data for the New gTLD: Next Round

Published for Public Comment

11 October 2025



String Similarity Evaluation Data for the New gTLD: Next Round	1
1. Background	1
2. Methodology	3
2.1 Scope of Work	3
2.2 Similarity Category	3
2.3 Data Collection and Preparation	3
3. String Similarity Evaluation Data	5
4. File Format and Conventions	8
5. Additional Significant Considerations	9
5.1 String Similarity Due to Uppercase Form	9
5.2 String Similarity Suggestions for ASCII Strings	10
6. Contributors	12
Appendix A: Details on ASCII Code Point Similarity	13

1. Background

As recommended in the Generic Names Supporting Organization (GNSO) [Final Report](#) on the New gTLD Subsequent Procedures Policy Development Process, the String Similarity Review is a part of the New gTLD Program: Next Round application evaluation. The objective of this review is to prevent user confusion and loss of confidence in the Domain Name System (DNS) resulting from delegation of similar strings. The Phase 1 [Final Report](#) on the Internationalized Domain Names Expedited Policy Development Process adds further details on this process and the scope of string similarity analysis to cover strings (e.g., other applied-for strings, existing top-level

domains, Blocked Names) and their variant strings. It also requires the development of the String Similarity Review Guidelines for the String Similarity Panel (SSE Panel).

The comparisons between strings are performed for similarity evaluation. When variant strings are included in the scope, the number of comparisons increases significantly. Therefore, the String Similarity Evaluation Tool (SSE Tool) is being developed to perform prescreening and generate the potential contention sets as an input for the SSE Panel to analyze and make decisions. The SSE Panel can override the suggestions by the SSE Tool.

Following the [publication](#) of the draft of the String Similarity Review Guidelines for public comment which proposed this methodology, ICANN gathered String Similarity Evaluation Data (SSE Data) for the New gTLD Program: Next Round. As groundwork for the String Similarity Evaluation (SSE) for Top-Level Domains (TLDs), data had to be gathered at code point level for the full repertoire of the Root Zone Label Generation Rules ([RZ-LGR](#)). The task was extensive and required script-specific knowledge. This data defines the level of similarity between code points or code point sequences in the RZ-LGR repertoire and will be used in the SSE Tool.

It is important to note that the data is designed to be able to identify the potential visually similar code points, beyond the variant code points. In case of ambiguity if the code point pair is similar or not, a conservative approach is used by including them into the similarity set. Additionally, some code points are included in the similarity set to maintain the transitivity of the set, even if they are marked with lower similarity scores.

The prescreening SSE Tool uses this SSE Data to determine the potentially similar strings which are presented in the potential contention sets generated by the tool. These contention sets are not necessarily the final contention sets and will be considered by the SSE Panel for the final determination of the contention sets. It is the responsibility of the SSE Panel through its manual review to finalize the contention sets arising from string similarity. The SSE Panel may change the contention sets identified by the SSE Tool using the SSE Data.

Once the SSE Data is finalized, given the SSE Tool and SSE Data produced output, ICANN org will develop the final version of the String Similarity Review Guidelines for the SSE Panel and publish them for public comment.

2. Methodology

2.1 Scope of Work

ICANN has collected input from the relevant script experts listed in Section 6. These experts analyzed the RZ-LGR repertoire for their respective scripts, as well as cross-script comparisons with related scripts.

The work had the following scope for a script:

- A. Compare all elements¹ to ASCII characters, in uppercase and lowercase form (a-z, A-Z)
- B. Compare all elements to each other
- C. Compare all elements to elements in the repertoire of other related scripts (if any)
- D. Compare all elements to:
 - a. Capitalized elements of the related scripts (if applicable)
 - b. Conjuncts or compounded characters (if applicable for the script)
 - c. Simple shapes in ASCII, Latin, and other related scripts (e.g., circles “o”, lines “l”, curves “c”, “s”, “u”, “n”)
 - d. Underlined elements as may be used in URL links

2.2 Similarity Category

The similarity categorization consists of the following five levels:

1. Identical or near identical (homograph) or already defined variants
2. Highly confusable (strongly similar, or near homograph)
3. Similar
4. Distantly similar (weakly similar)
5. Distinct, not similar

2.3 Data Collection and Preparation

The task was partitioned into multiple sub-tasks described below:

- Expert’s Work
 - Each expert reviewed their respective script repertoire and compared all code point glyphs with each other and with the code point glyphs of related scripts (including ASCII characters).

¹ An element in this context refers to the visual representation of either a single code point or any valid sequence of code points within the repertoire of the script in RZ-LGR to be reviewed as a single unit within a domain label.

- Each expert noted the similar code points along with the degree of similarity and the reason for similarity based on the criteria presented above.
- Some experts developed additional mechanisms to perform their work within the guidelines. These mechanisms are listed in the *Additional Notes on Script Data* section later in this document.
- Manual Review and Programmatic Checks
 - The similarity data received from the experts was reviewed based on a list of items ranging from syntactical checks (e.g., the glyph and code points match) to scope completeness checks (e.g., confirming that the script character and ASCII characters comparison has been conducted). Any issues were reported back to the expert for explanation or remediation. If necessary, this step was repeated.
 - Following syntactic and completeness checks, to review consistency, further cross-script alignment was conducted on the data. All cases were identified in which the cross-script classifications of the two script experts involved were different (e.g., one script expert marked a different category than another script expert on the same pair). In such cases, ICANN org coordinated with relevant experts to align the data for relevant scripts.
 - Following the alignment, the transitivity check was done by integrating all similarity data across scripts as well as variant information as defined in the RZ-LGR
 - All variant pairs in Common RZ-LGR, which includes all variant definitions across scripts, were added as “Variant” and marked as Category 1
 - In addition, pairs that were not identified explicitly by experts were added as “Imposed” due to transitivity. For example, if A and B are marked as similar, A and C are marked as similar, then B and C will be “Imposed” into the same similarity set. In such cases, the imposed pairs were marked with similarity Category 4, because they had not been identified by the script experts.
- Conversion to Data Files
 - The cumulative data for each script was converted into the XML structure in the Label Generation Rules format (RFC 7940) to be processed by the SSE Tool
 - The equivalent HTML version was created to facilitate the manual review of these data files

3. String Similarity Evaluation Data

The files being published for public comment include one Common Similarity Data file and 26 Script Similarity Data files. Both sets of files are needed for different purposes in the calculation of the String Similarity as per the details below.

- The Common Similarity Data file is the merger of all Script Similarity Data files and the Common RZ-LGR. It is used for calculating the similarity label sets.
- Script Similarity Data files are generated based on the script expert input and variant definition in the Common RZ-LGR. They are used for calculating the level of similarity between labels within a similarity label set. These files are presented in a streamlined manner to better support review of the degree of similarity between code points. To make these data files usable for the intended distance calculation, they will be mechanically augmented by some of the mappings, tags, classes, and context rules found in the SSE Common file.

Script	Similarity Data	
Common	HTML	XML
Arabic	HTML	XML
Armenian	HTML	XML
Bangla (Bengali)	HTML	XML
Chinese	HTML	XML
Cyrillic	HTML	XML
Devanagari	HTML	XML
Ethiopic	HTML	XML
Georgian	HTML	XML
Greek	HTML	XML
Gujarati	HTML	XML
Gurmukhi	HTML	XML
Hebrew	HTML	XML

Script	Similarity Data	
Japanese (Han + Hiragana + Katakana)	HTML	XML
Kannada	HTML	XML
Khmer	HTML	XML
Korean (Han + Hanguk)	HTML	XML
Lao	HTML	XML
Latin	HTML	XML
Malayalam	HTML	XML
Myanmar	HTML	XML
Oriya	HTML	XML
Sinhala	HTML	XML
Tamil	HTML	XML
Telugu	HTML	XML
Thaana	HTML	XML
Thai	HTML	XML

These files can be collectively downloaded with this [package](#).

Additional Notes on Script Data

Hangul Script

Pairs of similar Hangul syllable-initial letters are identified. An example is (ᄀ, ᄁ). Then each pair of syllables (ᄀ, sp1, sf1) and (ᄁ, sp1, sf1) is compared where sp1 is one of 21 syllable-peak letters and sf1 is one of 27 syllable-final letters or null (i.e., no syllable-final letter).

A similar approach is adopted for similar syllable-final letters: each pair of syllables (si2, sp2, ᄀ) and (si2, sp2, ᄁ) is compared where si2 is one of 19 syllable-initial letters.

Examples of similar syllable-peak letters are (ᄀ, ᄁ), (ᄂ, ᄃ), (ᄄ, ᄅ). For these, each pair of syllables (si3, ᄀ, sf3) and (si3, ᄁ, sf3) is compared.

Han script

The Han script data was analyzed by both a Chinese expert and a Korean expert. There are 19,685 Han code points in Chinese RZ-LGR and 4,758 Han code points in Korean RZ-LGR. Of these, 4,744 code points are overlapped.

Due to the large size of Han script code points to compare, the Chinese script expert used automated machine learning tools, including a [Hanzi similarity Natural Language Processing \(NLP\) tool](#) and an [image visual similarity tool](#), to identify and categorize code point pairs for experts to review and make final decisions.

The Hanja (Han character used in Korean language) script expert referred to several sources containing pairs or sets of similar Hanja characters. Sets of similar Hanja characters are converted to pairs of Hanja characters. Pairs of similar characters in shape are manually analyzed to determine how much the characters in each pair look similar. Small differences in the top or left component seem relatively well distinguished whereas minor differences in the middle, center, or bottom component does not seem so easily identified.

The inputs from the Chinese and Korean experts were then compared and aligned before publishing for public comment.

Neo Brahmi scripts

Neo Brahmi scripts in the RZ-LGR, including Devanagari, Bangla (Bengali), Gujarati, Gurmukhi, Kannada, Malayalam, Oriya, Tamil, and Telugu, form their words following a particular way known as Akshar. These Akshars are typically formed by systematic combinations of the characters from the basic categories: Consonants (C), Matra/Vowel sign (M), Vowel (V), Anusvara/Bindu (B), Candrabindu (D), Visarga (X), Virama or Halant (H), Nukta (N). The potential basic akshar shapes formed by these characters can be achieved by following combinations: C, CM, CB, CD, CX, CMB, CMD, CMX. In addition, there are special characters available in certain scripts which have their own systematic combinations i.e., Chillu in Malayalam, Addak in Gurmukhi, Khanda Ta in Bengali.

The consonant in the word structure can be C or a conjunct formed among the Cs joined by H, e.g. CHC, CHCHC, which creates conjuncted shapes which may or may not retain the basic shape of the consonants that participate in the conjunct formation.

Each script expert took into account these plausible combinations and the subsequent shapes they create into the similarity analysis.

4. File Format and Conventions

These SSE Data files present an overview of all the code points and sequences within and across the scripts that are deemed confusable. Though it uses the RFC7940 format in the XML and similar HTML representation as RZ-LGR, the contents of this document do not define Label Generation Rules but SSE Data.

In the HTML format, this summary is contained in the **Repertoire** table. For each listed item in the table, there are one or more applicable **Confusables Sets**.

Repertoire

This section lists the repertoire by code point (or code point sequence) for which a confusable is defined, additional information is provided in the Confusables column.

Confusables Sets

The data that define the mapping between code points and sequences, including the degree of their similarity or confusability is collected in Confusables Sets which are symmetric and transitive. They also include all variant mappings defined in the RZ-LGR.

Each Confusable Set is presented in the same format as a “Variant Set” in the RZ-LGR, even though only some of the mappings are actually defined as variant mappings. Those are marked appropriately.

Reason Code

Each mapping has an associated “Reason” code that is a highlighted part of the Comment column. The reason code and its explanation are:

- **Capitalization:** The similarity is between two uppercase forms or between an uppercase and unrelated lowercase form
- **Confusing:** A majority of users familiar with the script of either target, source or both will confuse the pair unless viewed side-by-side
- **Fallback:** One of the code points is a common fallback representation for the other.
- **Font:** The confusability depends on font choice.
- **Handwriting:** The confusability is caused or strengthened by users being familiar with a shape, even though it is normally only used in handwriting
- **Other:** None of the other reason codes applies. In particular, it applies when the category is homograph (1) for a mapping that exceptionally has not been defined

as a variant. NOTE: This code MUST be accompanied by a comment explaining the nature of the confusability and reasoning for choosing the assigned category value.

- **Underlining:** The presence of underlining makes these confusable. This is typically the case when the distinction between two characters rests primarily on a mark below or a descender that might be obscured when underlined.
- **Variant:** The mapping is defined as a variant in the RZ-LGR. Note: the category value is set to 1 by definition

In addition, there are two special reason codes.

- **Not_confusing:** Mappings that are marked with Reason code “not_confusing” are listed here for reference only. They will be ignored in screening or label distance calculations. It is included here primarily for reference to document that the mapping was considered by the expert team but explicitly resolved as not confusing despite some weak similarity. These mappings are flagged with a symbolic context “excluded-similarity” in the “context” column.
Note: Some of the variant pairs are in the excluded-similarity set due to transitivity. In such cases the context is “excluded-similarity” but the Similarity Category remains as 1.
- **Imposed:** This code is automatically generated when a mapping has to be imposed to make a confusable set transitive after Variants have been injected.
Note: All imposed mappings are assigned Category 4 by default. This value may need to be replaced with one of the other category values using an explicit assignment based on the input received during public comment.

5. Additional Significant Considerations

5.1 String Similarity Due to Uppercase Form

As some web browsers may accept domain names in uppercase forms and automatically convert them to lowercase form before resolving, the uppercase forms are included in the similarity consideration. This may result in some of the code points which are not visually similar in lowercase to be defined as similar if their uppercase forms are determined to be similar.

For example: n (U+006E, Latin Small Letter N) and v (U+03BD, Greek Small Letter Nu) have visually the same uppercase letters.

n (U+006E, Latin) — uppercase → N (U+004E, Latin Capital Letter N)
v (U+03BD, Greek) — uppercase → N (U+039D, Greek Capital Letter Nu)

Therefore, n (Latin Small Letter N) and v (Greek Small Letter Nu) are defined as similar in the SSE Data due to uppercase conversion. The SSE Tool will calculate the similarity between strings accordingly e.g. the string “ana” (Latin) vs “ανα” (Greek) will be reported in the same potential contention set because their uppercase forms are homoglyph. This potential contention set will be presented to the SSE Panel for their review and final decision.

ana (Latin) and ανα (Greek) (lowercase)
ANA (Latin) and ANA (Greek) (uppercase)

Further details of similarity due to uppercase form are included in Appendix A.

5.2 String Similarity Suggestions for ASCII Strings

In addition to the SSE Data, RZ-LGR defines variant definitions of all scripts. Variant definitions and SSE Data are combined to be able to identify the potential contention sets.

For example: v (U+0076, Latin Small Letter V) and ν (U+03BD, Greek Small Letter Nu) are variant code points as per the Latin script RZ-LGR and Greek script RZ-LGR.

Through transitivity, the code point v (U+0076, Latin Small Letter V) are included in the similarity set with n (U+006E, Latin Small Letter N) and ν (U+03BD, Greek Small Letter Nu) from the previous example.

Similarity set: { ν (U+03BD, Greek Small Letter Nu),
v (U+0076, Latin Small Letter V),
n (U+006E, Latin Small Letter N) }

Based on the data, the SSE Tool will put the following labels in the same potential contention set: “ana” (Latin), “ava” (Latin), “ανα” (Greek). So, it is possible to have two ASCII strings in the same potential contention set, presented to the SSE Panel for their review and final decision.

As per the RZ-LGR (Version 6) and the SSE Data being published for this public comment. The following ASCII code points are identified in the same similarity sets (see details in Appendix A).

Set 1: {f, t}

Set 2: {i, l}

Set 3: {h, n, v}

Set 4: {r, u, y}

Even though being in the same potential similarity set caused by transitivity and variant relationships within script or across script, the actual similarity between each pair of ASCII code points are all Category 4 (not confusing) or Category 5 (not defined), except for the i (U+0069, Latin Small Letter I) and l (U+006C, Latin Small letter L) where the similarity is defined as Category 1.

The i (U+0069, Latin Small Letter I) and l (U+006C, Latin Small letter L) are considered similar because the uppercase i and lowercase l are homoglyphs in some fonts. Even though they are a mixed-case comparison for ASCII, they can create confusion especially when there are no other code points to identify the case of the strings e.g. Ill. Therefore, i and l are included in the similarity set which will be used to produce candidate similar strings for manual evaluation by the SSE Panel for decision.

After potential similarity sets are defined, the actual similarity score is calculated by comparing each string (and its variant) within the set.

For example, when “how” and “now” are reported in the same potential contention set by the SSE Tool. The tool will compare the strings and its variants against each other and list the mapping category. For simple demonstration, only the primary strings comparison is listed below.

how = U+0068(h) U+006F(o) U+0077(w)

now = U+006E(n) U+006F(o) U+0077(w)

Based on the SSE Data for Latin script (being published for this public comment), the similarity between h and n is Category 4 while the remaining code points are identical (Category 1). The mapping category for “how” vs “now” is [4-1-1].

If a mapping is not all Category 3 or less, it means that there is at least one code point which can differentiate the two strings. The mapping category information will be presented to the SSE Panel and they may decide that these are not similar.

6. Contributors

Script Experts:

Akshat Joshi - Devanagari, Gujarati
Atiur Rahman Khan - Bengali (Bangla)
Dessalegn Yehuala - Ethiopic
Dmitry Belyavsky - Cyrillic
Gurpreet Singh Lehal - Gurmukhi
Harsha Wijayawardhana - Sinhala
Hiro Hotta - Japanese (Hiragana, Katakana)
Katarina Gevorgyan - Armenian
Kim Kyoungsook - Korean (Han, Hangul)
Kuldeep Patnaik - Oriya
Makara Sok - Khmer
Matitiahu Allouche - Hebrew
Michael Bauland - Latin
Naail Abdul Rahman - Thaana
Nabil Benamar - Arabic
Panagiotis Papaspiliopoulos - Greek
Parkpoom Tripatana - Thai
Pavanaja U B - Kannada
Saysomvang Souvannavong - Lao
Shanmugam Rajabadher - Tamil
Thin Zar Phyo - Myanmar
Uma Maheshwar Rao G - Telugu
Veena Solomon - Malayalam
Wang Wei - Chinese (Han)
Zakharia Pourtskhvanidze - Georgian

Contractor:

Akshat Joshi
Asmus Freytag
Michael Bauland

ICANN Staff:

Pitinan Kooarmornpatana
Sarmad Hussain

Appendix A: Details on ASCII Code Point Similarity

Detail of Set 1 {f, t}.

U+0066 (f) <-sim2->U+01AD (ƒ) <-simn2->U+0074 (t)

- The f (U+0066, Latin Small Letter F) and the ƒ (U+01AD, Latin Small Letter T with Hook) are similar as Category 2.
- The ƒ (U+01AD, Latin Small Letter T with Hook) and the t (U+0074, Latin Small Letter T) are similar as Category 2.

Detail of Set 2 {i, l}.

U+0069 (i) <-sim1->U+006C (l)

- The i (U+0069, Latin Small Letter I) and the l (U+006C, Latin Small Letter L) are similar as Category 1.

Detail of Set 3 {h, n, v}.

U+0068 (h) <-sim1->U+03B7 (η) <-var->U+006E (n)
<-sim1->U+03BD (ν) <-var->U+0076 (v)

- The h (U+0068, Latin Small Letter H) and the η (U+03B7, Greek Small Letter Eta) are similar as Category 1 because their uppercase letters are homoglyph;
 - H (U+0068, Latin Small Letter H),
 - Η (U+0397, Greek Capital Letter Eta).
- The η (U+03B7, Greek Small Letter Eta) and the n (U+006E, Latin Small Letter N) are variant code points.
- The n (U+006E, Latin Small Letter N) and the ν (U+03BD, Greek Small Letter Nu) are similar as Category 1 because their uppercase letters are homoglyph;
 - N (U+006E, Latin Capital Letter N),
 - Ν (U+039D, Greek Capital Letter Nu).
- The ν (U+03BD, Greek Small Letter Nu) and the v (U+0076, Latin Small Letter V) are variant code points.

Detail of Set 4: {r, u, y}.

U+0072 (r) <-var->U+0433 (Ɑ) <-sim1->U+03B3 (γ) <-var->U+0079 (y)
<-sim1->U+03C5 (υ) <-var->U+0075 (u)

- The r (U+0072, Latin Small Letter R) and the Ɑ (U+0433, Cyrillic Small Letter Ghe) are variant code points.
- The Ɑ (U+0433, Cyrillic Small Letter Ghe) and the γ (U+03B3, Greek Small Letter Gamma) are similar as Category 1 because their uppercase letters are homoglyph;
 - Γ (U+0413, Cyrillic Capital Letter Ghe),
 - Γ (U+0393, Greek Capital Letter Gamma).
- The γ (U+03B3, Greek Small Letter Gamma) and the y (U+0079, Latin Small Letter Y) are similar.
- The y (U+0079, Latin Small Letter Y) and the υ (U+03C5, Greek Small Letter Upsilon) are similar because their uppercase letters are homoglyph;
 - Y (U+0059, Latin Capital Letter Y)
 - Υ (U+03A5, Greek Capital Letter Upsilon)
- The υ (U+03C5, Greek Small Letter Upsilon) and the u (U+0075, Latin Small Letter U) are variant code points.