String Similarity Review Guidelines

Date: 2024-02-07

This is a Public Review Draft

Abstract

This document provides detailed guidelines for conducting String Similarity Review in the context of labels being considered for Top Level Domains for which a risk of visual confusability could be established. With a larger set of existing delegations plus variant labels, the scope of String Similarity Review and the associated challenges have increased. These guidelines provide background and criteria for the determination of various similarity factors plus a methodology for an efficient evaluation process by which this review can be conducted. They are designed as foundational input for the work of a String Similarity Review Panel in implementation of string similarity related policy recommendations.

CONTENTS

1 Overview

This document discusses the issues defining and affecting the confusable[1] visual similarity of strings used as labels in domain names. The degree to which a label that is presented to a user is confusingly similar to the label the user expects depends sensitively on a number of factors discussed here. They are related to the nature of the script of the label in question as well as the familiarity the user has with reading that script. The guidelines necessarily have aspects that are specific to individual scripts or families of related scripts, and the organization of this document will reflect that where appropriate. At the same time, the degree to which domain names are applied for is uneven across the scripts for which the Root Zone provides Label Generation Rule-sets (LGRs). This may result in a certain prioritization.

The document gives practical guidelines of how to prepare for and organize the review of applied for labels with respect to other applied for labels, already delegated labels, reserved words or their variants [Reserved]. The nature of the task makes brute-force approaches computationally infeasible in the general case and the guidelines attempt to suggest effective heuristics. Nevertheless, there is an irreducible aspect of String Similarity Review that will require manual review and judgment. As needed, this will be carried out by expert panels, with input from experts knowledgeably in the various scripts, but without the use of focus groups or end-user surveys.

String similarity as defined here is solely concerned with visual confusability and other, potentially related or even overlapping concerns have been ruled non-goals.

1.1 String Similarity in the Context of Variants

Among the basic assumptions made in this Study is that any label that is a blocked variant has already been eliminated via testing for label collisions enforced by the variant mechanism defined in the Root Zone LGR.[2] The definition of variants in the Root Zone covers labels that are considered the “same” based not only on appearance but also on other causes, such as semantic or phonetic equivalence. However, the definition of variants excludes any form of similarity that is confusable without being (nearly) identical visually, or that arises from the whole label.

There are some additional limitations that are a consequence of details of the procedure for defining Root Zone variants. For example, any equivalence between two ASCII characters was considered out of scope. Where possible, the process described in this study will make up for such limitations. In particular, the scope of this study includes visual similarities between ASCII strings.

Variants in the Root Zone may be “blocked” or “allocatable”, the latter allowing delegation to the same entity, but not otherwise.[3] For String Similarity Review in the ccTLD Fast Track Process,[4] there exists the possibility of allocating two confusable labels to the same entity, but the same does not exist in the current gTLD policy.

---

[1] This document uses confusable both as an adjective and noun, not unlike the use of the term variant.

[2] For definitions of terms or explanations of concepts see Section 2 “Key Concepts” and Section 15 “Definitions”.

[3] See Recommendation 25.6 in GNSO Sub Final Report on the new gTLD Subsequent Procedures. [SubPro]

[4] See Final Implementation Plan for IDN ccTLD Fast Track Process. [FastTrack]

1.2 Two Stage Process

As discussed here, brute force String Similarity Review, where a candidate label is compared against all existing labels (and their potential variants) is not feasible. Any realistic process has to be efficient and scalable. A promising approach will be a two-stage process consisting of a data driven pre-screening followed by a focused review of remaining candidates.

The review process will be undertaken by a contracted expert panel which may engage native experts in the scripts. The focus of this review, as well as for the pre-screening, is on confusable similarity, as experienced by a competent native user who is reasonably attentive.

1.3 Interaction with Scripts and Languages

Where scripts are shared by more than one language, users familiar with one language may not know of or expect features common in writing one of the other ones. As a result, they may confuse them with forms native to their language.

Likewise, users presented with a label in a different script may mistakenly accept the whole label as one in their own script, so for evaluating cross-language or cross-script similarity more than one frame of reference may be considered. However, for any pair of labels clearly in one language or one script, any possible confusion by non-native users is to be considered irrelevant.

Scripts have a history: they develop and change over time, but also give rise to other scripts derived from or inspired by them. For modern usage it means that scripts may have historical and other relationships that impact the extent of string similarity. In addition, font resources, particularly those used in user interfaces have exhibited a clear tendency towards harmonized shapes and metrics for related scripts, with obvious impacts for string similarity.

String Similarity Review has to take these differences and relations between scripts into account; any guidelines will by necessity include considerations that are specific to a given script or family of related scripts. Some scripts have seen a larger number of TLD applications; this may be a factor affecting a possible prioritization in the development of script-specific guidelines.

The following sections introduce the key concepts in more detail followed by a discussion of the String Similarity Review process.

PART I – BACKGROUND

2 Key Concepts

This section sets the stage by giving definitions for some of the key concepts:

- Confusability
- Similarity
- Variants

The first two are characterized by not being black-and-white, but existing on a spectrum; by contrast, the corresponding definitions from the policy documents [SubPro] implicitly assume a cutoff. For the purpose of the discussion here, which is to no small degree concerned with arriving at an operational definition of such a cutoff, it is beneficial to start with these more general definitions.

2.1 Confusability

Confusability: the degree to which two labels can be substituted for each other without that fact being noticed. Users are confused about which label they are selecting.

Confusable: said of two labels for which the degree of confusability exceeds a defined threshold.
- This threshold is a matter of definition, there is no a-priori "bright line" where confusability starts.

Sometimes "confusability" is used in a way that implicitly assumes a hard cutoff; a label is considered either confusable or not confusable with another label. That assumption of two mutually exclusive states is not an accurate description for the general case. It is therefore more useful to think of how readily two labels can be confused and that confusability represents a measure of this. In this view, once a certain threshold value of confusability has been reached, it becomes advisable or imperative to take mitigating measures. In short, two labels are "confusable" if their mutual confusability exceeds a defined threshold.

It would be a mistake to assume that it is possible to define a confusability threshold in a context independent way or to accurately "compute" the degree of confusability from the code point string, whether in absolute terms or in relation to a threshold. There have been attempts to empirically determine confusability based on presenting the rendered forms of labels to test subjects.[5] However, no widespread consensus has been developed regarding what constitutes an acceptable methodology, nor is the protocol necessarily scalable to large numbers of labels, or labels more complex than two-letter ccTLDs. Instead, it is better to consider confusability an approximate measure and moreover one that is subject to be significantly affected by the linguistic background of the user and the context in which the confusing label is presented.

When determining the threshold for confusability for the purpose of these guidelines, it is necessary to consider the nature of the Root Zone and the presence of very short labels of two code points (or potentially fewer).

[5] This includes studies like the "Extended Process Similarity Review Panel" carried out for the .eu ccTLD (https://www.icann.org/en/system/files/files/epsrp-european-union-30sep14-en.pdf). For the Root Zone LGR, the Japanese Generation Panel solicited studies comparing highly confusable code points using two survey test subjects but with different methodologies [Survey].

While analyzing the confusability between isolated code points is insufficient to derive the confusability of the labels they constitute in a deterministic way, such analysis is a useful precursor to gain a rough approximation or estimate, which might be utilized in mechanically pre-screening labels (see 9.3 “Pre-Screening”).

2.2 Similarity

String Similarity: similarity in appearance, of a kind that contributes to the confusability between two labels, visual similarity.

In the context of this discussion and in particular for this document, the term “similarity” is understood to mean “visual similarity”.

Some fonts, including the ones used predominantly in user interfaces, reduce the distinction between characters by reducing the amount of detail in each glyph. Others deliberately “harmonize” the appearance of letters across related scripts.[6] Such effects can increase the similarity between strings when presented that way. In evaluating string similarity, we proceed from the assumption that the least distinct presentation is used, thus accounting for any situation where similarity is increased to the maximum by the manner of presentation.

Confusable Similarity: the degree of similarity required to make two similar labels “confusable”.

The definition of “String Similarity” specifically references such similarities in appearance as can contribute to the confusability between two labels. Like confusability, similarity is a matter of degree and depends on the background of the users and context of presentation. Strings do not have a fixed appearance; they are presented to the user in rendered form, the details of which depend on the system or process doing the rendering as well as the data used for the purpose, including font resources. String similarity is not a deterministic function of comparing the code points in the two labels.

Confusability goes beyond direct similarity between the appearance of two labels presented side-by-side but is based on trying to match a presented label with an intended one. A label can look similar to an intended label even if the difference could be obvious in a side-by-side presentation.

Users familiar with a script will also be familiar with common alternate forms of any character they know and may accept look-alikes to these forms, not only look-alikes to the regular forms. Particularly, if they are presented in the context of a word at the expected position. Unlike trained typographers, general users will not keep track of distinctions and visual features that do not matter to understanding.

For example, some characters have noticeably different shapes when typeset in italics, or certain font families. An example is the Cyrillic small letter “i” (regular: *и*, italic: *u*), which in its italic form resembles the italicized form of Latin small letter “u”. Another, more subtle example is Latin small letter “f” (regular: f, italic: *ƒ*) which resembles Latin small letter f with hook (regular: ƒ, italic: *ƒ*). The latter is an example of a letter unfamiliar to most users of the script, and therefore not recognized as a potentially distinct code point.[7]

Similarity is therefore not an objective or absolute measure. Like confusability, it is affected by past experience and present expectations of the user. For the context of the mitigation measures discussed in

[6] For examples, see Section 8.6, “Relationships between Writing Systems and between Scripts”.

[7] The Root Zone LGR considered this example as particularly risky and treats it as a variant.

this study, only similarity experienced by users familiar with a script will be considered; but this does not necessarily assume that the user is familiar with all possible writing systems based on that script.

2.3 Variants

Some strings are so similar that they have identical or near identical appearance. They are thus examples of variants:

Variant: a label that a user will accept as "same as" the original label. Variants based on visual similarity have identical or near identical appearance, but variants are not limited to visual similarity.

The Root Zone defines variants as mappings between code points or code point pairs both within scripts as well as across scripts. A label that can be reached by applying some combination of these mappings from an original label is considered a variant label. Its disposition depends on which mappings are used.

Original label: Here: the label which the variant label is a variant of; the label to which variant mappings are applied to reach the variant label. Called *Primary Label* in the context of an application.

Variant mappings exist between code points or code point sequences of (near) identical appearance, but also between code point that have the same meaning or pronunciation, or that can otherwise be freely substituted for each other without changing the intended meaning.

Variants in the Root Zone LGR may have dispositions as "blocked" or "allocatable", the latter allowing delegation to the same entity, but not otherwise. Labels that are blocked variants of each other, or are allocatable variants, but applied for by different entities do not usually need to be compared for similarity: they would have been rejected in an earlier stage of the process, or they are already in contention as having been applied for at the same time while being blocked variants.

Because variant labels represent alternate forms of a label that users will not distinguish from the intended label, any label that has confusable similarity to a variant label is de facto confusable with the original label.

For a more detailed discussion of Variants and how they intersect with String Similarity Review see 4.2 "Interaction of String Similarity Review with Variants".

3 Recognition Task and Context Dependence

As we have established, what looks similar and what can be confusable depends on the background of the user and expectations learned in reading native language in native script.

One factor that strongly influences confusability is the familiarity of the user with the script or writing system (orthography). Users unfamiliar with a script may not be able to reliably tell two labels apart that present no such difficulty to anyone used to reading it. Conversely, users familiar with a script may treat two characters as acceptable alternates of each other, even though they do not look the same or even similar. A common reason for that would be that they represent the same meaning or sound.

Finally, users familiar with a modern orthography for their language, written in a given script, may not expect to see some historic or obsolete characters, nor characters used only in other languages for the same script. In these cases, users may well accept such characters as a substitute for a familiar character, especially if an unfamiliar character is substituted that happens to look like a familiar character in that position in a word.

3.1 Target User

Defining guidelines for evaluating two labels for confusable similarity requires further definitions:

Target User: a typical user in terms of which confusability is defined. This user is characterized by:
- being reasonably attentive
- being natively familiar with script or language (of the intended label)
- having some familiarity with the ASCII subset (no matter the user's native script)

For the purposes of this study, "confusability" is thus based on that experience by a native user of a script[8] when presented with a plausible alternative to an expected label. The typical target user is also assumed to have some familiarity with the ASCII subset and may be habituated to seeing non-IDN fallbacks; such fallbacks may have originated from the time before the introduction of IDNs, or they may represent convenient "universal" labels that are accessible to a wider audience than a more linguistically faithful label represented by an IDN. Confusability is further understood as something that is evaluated against a threshold, and one task of this study is to suggest a working definition of such a threshold.

3.2 Recognition Context

Finally, implicit in the definition of the Target User are assumptions about the recognition task. Many laboratory experiments with confusable strings present alternate versions of the string either side by side or in close temporal proximity. A more realistic real-world scenario is one where the user "knows" a name (based on having heard or seen it on an earlier occasion) and is now presented with a label that purports to be based on that name. This leads to the following definition:

Recognition Context: the context in which the Target User is asked to distinguish between the intended label and some alternative. For this discussion, the recognition task is assumed to:
- not require that both labels are presented simultaneously,
- include the case of a user being presented with label in a foreign script or language when looking for a native label,
- exclude the case of a user attempting to decide between two labels in a foreign script or unfamiliar language,
- proceed holistically for the whole label, not character-by-character.

While many users will have the visual acuity to note that two labels presented side-by-side differ in the number, kind or placement of diacritical signs, for example, they may not spot an extraneous or modified diacritic when the label is presented in isolation (or in a context, like a phishing website, that prepares the user to expect a particular label, for example by duplicating the look and feel of a legitimate site).

It is thus important that small distinctions are not given overly prominent weighting when deciding in favor or against a verdict of confusable similarity.

3.2.1 Identifier Recognition vs Reading

The task of identifier recognition is clearly related to reading, but there are important differences. In ordinary reading, the primary task is to identify words in contexts where these words are presented with their correct spelling and where any variation in the shape of individual letters is ordinarily the effect of stylistic parameters, such as font choice. Even where such presentation is fanciful and out of the ordinary, such as when decorative fonts are used, or when the text is presented in a random mixture of

[8] Most users have some familiarity with ASCII letters in addition to their own script. Beyond that, familiarity with more than one script (or more than one orthography) is less common.

fonts and styles (“ransom note”), most readers have little trouble to identify words and to extract the meaning of a text passage.

Some words resemble other words, or they are particularly difficult for readers to distinguish based on being unfamiliar with them, such as “militate” *vs.* “mitigate”. The most difficult of these cases are those where both words not only look vaguely similar but can be substituted without grammar constraints. Only by checking the assumed meaning against a larger context would a reader be able to ascertain whether their reading is correct.

In contrast to reading, successful identifier recognition shows more similarities to the task of spell-checking. In a first pass, the identifier is recognized, either against a known word, or against a known mnemonic string, or perhaps compared to a pronunciation. The second pass would require the user to then verify the spelling: does the sequence of letters match a known identifier, or is there a deviation in the presentation that hints at a possible substitution of a confusable label?

3.2.2 Recognition Applies to Whole Labels

Word recognition[9] is holistic; likewise, the complete label sets up a semantic or recognition context where the reader, having seen the entire label, has an expectation for the spelling of it that can subconsciously override certain infidelities in presentation. If a reader expects a “v”, but sees a “ν” (Greek nu), the slight difference in presentation will most likely be ignored, or, if noticed, put to a quirk in font design. In addition, it is accepted that word detection relies on overall features of a word; when these are closely spaced, readers don’t always resolve which letter contains which feature.

In this context, it is worth noting that when evaluating visual evidence, users may reach conclusions that are directly opposite to the visual input received, due to any number of preconceived notions. Examples of this effect include French users denying the actual use of accented uppercase letters because they were taught, they are not used, Japanese users rating confusability between Kanji and Kana lower than revealed in tests because they assume them to be inherently distinct, and many more.

Therefore, in evaluating whether two labels are confusingly similar, it is not sufficient to proceed on a character-by-character basis, but necessary to work in the context of the entire label string.

On the other hand, a label may be deemed unique (not confusable with any label) if it contains a substring with unique visual appearance, particularly if that unique part is visually prominent.

3.3 Candidate Label and Review Targets

Candidate Label: a label that is to be evaluated for confusable similarity against one or more other labels.

Usually, an applied-for label would be the candidate label which is then evaluated against the already delegated labels and their variants, but also against labels from other applications, or reserved words [Reserved] and so on. For simplicity of discussion, when not otherwise required, we will assume the case of a single candidate label being evaluated against delegated labels and not always mention explicitly any other labels that the candidate also needs to be reviewed against. This is not intended to suggest that in actual String Similarity Review the other labels, including other pending applications, reserved words, and so on can be ignored, nor that their variants do not have to be taken into account.

---

[9] Interesting reading on the subject can be found at https://learn.microsoft.com/en-us/typography/develop/word-recognition

The candidate label is reviewed against a shortlist of review targets.

Review Target: a label that is shortlisted for further review against the candidate label.

- typically selected in a pre-screening process that eliminates labels of low confusability.
- when evaluating the contention between two or more applied for labels, each one is in turn considered the candidate label while the others are treated temporarily as review targets.

The set of labels that a candidate label must be evaluated includes delegated labels, reserved words, and their variants, as well as other applied for labels in the same round. This set is potentially too large to allow a detailed review against all of them. Some means of shortlisting (automated) must be employed to remove from consideration any labels with predicted low similarity or confusability.

3.4 Goal of String Similarity Review

Anybody who ever tried to locate a misspelling knows the problem of typos that seemingly hide in plain sight. A reader's expectation of what the correct spelling ought to be can interfere with the task of recognition, making detection of an error (or a substitution) nearly impossible.

The task of String Similarity Review can be summarized as an attempt to ensure that all labels are sufficiently distinct from each other, by disallowing concurrent[10] labels where either part or the entire label is substituted for a confusingly similar string.

But as the experience with typos shows, it would be daunting to eliminate all strings that might possibly be confused by some readers some of the time. Instead, the task should be focused on those substitutions where the similarity is strong enough to engender a strong probability of confusion, perhaps affecting most readers most of the time.

4 Relation and Contrast to Variants

When it comes to understanding labels that are similar, it is useful to consider them in the context of variant labels. This poses the following questions:

- How are variant labels different from confusable labels?
- Where and how does String Similarity Review interact with defined variants?

4.1 How String Similarity and Confusability Differ from Variant Relations

Recall the definition of *Variant*: a label that a user will accept "same as" the original label.

This definition seemingly does not differ much from that of a confusing similar label. However, there are a number of differences based on the distinction between "similarity" and a "same as" equivalence:

- Variant is an equivalence relation, and therefore transitive:
    - All variants for a label are variants of each other as well,
    - Variant based on visual similarity typically have identical or nearly identical appearance.
- In contrast, "mere" similarity implies some distance in Perception Space:

[10] The exceptions are allocatable variants, that are delegated to the same registrant and that are intended to access the same or equivalent resource, so that substitution is benign.

- Two labels similar to a third may not be similar to each other.

⇒ Similarity and Confusability are not transitive in the general case.

Another set of difference arises from the kinds of relation between labels that are considered or not considered for variants or string similarity:

- Variants are not limited to visual similarity; in fact, many variant code points are defined on the basis of equivalent sound, or equivalent meaning,
- Variants do not consider uppercase-based similarity,
  - However, effects of uppercase-based confusability are in scope for string similarity.
- Variants are defined based on mappings between code points or code point sequences,
  - They fall short in addressing confusability of whole strings.

Finally, it is worth noting that variant labels are explicitly defined in the [RZ-LGR] on the basis of variant mappings for specific pairs of code points or code point sequences. For the purpose of String Similarity Review, whether two labels are variants of each other can be computed based on their strings alone. The same is not true for similarity or confusability.

The difference becomes apparent when comparing a variant set, which is the collection of all variant labels or code points that are mutually variants of each other with the definition of a confusables cluster:

Confusable Cluster: confusables for a certain label cluster around it in perception space.

- applied to code points, a confusables cluster represents all code points or sequences that are directly or by transitivity confusable with a given code point or sequence;
- not all members are equally confusable with each other.

A confusables cluster treats confusability as transitive. That creates a superset that should include all code points or labels that are in fact confusable, but also some that under more detailed review will be found not to be confusable after all.

4.2 Interaction of String Similarity Review with Variants

Some of the delegated labels may have allocatable variants that are also delegated, but many more have a large number of blocked variants. The applied for label may also have variants. These variant labels must be considered when evaluating whether two labels are confusable.

- All variants of a Candidate Label are also confusable with the delegated label, independent of whether they are applied for or not.
- A Candidate Label may be confusable with a variant of a delegated label, even if that variant is not delegated.
- Any variant label would be accepted as a “same as” of label of which it is a variant.
- A Candidate Label or its variant that is confusable with any variant of a delegated label should be evaluated on same footing as a Candidate Label that is confusable with the delegated label itself.

Because variants are defined according to what users consider as effectively “same” as the original label, any new label that is confusingly similar to any such variants would look like something that the Target User would consider as equivalently “same” as the original label. From that it follows that it is necessary to consider the interaction between all variants of the two labels being evaluated for confusability, as

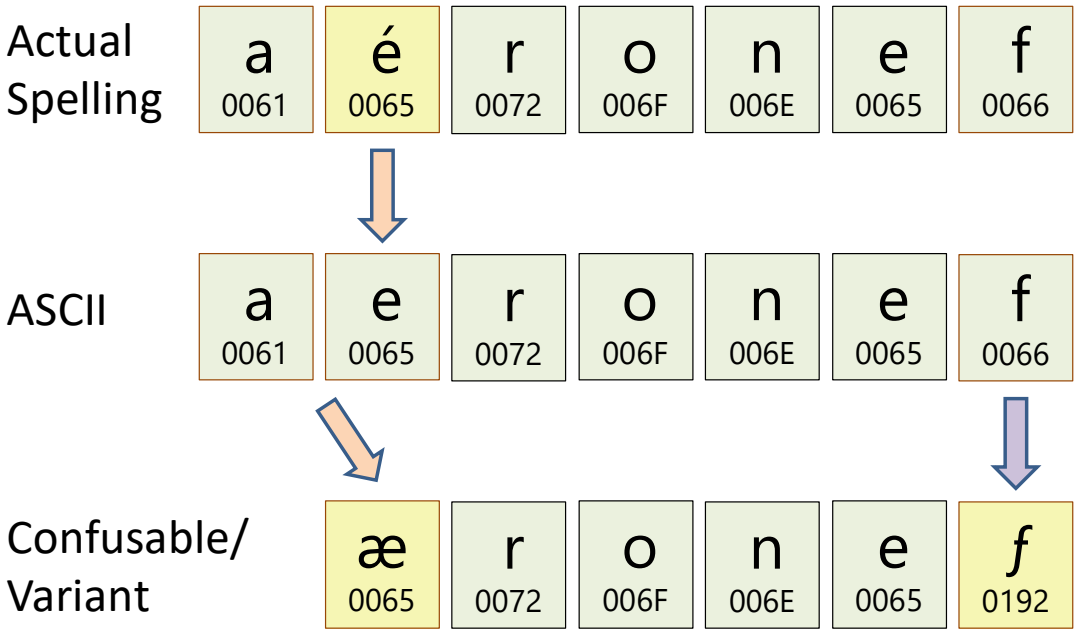
long as they are themselves potentially valid labels (and not, for example, mixed-script). The Target User has no way of knowing whether a given variant of any label is delegated or not; the user can only decide whether it looks like it could be the intended label or something that is fully equivalent.

Reviewing string similarity against variant labels creates an interesting challenge. Any label could potentially have quasi-infinite numbers of blocked variants, especially if the label is long. As discussed below (see Section 9 "Section Scalability of String Similarity Review") the enumeration of every possible variant may not be feasible and therefore requires special techniques that bypass enumeration.

4.3 Labels That Have Confusables as well as Variants

One of the challenges in dealing with confusables in the context of variants is that one and the same label can have code points with both confusables and variant mappings. While labels with only variant mappings are taken care of in stages that precede String Similarity Review, the mixed cases still need to be handled.

The following example shows first an actual word, followed by its fallback ASCII spelling and lastly, a potential confusable that also exhibits one of the variants defined in the Root Zone LGR (purple arrow). Any number of further permutations are possible, but not shown. Note that one of the confusables is between a code point and sequence.



The easiest way to deal with such a mixed case might be treating the variant as a "super confusable" because a variant is by definition something a user can be expected to accept as a substitute, just like a code point of identical appearance. A confusable label that also contains variants would thus be treated as having a smaller distance in perception space than one that only contained confusables of average confusability. (See also the discussion in Section 5.3.1, "Confusability and Proximity in Perception Space".)

5 Type of Similarity for Code Points

Some types of confusability are exclusively or predominantly encountered in certain scripts or sets of related scripts. Even if they can be encountered in some form in some unrelated scripts, their discussion is not repeated.

- Diacritics and Confusability are treated in Annex A.2.
- Digraphs and Uppercase are considered in Annex A.3 and Section 6.

The remainder of this section discusses the various forms of similarity in descending order of probable confusability:

1. Identical or Near Identical Labels
2. Highly Confusable Labels
3. Similar Labels
4. Distantly Similar Labels
5. Distinct Labels, not similar

5.1 Identical or Near Identical Labels

5.1.1 True Homographs

There are many cases where a pair of strings look exactly identical, even though each one is written in a different script. For example, many such correspondences exist between Kannada and Telugu scripts, or between Latin and Cyrillic, as in this example:

EXAMPLE: “.cap” (Latin) *vs.* “.cap” (Cyrillic)

Such true homographs usually lead to a collision when processing the label using the Root Zone LGR because the corresponding code points are treated as cross-script variants. (Any additional instances of true homographs found in String Similarity Review that were omitted from the variant definition for any reason, should nevertheless be treated as confusable)

5.2 Highly Confusable Labels

Some correspondences are not close enough to be considered identical or nearly identical, but they are nevertheless confusing.

5.2.1 Near Homographs

The line is not sharply drawn. Some near homographs have been assigned as variants in the Root Zones, while others are in scope for String Similarity Review, including those that are confusable despite some degree of visual distinctness.

EXAMPLE: “.ρορ” (Greek) *vs.* “.pop” (Latin)

The lack of a “corner” for the “p” makes these distinct in a side-by-side comparison, but users familiar with the Latin script will have encountered fonts that omit this feature for stylistic reasons. If presented in isolation, the Greek homograph will therefore be treated as an acceptable substitute.

Some letter shapes change with italic fonts, for example the italic “*α*” looks much closer to a Greek alpha (*α*) than the more common glyph used in regular Latin fonts (a).

EXAMPLE: “.*voα*” (Greek) *vs.* “.*voα*” (Latin), both presented as italics

As a final example, here are pairs of more distantly similar labels, but while there might be something "odd looking" about the Greek version to a Latin user, if they are not familiar with the Greek script, they might attribute the difference in this, or similar cases, to a poor choice of font.

EXAMPLE: ".ενοçατίνε" (Greek) *vs.* ".evocative" (Latin)

EXAMPLE: "*.ενοçατίνε*" (Greek) *vs.* "*.evocative*" (Latin), both presented as italics

These examples also show why it may be insufficient to merely look at two labels side-by-side or rendered with the same font/font-style.

The French expression "ça va" might become a label "çava", by dropping out the space, which then is potentially confusable with Greek "çανα". In this example, only the French label represents a term with an independent linguistic identity while the putative Greek one is linguistically implausible, as it starts with a final sigma. But as linguistic accuracy or plausibility are not generally a requirement for domain names, this example may serve to illustrate the point that despite some lack of fidelity, the Greek expression can be confusable.

EXAMPLE: Maçon *vs.* Μαçοη

In French the c cedilla (ç) changes the pronunciation between a hard C and soft C. A city that spells its name with c cedilla will never accept the alternative without in ordinary writing. Yet Montluçon city uses montlucon.com and appears to have done nothing to protect the true name in either .com or .fr.

5.2.2 Minimalist Shapes

There are a number of characters that have very simple glyph shapes, some consisting of a single stroke. Such shapes effectively give no hint of script identity. One example involves a "circle glyph".

Letters involving a "circle glyph" include

- U+006F (o) LATIN SMALL LETTER O
- U+03BF (o) GREEK SMALL LETTER OMICRON
- U+043E (o) CYRILLIC SMALL LETTER O
- U+0585 (o) ARMENIAN SMALL LETTER OH
- U+05E1 (ס) HEBREW LETTER SAMEKH
- U+0B20 (ଠ) ORIYA LETTER TTHA
- U+0D20 (ഠ) MALAYALAM LETTER TTHA
- U+101D (ဝ) MYANMAR LETTER WA

For the Root Zone, these particular examples are defined as blocked variants, as are variants based on characters that look like "c", "i", "s", "v" and "x". In principle, that means they should not need to be considered explicitly in String Similarity Review. However, there are limitations to the way variants are defined, due to the requirement for transitivity. For example, there are many characters that look similar (or identical) to "l", such as U+04C0 (I) CYRILLIC LETTER PALOCHKA, but not all of them could be defined as variants, or transitivity would have resulted in variant relations including some rather less similar looking characters, which is undesirable.

Because very simple shapes do not convey hints as to which script they belong to, labels like ".coco" result in confusability even with scripts where ordinary words generally do not look like Latin.

- .coco Latin
- .coco Cyrillic

- .COCO Myanmar

In addition to the simple shapes, many ASCII characters with not so simple glyphs, like “a”, “n”, “p” and “u” are also variants of characters in other scripts, mostly, Greek, Cyrillic and Armenian. Again, to the extent that this does not result in unduly extensive variant sets due to transitivity, these have been defined as variants in the Root Zone LGR and therefore it is only the edge cases that require resolution in String Similarity Review.

When considering applications consisting only of highly confusable code points, the strong presumption should be to treat the labels as confusable.

5.3 Similar Labels

How to account for accumulation of small infidelities, *vs.* a single larger one?

5.3.1 Confusability and Proximity in Perception Space

Notionally, a label occupies a position in “perception space”, with labels that are similar to each other located in proximity. Two labels that have identical appearance would coincide (whether or not they use the same encoded character string). A threshold for confusability can then be considered as a radius around that position, with two labels considered confusable if the areas thus defined have significant overlap. Where to draw these lines and how to estimate the distance between two labels is ultimately a judgment call. The actual experience of confusability depends on the background of the user and the context in which the labels are presented.

As argued elsewhere, String Similarity has to be evaluated for the whole label, but in the following example we consider single code points for simplicity. At the top of the diagram below there are two letters, Latin small letter eng (ŋ) on the left and an Armenian letter ghad (ղ) on the right, that can be distinguished by the direction of curvature of the tail. They are clearly somewhat similar, but one would need to either make a judgment call or test user reactions to decide how far that similarity crosses into actual confusability.

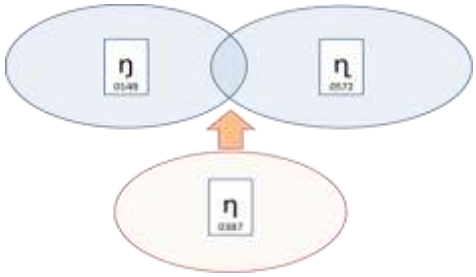


Figure 1 Visual Overlap Connecting Otherwise Distinct Shapes

In the diagram, their visual similarity is indicated by a small overlap in their similarity neighborhood in perception space. They might thus be distinct enough to co-exist in the same zone. However, the situation changes when a third code point is considered, such as the Greek character (eta) at the bottom. Its tail does not curve in either direction, placing the shape in the midpoint between the other two. Each of the similarity neighborhoods for the original two code points now has a significant overlap with that for the Greek small letter eta (ŋ).

In analogy to this example, but applied to whole labels, String Similarity Review would treat both end points as mutually confusable with the midpoint, but perhaps not always require that the endpoints be considered mutually confusable unless a label with the intermediate is already delegated.

For this particular example, the panels developing the Root Zone LGR concluded that the similarities are so strong (or conversely, that the distinctions are so subtle), that all of these code points satisfy the "near equivalent" or near homoglyph condition and therefore all three of these are currently defined as "blocked" variants of each other. This is because leaving out a hook or curve from a glyph shape is a common feature for simplified fonts, such as sans-serif font styles, users familiar with such simplifications could be expected to readily accept the eta as substitute for either the *eng* or the *ghad*.

Another way of putting this, is that two or more labels are often considered visually equivalent (and correspondingly treated as variants) if the maximal separation between any two of them is judged sufficiently small,[11] even if not zero. In contrast, confusability exists even with non-zero distance in perception space.

The characters in this particular example being defined variants, no two of them can be delegated simultaneously in the Root Zone. For other cases where the similarity has been deemed less close between some of the code points, the determination of visual confusability is left to the String Similarity Review. Under that process, it might be possible to allow the outlying members (corresponding to *eng* and *ghad* in this example) to coexist, but not the intermediate one (eta in this example) with either of them. Such flexibility is particularly appropriate where code points (and labels) have different degrees of visual similarity such that some could be considered distinct in the absence of any intermediate forms.

When evaluating labels for confusable similarity, this kind of judgement must be applied.

5.3.2 Example of Uppercase and Similarity

Many letters from the European scripts share uppercase forms, in some cases they are indistinguishable using the same outlines in a given font. Even though IDNs in [IDNA2008] are always lowercase, browsers will accept domain names in uppercase. Thus, a label in the wrong script, contrived to look like an uppercase version of the intended label can be used for spoofing and should thus be rejected in String Similarity Review.

EXAMPLE: .PAY (Latin) *vs.* .PAY (Cyrillic)

For the detailed discussion, see Section 6.

5.4 Distantly Similar Labels

5.4.1 Fake Homographs

There are many scripts that have enough letters with a disparate appearance so that it is possible to find some letters with a certain similarity (however remote) to most or all Latin letters. In this way it is possible to "spell" English words using, for example, Chinese ideographs, or some mix of ideographs and Japanese Kana. The result is legible, but looks quite fanciful. Here is an example[12] that spells out the word "domain-label":

[11] The actual cutoff is a matter of judgement.

[12] The example was generated using the Chinese letters generator - cool text generator (megacooltext.com)

EXAMPLE: 刀口爪丹工れ−ㄥ丹乃モㄥ

While these correspondences are often used in a playful manner, the resemblance is so tenuous that they don't give rise to serious spoofing attempts. They should therefore be categorically excluded from the scope of String Similarity Review.

5.5 Distinct Labels, Not Similar

Finally, there are labels that are easily recognizable and distinct from any other labels. They may be written in a script that does not have any (or very few) code points with shapes resembling those in any other script, or they may prominently contain code points that have a distinct shape, not related to any other shape in the Root Zone Repertoire, whether corresponding to another code point, or approximated by a sequence.

However, for practical purposes, a label only needs to be distinct across the registry; many more potential labels will qualify for that.

6 Effect of Casing on Confusability

6.1 General Considerations on Casing

IDN labels are restricted by IDNA 2008 to be in lowercase, however, user agents (commonly called browsers) will accept labels presented to the user as uppercase or mixed case and then lowercase them for lookup.[13] From the perspective of confusability, it is therefore necessary to consider the uppercase and mixed case forms of labels. This cannot be done using the standard variant specification, because transitivity would create undesirable mutual exclusion.

Not all scripts are cased, but the most widely used ones tend to be European scripts that share a common ancestry and therefore include many common shapes across scripts.[14] In addition, there are a few non-cased scripts for which some letter shapes correspond to uppercase forms of another script and which therefore could be used to spoof the uppercase presentation of a label in the latter.

In many cases the capital forms are exact equivalents of each other, which in most modern fonts would use identical outlines. In a few cases, there are small differences, but resemblance is very close. All of them would normally have been assigned variant status, except that capital letters themselves are not formally part of IDNs. From the point of String Similarity Review, the way labels are registered (as lowercase) matters perhaps less than how they are presented to the user. Most browsers will accept uppercase labels and convert them to lowercase for lookup. Therefore, labels that a user can mistake for the uppercase form of an intended label should not be allowed concurrently in the same zone due to confusability.

6.2 Transitivity Effects

If confusables are treated as transitive, transitivity will effectively "chain" both the uppercase and the lowercase confusable relations. To complicate matters, the overlap among uppercase forms between any two scripts tends to affect different characters than the overlap between lowercase forms.[15] In

---

[13] Labels that consist of ASCII letters, digits and hyphen, the so called LDH labels, are always case insensitive.

[14] The Root Zone does not support any of the bicameral scripts that are not in modern European use.

[15] In some other cases, like 'o', the lowercase corresponds with 'O' across several scripts. In these cases, variant definitions have been defined directly between the lowercase characters.

some cases this results in confusability between two clearly unrelated characters **inside** one of the scripts (or both).

For example, the following illustrates how transitivity of upper/lower case mappings and variant/confusable relation would result in ASCII “h”, “n” and “v” to be considered mutually confusable:

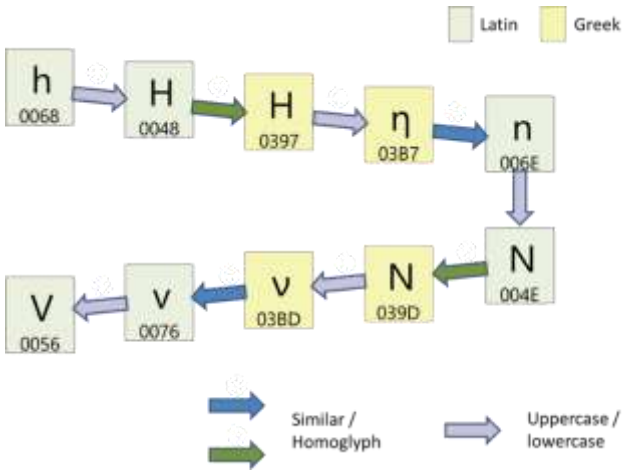


Figure 1 Example of Confusability Based on Similarity of Upper- and Lowercase Forms

While “H” and therefore “h” is clearly confusable with Greek letter eta (H), so are “n” and “N” with its lowercase form (η) (particularly in contexts where users are not familiar with Greek and will accept the “η” as a perhaps fanciful font rendition in a place they would expect “n”; they would not attach a significance to the “leg”, because to them it doesn’t represent a distinction that they need to make in reading). Likewise, “N” maps to the uppercase Greek *nu* (ν) and “v” would be confusable with its lowercase.

This points to the need for considering confusability in a non-transitive manner: labels “hhh” and “nnn” can happily coexist but neither can coexist with a Greek (ηηη), the same is true for “nnn” and “vvv” and Greek (νvν): without the Greek label present, the two Latin labels are clearly distinct and do not cause a problem if they coexist in a registry. But in the case, where the Greek label was delegated first, they both conflict with it and could not be added.

A challenge for string similarity would be to design the process so more labels inside a given script can coexist — as long as there are no labels delegated in the other script that could “bridge” them via combined upper- and lowercase mappings.

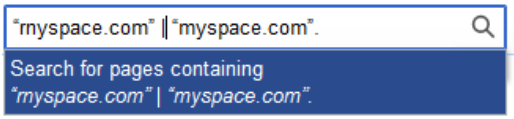
The same concern applies to any level of confusability that is a bit more distant than identical or near identical appearance. Here also, two labels at the end of a chain might safely be delegated simultaneously, as long as there is no existing label that “bridges” the gap.

7 Confusable Sequences

7.1 Accidental Similarity

A typical example of accidental similarity for a sequence is “.corn” *vs.* “.com”. These are hard to predict. In this case, a reader may recognize the “m” primarily by the three stems, and ignore the small break in the arches for the “rn” substitute. There also appear to be no systematic attempts at data collection in this area, and certainly not for ASCII. The “rn” *vs.* “m” example is of some long standing: the fake label “rnyspace” was successfully used to spoof the “myspace” domain name.

EXAMPLE:



Shown in a typical UI font as it would show up in a search bar. (Note the hint in italics is even more confusable).

There are other cases where one sequence can look like another, or where a sequence can look like a single code point. The Root Zone LGR has a number of cases where the similarity is so close that they were defined as variants.

7.2 Repeated Feature

Confusability may exist where visual features are repeated and the user doesn't realize the count is off. This effect can be seen in multiple writing systems, and appears to be related to the way that automatic reading recognition deals with such repetition.

EXAMPLE: Myanmar script ကျုပင် *vs.* ကျ‌ုပင် (left is a typo, right is a name of a "giant reed").

In the preceding example, both strings have the same number of code points, but one looks like it has two vertical strokes instead of three.

This is related to the example given earlier of "rn" vs. "m", where the issue was one of distribution of features across a sequence vs. a single code point (with the small gap not being a sufficiently strong signal to reliably render these distinct in ordinary word recognition).

For a large part, this is a purely visual mis-cue, and not limited to particular words, languages or scripts, but there may additional language or script-specific reasons why users may be more tolerant of the wrong repetition count in certain contexts.

The phenomenon is different from, but related to, differences in the repetition count of letters.

EXAMPLE: "Schifffahrt" vs. "Schiffahrt" (two spellings of German for "maritime industry")

There are many examples where words duplicate or lose a letter when borrowed across languages, which means that many non-contrived examples can be found. However, there is a good argument that cases that relate to *spellings* with different counts of repeated whole letters should be handled consistently with other spelling variations and therefore not be considered in scope for String Similarity Review. (See also Annex A.5 "Non-goal: Plurals and Similar Spelling Variations".)

7.3 Related Letter in Sequence

In many scripts the unit of writing is not the letter, but the syllable. Internal to the syllable, structural rules prevent any free-form arrangement of elements. Syllables can vary in length and complexity, even in the same script. These rules are enforced as part of the LGR mechanism and are limited to enforce structure, not to prevent syllables that are structurally possible but never used.

Sometimes, letters or subsequences have related shapes, akin to similarities like "b" and "d", "l" and "j", or "n" and "m" for ASCII. When letters are combined into syllables the act of forming a cluster may further adjust the shape (akin to ligatures). The result will be that some syllables become less distinct than would be expected by comparing all letters in a sequence individually.

There are a few digraph letters that were not excluded from the Latin script repertoire for the Root Zone. These include “œ” and “æ”; in domain names, fallback representations such as “oe” and “ae” have been common, which makes them confusable with these fallback sequences.

For well-known cases where the rendering is identical or close to it, the variant mechanism in the LGR has taken care of the confusion, but that doesn’t mean that there are no cases of slightly looser similarity that could be used to trick users. Due to the combinatorics involved in comparing all possible sequences (even if limited to well-formed syllables) the research in this area has not been exhaustive.

While the examples given above contain Latin letters, the concerns apply even more to scripts where the basic unit is not the letter, or scripts which form graphical clusters like conjuncts in the Neo Brahmi scripts. Other examples include shapes that aren’t rendered in isolation, such as cursive positional or joint shapes.

[Note to reviewers: the authors would appreciate suggestions of additional, non-Latin examples.]

7.4 Reordering May Affect test for Similarity

Take the Myanmar label “GO” which looks similar to the string “60” or “GO”.[16] It consists of the two code points U+101D (O) and U+1031 (G◌) in that order. U+1031 is a combining mark and as can be seen, in display U+1031 (G◌) is reordered in front of U+101D (O). Therefore, even if one were to look for a correspondence between “G◌” and “G” or “” and “6” and another between “O” with “0” or “o” at the code point level, the similarity and any resulting label confusability would not be spotted. Instead, the whole label needs to be considered, where reordering can affect the placement of a mark. Reordering is common in Neo-Brahmi scripts of South and South East Asia, and whether a mark is intended to reorder in this fashion is documented as part of the Unicode character properties.[17]

8 Scripts and Similarity

The Root Zone supports labels in multiple scripts. The following sections discuss how this affects String Similarity Review.

8.1 Assumption on Script Guidelines

The development of detailed guidelines on detecting confusability depends on an intimate understanding of the typography of the script(s) involved, as well as an insight as to how native readers perform the recognition task. Letters and similar units of writing do not have a fixed shape, but rather may be presented in a variety of shapes, based on the combination of font and rendering system. Native readers are adept at interpreting a given string, resolving it into a word (or, as the case may be, a letter sequence). In doing so, they resolve a series of shapes into a sequence of abstract letters, each of which, of course, corresponds to a Unicode code point in turn.

This process proceeds not by schematic matching of an ideal shape against a collection of pixels, but by having a working knowledge of the expected as well as possible range of variations in the appearance of each letter. This acceptable range of shape variations determines whether the glyph shape corresponding to a different code point is accepted as a substitute by the reader, or not.

The process of reading, or recognition, is highly efficient. Native users expect to see certain letters followed always, or almost always by certain other ones; or they expect a whole word to have an overall

---

[16] The fact that no Root Zone label will have digits can be ignored for the purpose of this discussion.

[17] Unicode character properties are maintained in the Unicode Character Database (https://www.unicode.org/reports/tr44)

shape. Where a confusable alternative meets these expectations, it is unlikely to be detected, even by a reader who isn't careless.

The conditions under which that occurs are thus heavily influenced by the experience of the native reader; any attempt to discern whether two labels are confusable must likewise be informed by such native knowledge.

As a result, guidelines necessarily have a script-specific portion; they also must be created and refined in collaboration with experts that are also native users of the scripts in question.

8.2 Script Dependence

Visual similarity and confusability cannot be considered independent of the script of the intended label. A target user trying to match a label presented to them with the intended label will apply a reading recognition that is trained on the features of the target script, or in many cases, on the features of the subset used by the language.

The typographic and other features of the script influence the task of recognition of a label and discrimination from other labels. Users are trained to automatically look for those features that allow discrimination between written words in their language, and occasionally between specific letters. At the same time, they may ignore features that aren't as important, or ignore small infidelities in a spoofed label as irrelevant for the task of recognizing the expected word or name.

Scripts have different amounts and quality of data on suggested confusables.

8.3 Script Families

Some groups of scripts are closely related by history and features.

- Latin/Greek/Cyrillic/Armenian: common origin, many common shapes
- Chinese, Japanese, Korean: overlapping repertoire, similar features
- Neo-Brahmi[18] Scripts: common origin, similar structure, common shapes

Modern comprehensive UI fonts (such as Segoe UI) show a tendency for harmonized glyphs across scripts, most prominently for scripts with similar typography.

Whether scripts are strongly correlated from the point of visual similarity is not always determined solely by a common historical origin. The following subsections discuss the details.

8.4 Strongly Correlated Scripts

The following script pairs show a particularly strong overlap in letter shapes or they are part of a group of related scripts. They are listed here in descending order of cross-script variants defined in the Root Zone. (Note that cross-script variants are exclusively visual.)

- Chinese/Japanese/Korean (80)
- Kannada/Telugu (34)
- Latin/Cyrillic (28)
- Devanagari/Gurmukhi (26)

[18] The term "Neo-Brahmi" is used in its wider sense here, not limited to the Neo-Brahmi Generation Panel LGRs.

- Latin/Greek (22)
- Armenian/Latin (19)
- Armenian/Greek (12)
- Armenian/Cyrillic (5)

The counts reflect pairs that have been defined as variants. Any visual confusables would be in addition.

- Latin, Greek and Cyrillic share many uppercase forms,

  ⇒ confusable if presented to the user in uppercase

  ⇒ many Latin or ASCII labels can look like Greek or Cyrillic uppercase

- Latin and Cyrillic, and to a lesser degree Greek share many lowercase forms
- Some Cyrillic lowercase looks like “small caps” Latin

  ⇒ Uppercase LDH labels can look like Cyrillic lowercase

- Armenian (rendered in modern font) shares forms with Latin

EXAMPLE: Latin (ASCII) “j” *vs.* Armenian “ȷ” has the same level of confusability as ASCII “i” *vs.* Turkish “ı”

EXAMPLE: Latin (ASCII) “.com” *vs.* Cyrillic “.сом”

EXAMPLE: Latin (ASCII) “MANTA” *vs.* Greek “MANTA”.[19]

EXAMPLE: Latin (ASCII) “USU” *vs.* Armenian “USU”

In contrast to the above, Thai/Lao, while strongly related in structure and origin, typically have distinct font styles, making it easy to recognize which script is used.

8.5 Weakly Correlated Scripts

These scripts have relatively small sets of shapes that are visually correlated; some of them may bear an intrinsic relation, while others may be accidental.

- Han *vs.* Katakana/Hiragana (8 – some shapes have common origin, or simply reflect a typographical relationship)
- Tamil/Malayalam (6)
- Katakana/Hiragana (3 – some shapes have common origins)
- Hangul/Han (3 – some Hangul characters reflect accidental typographical similarities with Han)
- Devanagari/Bengali (2)

The following scripts have a more distant or no relation, but happen to share some similar letter shapes, these are usually instances of shapes with low complexity and therefore few distinct characteristics that would allow identifying a letter as belonging to a different script.

- Hebrew/Latin and Greek (1 – accidental)
- Myanmar/Georgian (1 – accidental)
- Malayalam / Latin (2 – simple shapes)

[19] While the Greek string isn’t a valid IDN, most user agents will lowercase it before lookup.

- Myanmar / Latin (2 – simple shapes)
- Oriya / Malayalam (1 – simple shape)
- Malayalam / Latin (1 – simple shape)
- Armenian / Ethiopic (0 – but several accidental matches with uppercase Armenian)
- Latin / Ethiopic (0 – but several accidental matches with uppercase Latin)

Any counts given above reflect visual variant definitions, any confusables would be in addition.

8.5.1 Scripts without Cross-script Variants

The following scripts don't have any cross-script variant mappings defined in the Root Zone LGR:

- Arabic
- Ethiopic
- Gujarati
- Khmer
- Lao
- Sinhala
- Thai

This does not imply that there is no possibility for confusables. Note, that the Latin Generation Panel in their proposal for a Root Zone LGR concluded that U+12D0 (0) ETHIOPIC SYLLABLE PHARYNGEAL A looks like the letter “o” (or its uppercase). While a single cross-script confusable severely limits the possible cross-script confusable labels, one of them (.ooo) is already delegated. This means that even without cross-script variants, there may be a need for some amount of confusables mitigation.

If uppercase-based confusables are considered, there are a number of Armenian uppercase characters that could be successfully spoofed with Ethiopic syllables. (See example in Section 8.7)

However, if String Similarity Review Panel concludes that some of these share no confusingly similar shapes with another script, they and their labels could be ignored in looking for cross-script confusables.

8.6 Relationships between Writing Systems and between Scripts

The development of most scripts has been influenced by one or more scripts existing at the time; the impetus ranges from a need to find a better match between written and spoken language, to the desire to mark a visible distinction to the way how text in other languages is written. In many cases, the adaptations are merely extensions. Latin, Cyrillic and Arabic are examples of scripts used for dozens, if not hundreds of languages, each using core elements of the script, while adding specific extensions. As an example, note the myriads of Latin-script alphabets developed for languages worldwide.

In many other cases, new scripts were derived in ways that mostly changed the letter forms, but also added some unique ones; for scripts that use conjuncts, such as the scripts derived from Brahmi now sometimes collectively called Neo-Brahmi, the rules of conjunct formation may differ, sometimes substantially.

Where scripts are closely related, many character pairs can be expected to be visually strongly related or similar (whether or not they are used for the same phonetic value or meaning). This allows for many cross-script confusable label pairs, where one label is entirely in one script and the other one entirely in the other.

EXAMPLE: ಜಬ *vs.* జబ (Kannada/Telugu)

EXAMPLE: Kannada/Telugu similarity for the expression: India is a beautiful country[20]

Kannada - ಭಾರತವು ಒಂದು ಅಂದವಾದ ದೇಶ (bharatavu ondu andavaada desha)

Telugu - భారతదేశం ఒక అందమైన దేశం (bharatadesham oka andamaina desham)

Just because some historical connection exists, String Similarity Review is still based only on the modern forms. For some scripts, these have diverged further over time, but for other script pairs, there has been an active development favoring a "harmonized" style that downplays differences in overall appearance and makes it easier to present both scripts in the same user interface or in mixed text.[21]

An example is the recent preference for a less "traditional" style of Armenian letters. The traditional form which has serifs and is slanted, can be seen in the Unicode code charts, while modern fonts in user interfaces tend to be upright and sans-serif.[22]

EXAMPLE: Armenian Հայաստան (traditional) *vs.* Հայաստան (modern, sans-serif)

EXAMPLE: Greek αλφάβητο (traditional serif) *vs.* αλφάβητο (modern, sans-serif)

EXAMPLE: Greek phi: φ *vs.* ϕ (alternate forms, the latter considered calligraphic)

The consequence for String Similarity Review is the need to pay attention to letter shapes preferred for on-screen reading or user interface elements, such as address bars, but with some awareness of whether native users might accept something that looks like an alternate shape that they are familiar with, but that would ordinarily not occur in the context of a modern or user interface font. Target users are not assumed to be aware of the fine points of typography.

8.7 Accidental Similarities with an Unrelated Script

A number of scripts that are unrelated may nevertheless have shapes that happen to correspond rather closely and that would allow some labels in one script to be spoofed in another script. Ethiopic is not usually considered related to the European scripts, but has some shapes that are confusable; see also 5.2.2.

EXAMPLE: በበበ (Ethiopic) *vs.* በበበ (Armenian)

EXAMPLE: ሀሀሀ (Ethiopic) *vs.* UUU (Latin) *vs.* UUU(Armenian)

These particular examples show uppercase for the Armenian and Latin script parts. Because user agents would accept these strings even if they are later converted on the fly for lookup, a user presented with the Ethiopic label .ሀሀሀ would not know that it isn't a match to Latin .uuu in a context where Latin .UUU would be a possible presentation.

Among these, there are several mappings from ASCII to "simple shapes" in unrelated scripts:

---

20 Source: https://dravidianlanguages.quora.com/Is-Kannada-more-similar-to-Telugu-or-Tamil

21 An example of the latter is the Meyrio font, see https://learn.microsoft.com/en-us/typography/font-list/meiryo#Overview which was created to harmonize traditional Japanese with Latin/Roman for improved mixed text layout and on-screen legibility.

22 See https://news.am/arm/ for examples of two modern styles (serif and sans serif), not slanted. Omniglot in https://www.omniglot.com/writing/armenian.htm shows the less common non-slanted traditional style, barely in use these days.

- “c” shape mapping to Myanmar
- “i” shape mapping to Hebrew
- "o" shape mapping to Hebrew, Oriya, Malayalam and Myanmar
- “s” shape mapping to Malayalam

These may have to be considered in the context of any label confusable with a label made up only of simple shapes, see 5.2.2 “Minimalist Shapes”.

PART II – STRING SIMILARITY REVIEW PROCESS

9 Scalability of String Similarity Review

Any label can potentially have a quasi-infinite number of blocked variants and an even larger number of confusables. Each variant can have its own confusable cluster, because not all variants are visual, so their confusables clusters do not necessarily overlap in visual perception space. This creates an interesting challenge: in the general case it is not feasible to exhaustively enumerate either variants or confusables within reasonable bounds of space and time. A different approach is therefore required as discussed in the remainder of this section.

9.1 Size of the Registry

In discussing possible approaches to String Similarity, it is necessary to make assumptions about the size of the registry. For a registry with a few hundred to a few thousand delegated labels, manual review after pre-screening is feasible; for larger registries the picture may be different and the same procedures might be prohibitive.

If delegated labels are distributed relatively uniformly over the various scripts, the chance of collisions is reduced as many more labels can be expected to have a letter or sequence that has a very script specific appearance and is otherwise unique.

If a single script or a group of related scripts dominate the registrations, the density of labels in that area of perception space increases, making some of the approaches discussed here less efficient already at a lower number of total registrations for the registry.

For the foreseeable future, the size of the Root Zone will be small enough to support the strategies discussed below. As currently delegated, LDH labels dominate the Root Zone and most of them correspond to English words [ROOTDB].

9.2 Scalability of String Similarity Review

As a registry reaches a certain size, comparing a candidate label for string similarity against all delegated labels (and their variants) by brute force becomes infeasible. The same is true for checking whether a candidate label is a possible variant of any existing label, and lessons learned from variant detection may be applicable to String Similarity Review.

Experience with defining variants (including in-script and whole-script labels of identical or practically identical appearance) has shown that even moderately complex labels can generate a large number of variant labels; the number can easily be in the hundreds or thousands, or, in some cases, more than can be enumerated with reasonable effort. Any attempt of comparing all the variants of one label against those of another[23] is impracticable for the general case and infeasible for any registry of more than the most minimal size.

By definition, variant relations are equivalence relations and therefore transitive. That supports the definition of an Index Variant Label or Index Label. (This is sometimes also called a “skeleton”[24]).

---

[23] Because variants may be other than visual, a review for visual similarity would have to compare not only the label but all its variants.

[24] See Unicode UTS#39 [UTS#39] and Unicode UTS#55 [UTS#55].

Index (Variant) Label (or also "skeleton"): a label computed from any label such that all labels that are defined as variants of each other result in the same Index Variant, and no label not a member of this set of labels that are mutually variants of each other will result in the same Index Variant.

All labels that are mutual variants generate the same index, so all that is required for collision testing is a single index variant computation per candidate label, which is an O(1) or constant time operation. Index labels for all other labels already in the registry are assumed to be pre-calculated; the comparison for collision testing is then equivalent to a search, or O(log n) operation.[25]

The calculation of an Index Label is logically equivalent to defining a variant label set, with all labels that generate the same Index Label as its members. If each label can only generate a single index label, and all its variants (if enumerated) also generate the same index label, all of the variant sets are disjoint from each other.[26]

Just like an index variant label uniquely identifies which variant set a label belongs to, the same approach could be used to identify a confusables cluster. Unlike the case for variants, which are inherently transitive, applying this approach to confusables has the downside that this generates many false positives: not all labels inside a confusables cluster are inherently confusable with each other; see 5.3.1 "Confusability and Proximity in Perception Space" and also compare the discussion of 5.3.2 "Example of Uppercase and Similarity".

9.3 Pre-Screening

However, even if confusability is not inherently a transitive relation, defining certain types of confusables as if they were variants might be a useful tool for pre-screening. Testing for variant collisions is fast, and the number of false positives when comparing a candidate or test label against a set of existing labels may be small enough to allow more computationally expensive or even manual review methodologies to resolve such potential confusability in a second pass.

Whether and how well this approach works depends ultimately on how confusables cluster in perception space. It would not be surprising to find some difference in behavior for different scripts or script families. Under the assumption that actual clustering allows a reasonable heuristic, it would return more labels as confusable than the actual set, but hopefully a small enough set that pruning in a manual process by a String Similarity Review panel is feasible.

Pre-Screening: Goals
Reduce the number of labels to be verified in detail.
- ⇨ Filter out any label pairs that are dissimilar enough to not be confusable,
- ⇨ Automated, data-driven process to be scalable.

A possible implementation of a pre-screening process will attempt to leverage a modified variant mechanism to quickly identify sets of potentially confusable labels. Unlike variants, these sets would **not**

[25] https://en.wikipedia.org/wiki/Big_O_notation

[26] This restriction is the same as requiring variant *labels* to be transitive. Note that defining transitive variants on the *code point* level does not automatically guarantee that the enumerated variant *labels* obey this restriction in certain special cases. However, for the Root Zone, the definition of code point variants was adjusted to satisfy this constraint and all enumerated variant labels should resolve to the same Index Label.

consist only of mutually confusable labels. While each label would be confusable with at least some labels in the set may include some that shouldn't be considered confusable.

In other words, the process would use a heuristic that treats confusables as superset of variants ("loose variants"). Because, unlike variants, this relationship is not akin to an equivalence, it is initially not a "yes/no" question, but a question of degree. An approach would consist of creating a simplistic score that allows labels to be ranked "quick and dirty") by suspected similarity based on letter similarities on a five-point scale: Dissimilar, Weak, Similar, Strong, Identical, where the end points represent absolutes.

9.4 What kind of DATA are available

The following summarizes the sources for data available to seed a pre-screening process. For citation of specific sources, see Section 16.1 "Data".

- Some data are available for Latin/Greek/Cyrillic/Armenian scripts (single code points)
    - Byproduct of Root Zone LGR development
    - Additional data from Registry policies [EURID]
- Unicode confusables data [UTS#39]
    - Current status not consistently maintained
    - Future direction being reviewed
    - Need vetting and alignment with Root Zone
- Data for sequences are not generally available
    - ⇒ Handicap for scripts that write in "clusters" (Neo Brahmi)
    - Sequences not limited to Neo-Brahmi (.corn vs .com)
    - Panel can review labels in addition to those flagged in pre-screening
- String Similarity Review will develop contention sets based on confusable similarity

10 String Similarity Review Process

String Similarity Review is only one part in processing applied-for labels. The table summarizes the possible outcomes of the String Similarity Review itself, so the status assigned to a label are not final, but relevant to that part of the process.

Status	Description
Ineligible	The label fails to meet the validity criteria for a label to enter String Similarity Review.
Pass	The label is not confusable with any of the labels it has been reviewed against, and is a candidate for delegation subject to all later stages of the application process. Or, the label is confusable only with labels delegated to or applied for by the same entity and delegation is permitted in the relevant policy.
Contention	The label is confusable only with another applied-for label. It will be placed into a contention set. Other stages of the application process will resolve its final status
Fail	The label is confusable with any existing or reserved label and no longer considered a candidate for delegation.

10.1 Steps in String Similarity Review

The following summarizes the steps in String Similarity Review.

1. *Verify Prerequisites.*

   Verify that the label is valid and either a primary label or an allocatable variant as determined by applying the RZ LGR according to Section 5.4, "Steps in Processing a Label" in [RZ-LGR-Overview]. Also verify that it is not a reserved label [Reserved] or otherwise ineligible. A label that meets the prerequisites is a "Test Label".[27]

   Note that variants are considered independently of the primary label. Variants may be defined on the basis of semantic of phonetic equivalence, as the case may be, and can be visually distinct; they therefore must be independently reviewed for string similarity. Blocked variants are eliminated during the preceding stages of label processing and thus ineligible for String Similarity Review.

   If the label is ineligible, stop the String Similarity Review and reject the label as not reviewable and therefore not a candidate for delegation.

2. *Pre-screen the Test Label to establish that it is potentially confusable.*

   Labels that contain one or more highly distinct letters or sequences are unlikely to be confusable with other labels. "Distinct" here means a shape that is not similar to any other shape in the repertoire. "Sequence" allows both for combining sequences and similar clusters native to a writing system, but also sequences that could be "accidentally" similar to each other or to some letter.

   A label from a script that has no variants or pre-determined confusable code points with any other delegated script would be classed as not potentially confusable.

   This scan can be automated based on data prepared beforehand.

   If the label is not potentially confusable, skip to step 4.

3. *Pre-Screen for existing labels that are potentially confusable with the Test Label.*

   Mechanically evaluate the label against existing labels to select all labels that can be predicted to be potentially confusable with it as targets for review. In the following, a "Review Target" is any label that is being further evaluated for confusability against the Test Label.

   In pre-screening, any existing label that contains a letter or sequence that is uniquely distinct and therefore would fail a test like the one performed in the preceding step for the Test Label would be considered not potentially confusable with any Test Label and therefore not a potential Review Target. The list of existing labels to be excluded for this reason can be computed beforehand.

   As much as possible, the pre-screening process should be optimized to avoid or streamline reviewing delegated labels that cannot be confusable with the Test Label, for example because there is no confusability between the two scripts.

---

[27] Sometimes also referred to as "candidate".

Note that variants can be defined based on criteria other than visual similarity. Such variants still need to be considered in String Similarity Review:[28] Any label that can be confused with a (delegated or not) semantic variant of a label is just as much a confusable as one that can be confused with the label itself. Confusability does not depend on visual side-by-side comparison, but on deciding whether a presented string matches the intended label. Therefore, code points that have variants, even non-visual *variants* should not be considered "distinct" in this step.

The remaining labels are evaluated using appropriate automated methodologies that reject as Review Targets only labels that are clearly distinct from the Test Label. The data for this scan can be prepared beforehand.

When selecting Review Targets, all delegated, allocated, reserved and (eligible) applied for labels except the Test Label itself should be considered. The purpose of this step is to remove clearly distinct labels from the pool of potential Review Targets.

The result of this step is a shortlist of labels for further review against the Test Label.

4. *Add manually requested Review Targets.*
   This step ensures that any labels that are suspected of being confusable, or that have been contested on grounds of visual similarity can be evaluated even if they fail the pre-screening.[29]

   Note that this step allows identification as Review Target even for a Test Label that is considered "distinct" in pre-screening.

   If the set of Review Targets is empty, stop the String Similarity Review and resolve the label as "pass".

5. *Review the Test Label against all Review Targets*
   Apply the appropriate review methodologies, whether manual or automated, to each Review Target in turn to establish whether it is in fact confusable with the Test Label.

   If one of the Review Targets is actually confusable with the Test Label, proceed as follows:

   a. *If confusable labels may not be delegated to the same entity, stop the String Similarity Review and reject the label.*
   b. *Otherwise, continue the current review step for all other Review Targets.*
      i. *If any additional Review Target is actually confusable with the Test Label, and the Review Target is not delegated to the same entity as that applying for the Test Label, stop the String Similarity Review and resolve the label as "fail".*
      ii. *If all confusable Review Targets are applied for by, or delegated to the same entity, resolve the label as "pass".*

---

[28] In addition, variants are defined on the code point level. Labels may have both variant and confusable mappings for some code positions in the label These are handled by treating code point variants comparable to confusables with identical visual appearance for the purpose of String Similarity Review.

[29] For example, for any candidate label that is short enough and that visually might match a 2-letter ASCII label, the full set of such 2-letter labels could be added to ensure that none of them, whether delegated or not, is confusable with the candidate label.

If none of the Review Targets are actually confusable with the Test Label, resolve the label as "pass".

6. *Further Processing of Test Labels*
   After ascertaining whether status of a Test Label (and its variants) is confusable against other applied-for labels (and their variants), or against delegated labels (and their variants), the label is ready to be processed based on other provisions in the policy.

   a. *Process labels that are resolved as "pass" according to other steps in the policy.*
   b. *Place applied for labels that are rejected as mutually confusable with another candidate label in a contention set, to be resolved according to policy. The same applies if its variants are confusable with a delegated label or its variants.*
   c. *However, if a candidate label or its variants are confusable with a delegated label or reserved word or its variants, do not process the label further and resolve the label as "fail".*

Some potentially useful screening heuristics are discussed elsewhere; see Section 11.

11 Developing Screening Heuristics

It is not generally possible to compute the similarity or confusability of arbitrary labels based on their code point sequence. Nor is it possible to assign coordinates that define a context independent location of a label in perception space. Even if one of these problems could be addressed, the results might not correlate to what a panel of users would report when considering an actual label when compared to an intended label. Their results would be influenced by the way individual users interpret the context and the background that they bring to the task.

Nevertheless, it might be possible to construct some heuristics that could allow filtering out those labels that are unlikely, or less likely to be confusable with a target label.

If we assume that similar labels are clustered in perception space, and that clusters are reasonably disjoint, then the task narrows down to identifying all labels belonging to the same cluster as the target label. That is similar to the way variants divide the label space into disjoint variant label sets. Where the latter consist of elements that are all variants of each other, the members of a cluster are not equally similar or confusable. In other words, while the variant relation is an equivalence relation and therefore transitive, similarity is not.

If you visualize perception space as a soup of all possible labels, then the ability to pre-screen depends on how "chunky" that soup turns out to be. A fully homogeneous soup would provide no suggested cutoff. On the other hand, a very lumpy soup suggests that similar labels cluster together, but that there is some good separation between dissimilar labels. Screening heuristics that place a cutoff somewhere at the edge of a typical clump would collect mostly labels that are more or less strongly similar; perhaps with labels placed on opposite sides of a clump being distinct from each other, something that would need to be determined in a more detailed review.

If, on the other hand, the actual perception space turned out to be largely homogenous, such heuristic screening would largely fail, and no adjustment of screening parameters is likely to improve performance. The actual composition of perception space is not known or knowable; what can be determined is whether pre-screening is successful in reducing the number of potential review targets.

Once we have identified a cluster, we need another heuristic to help us estimate the expected strength of the similarity or confusability. Because an exact value is not available, the goal should be something like a rough classification into these categories:

1. Identical or Near Identical Labels (homograph)
2. Highly Confusable Labels (strongly similar, or near homograph)
3. Similar Labels
4. Distantly Similar Labels (weakly similar)
5. Distinct, not similar

The simplest scheme for scoring two labels in the same cluster would be to assign as score the highest value in the list above. If any two labels have a dissimilar code point in the same position, the labels would be scored "dissimilar". And so on. To mimic the way variants are defined, "identical" should be taken to also cover "practically identical" appearance as defined by "blocked" variants in the Root Zone LGR as used in collision testing.[30]

Assuming that clusters are disjoint, it should be possible to calculate a unique cluster index for that can be calculated for each label.

The variant mechanism used in collision testing represents just such a classification, except it only has the values:

1. no variant
2. blocked variant

However, each variant mapping has an explicit variant type which is in principle not limited to "blocked"; these variant types can be freely assigned. If the ways in which labels are similar allows cluster-based heuristics, it would be possible to express the degree of similarity by overloading the variant type with the similarity categories.

A heuristic defined along these lines can be implemented with the LGR mechanism of [RFC 7940] and after suitable tables have been created, the labels can be evaluated using existing tooling.

11.1 Steps in Heuristic Pre-Screening for Similarity

11.1.1 Preparatory Steps

A. *Create a similarity map on the code point level using the LGR format[31]:*
   a. Code points that are not dissimilar are given a "var" mapping with a "type" value from the list above.
   b. Dissimilar code points are not given a "var" mapping.
   c. All code points are given an additional reflexive variant of type "r-identical" which reflects a code point that is the same between two labels in a label pair.

[30] Another way of calculating the "degree of similarity" between two labels might be as the average of the similarity score of each of the constituent code point mappings used to get from one to the other. A value of 5.00 would indicate a homograph label, while 1.00 would indicate two entirely distinct labels. This might better account for labels with many moderate differences, but at the cost of needing modified tooling that implements that algorithm, whereas the simplistic form can be realized using a rule set in the [RFC 7940] format.

[31] This step would draw on existing data collections, where identified, see [TBD].

B. *Create "actions" that correspond to the classification in step 3 of Processing steps below*.
C. *Collect a list of all delegated labels and reserved words [Reserved].*
    a. Add 2-letter ASCII labels as required.
    b. Include all variants[32].
D. *Compute a cluster index for all labels already delegated (or otherwise to be evaluated against the candidate label).*

11.1.2 Processing Steps

1. *Calculate the cluster index for the target label.*
2. *Collect all existing labels belonging to the same cluster*.
    a. If the set is empty, the target label is likely not confusable.
3. *Compare each label to the target label.*
    a. For each code point, identify the mapping that would lead from target to the existing label.
    b. If any similarity type is below a threshold, e.g., "dissimilar" or "weakly similar", stop; the label is likely not confusable with the target label because it has sufficient unique elements.
    c. If no similarity type is above another threshold, e.g., "strongly similar" or "identical/r-identical", stop; the label is likely not confusable with the target label because no part is strongly similar.
    d. If all similarity types are above the "strongly similar" threshold, consider the label to be likely confusable with the target label.[33]
    e. Otherwise, the label should be evaluated further.
4. *For any label that has not been resolved in step 3, perform manual evaluation[34] against the target label.*

If the evaluation in step 3 is done by calculating dispositions using the method defined in [RFC 7940] then the only options are "any", "all" or "none". However, if a modified LGR tool can be used, it would be possible to also base the score on the number that each type is occurring. Practical experience would show whether this refinement is productive.

11.2 Expected Performance

Cluster index calculation is an O(1) process. Assuming the data on existing labels is sorted by index value, collecting the cluster becomes an O(log *n*) process, where *n* is the number of labels in the registry. It will be a "win" if the ratio *m:n* is small, where *m* is the number of members in the cluster.

---

32 If the pre-screening data are properly designed, it is not necessary to actually assemble a list of all potential variants. The pre-screening process is supposed to subsume this implicitly (not least because doing it by direct enumeration may be computationally infeasible).

33 The regular LGR process should have covered the case where all code points in the label are either the same or practically identical to the target label. Such cases should have been defined as blocked variants for collision testing, and any label affected should have been rejected. For String Similarity Review, we only care about cases where practically identical code points occur mixed with not so identical ones.

34 For Manual Evaluation see Section 12, "Manual Evaluation (Post-screening Process)".

The screening of labels in the cluster is an O(*m*) process, and it will be a win if the ratio *r:m* is small, where *r* is the number of labels needing detailed review.

Practical experience will show whether these ratios are small enough to make this heuristic useful.

11.3 Alternative Metrics for Similarity Distance

Instead of a simple similarity cutoff, the use of some other, more detailed metric for String Similarity may be motivated. A number of such metrics exist for the case that two strings are similar because they are close in spelling. Several of them are modifications of the Levenstein distance[35].

The idea behind them is to define a number of basic editing algorithms such as substitution, insertion or deletion that, if carried out in succession, morph one string into another. Here we consider only direct operations, not chains. Each operation can be given a weight, either globally based on the type of operation, or based specifically on the source and target. The value of the distance depends both on the weights associated with each operation and how many of them were needed across the label.

1. *Given a set of confusable mappings[36] that exhibit clustering behavior, first determine all existing labels that are in the same confusables cluster as the target label.*

   This will include many labels that may not be very similar. To prioritize review, a metric is needed identifying those that are most closely similar (that is most likely to be confusable).

2. *For each label in the cluster, compute a distance metric.*
   a) The metric assigns a numeric value to each type of mapping, using 0 for a mapping to a variant, or identical appearance, and higher values for more distant similarity. The absence of a mapping is given some high value that should, however, not be infinity.[37]
   b) Proceeding from the start of the label, each code point or sequence is tested for available confusables mapping and apply any that will lead to the corresponding code point in the candidate label.
   c) The value corresponding to each mapping or absence of mapping is collected.
   d) Finally, the raw distance is suitably scaled. For example, the sum of the distance values can be divided by the number of code points in the label to obtain the average distance per code point.

3. *The distance is calculated for each possible partition of a label.[38] For multiple partitions take the lowest distance.*

4. *Define two cutoff points.*
   a) Any two labels below the first cutoff are considered confusable, and are added to the contention set or rejected.
   b) Any two labels above the second cutoff are considered distinct, and are allowed.

---

35 Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other [Levensthein].

36 We treat variant mappings as confusables with maximal level of confusability.

37 A single distinct letter may be overlooked, especially if part of a long label. Therefore, the value should be set low enough that such a case potentially falls below the confusability threshold.

38 The issue of partitions is discussed in [RZ-LGR-Overview].

c) Apply some level of manual verification to ensure the cutoffs are correct and the results are as intended. This may take the form of statistical sampling.

5. *Finally, apply manual review to all labels that fall in between the two cutoffs.*
   a) If the two labels appear confusable on visual inspection reject the candidate (or, if the review label was itself an applied for label, place both into a contention set).
   b) Otherwise, because of semantic and phonetic variants as well as uppercase-based confusables, two labels that are confusable may not immediately be similar in appearance.
   c) In those cases, review requires considering the actual mappings that led to the lowest distance score. If the review indicates that one of them may be similar in appearance to a variant of the other, reject the candidate label (or, if the review label was itself an applied for label, place both into a contention set).
   d) Otherwise, accept the candidate label for further processing.

Caution must be applied when deletion of one element of a sequence is used as one of the operations. Experience has shown that this can lead to unanticipated consequence in the presence of confusable or variant mappings involving other sequences, unless the process is carefully curated.

11.3.1 Detecting Dissimilar labels

Using the same data set, scan all labels for the presence of code points that have no confusables or variants. Such labels should be "dissimilar" compared to all labels. If the target label is "dissimilar" no further evaluation should be needed, if any of the existing labels are "dissimilar" they can be excluded.

This is an O(*n*) process.

Practical experience will show whether such a heuristic reduces the number of labels that can be ruled out as most likely not confusable to a number that is small enough to not need further heuristics

11.4 Effect of Confusable Sequences on Heuristics

Individual code points can be confusable with code point sequences (such as "rn" *vs.* "m" in the example ".corn" *vs.* ".com"). Sequences may likewise be confusable with other sequences, including sequences of different length. This undercuts the ability to simply rule out existing labels just because they are not the same length as the target label.

Confusable sequences are not limited to sequences already defined as repertoire elements in the Root Zone LGR. However, many sequences defined as repertoire elements do have one or more counterparts of identical or practically identical appearance defined as variants. They add to the number of potentially confusable labels.

Sequences may overlap. A sequence may be confusable with another sequence, while a part of the sequence may be confusable with something else.[39]

EXAMPLE: n *vs.* n n̄ *vs.* ñ

0578 006E 006E 0304 00F1

[39] The examples given here happen to be defined as variants in the Root Zone, but analogous cases would apply to confusables. The alternation between macron above and tilde above is treated as variant in the Root Zone because at small sizes typical in an address bar the distinction practically disappears.

In this example, the Armenian character U+0578 does not form a sequence with U+0304 *macron* nor does it form a sequence with a combining *tilde* or any other confusable diacritic. However, there are cases where such overlaps result in additional labels that are potentially confusable.

A sequence may be confusable with part of itself. In the case of variants, we call such a case an "effective null variant" because effectively the mapping "nulls" out some code points in the sequence. An example would be treating a combining sequence with diacritic as confusable with the base character alone (nulling out the combining mark).[40]

EXAMPLE: आ *vs.* आ़

0906 0906 093C

These cases lead to similar complication as they do for variants. For an in-depth discussion see 6.6 "Overlapped Variants" in [RZ-LGR-Overview].

Some code points reorder in presentation. In those cases, the entire sequence that is subject to reordering has to be mapped to any code point or sequence that is confusable with it in the reordered presentation (for an example see 7.4 "Reordering May Affect test for Similarity").

Existing data collections on potential confusables like those defined in Unicode UTS#39 [UTS#39] often exclude sequences altogether, or limit them to the 1:*m* case (a single code point being confusable with a sequence of length *m*).

12 Manual Evaluation (Post-screening Process)

The goal of the manual, post-screening process is to come to a yes/no answer to the question "is this candidate label confusable with the review target?". Unlike the pre-screening process, there is no algorithm to deterministically arrive at the correct answer.

The process will start with the automated calculation of a similarity / confusability score determined using the tools and data tables used in pre-screening. It may be necessary to review how the particular score was calculated to verify that it is indeed meaningful as is, or, if necessary, adjust it.

Built into the data tables are some of the considerations weighing in favor or against a presumption of confusability. At this stage, additional considerations may be considered. These may be based on the particular script in question and drawn from the information in the Appendices below, or based in additional input from native users or experts.

Some candidate labels may pose idiosyncratic or ad-hoc issues, these need to be resolved taking into account the overall spirit of these guidelines and the goal of weighing the risk to the security of the DNS against the utility of allowing a particular label. This may require setting aside any computed confusability scores specific to that label.

In keeping with RFC 6912, the overall approach should be one of conservatism. If in doubt, the ruling should be in favor of declaring confusability, particularly if not doing so would create a risk to the DNS.

---

[40] A nukta placed below a vowel occurs in a context not anticipated by a majority of users of the script and therefore it is likely to be treated as if it wasn't there; for that reason, the Root Zone defines a variant for this case, but analogous cases apply for confusables.

Decision by consensus is preferred, particularly when overriding or adjusting down an automated score showing high probability of confusability.

While the String Similarity Review Panel may consult experts and native users, the use of formal focus groups is not required or contemplated by these guidelines. Any review decision requires the contribution of at least three reviewers, each of which has at least some familiarity with the script. At least one reviewer must have strong familiarity with the script and for an even numbered panel, that reviewer is the tie breaker.

All aspects of a decision are to be documented, including the analysis of the score as well as factors weighed and how the panel applied them in arriving at the decision.

If manual review results in a verdict of confusability with a delegated label or reserved word including their variants the label is rejected. If the confusability is against another applied for label, they are placed in a contention set. Otherwise, the label is accepted for further processing.

12.1 Summary of Post-Screening Process

- Steps in manual evaluation
    - Input: set of review labels with scores from pre-screening (levels 1-5 integer, or decimal)
    - Task: to verify score from pre-screening
    - Reach a decision confusable similarity yes/no? (May be based on adjusted score)
    - Apply from the list of considerations to reach decision (e.g. upper case, italic alternates, homoglyphs—those that are ruled in scope for the script)
    - Suggested types of considerations are provided in this study (including script guidelines)
        - Allow for novel considerations discovered ad-hoc based on specific nature of candidate label
- Final decision left to String Similarity Review panel
    - Panel may consult script experts
    - Panel will not use focus groups
- Procedural aspects
    - Minimum number of reviewers (3)
    - Minimum qualification (all have to have -some- familiarity with script, at least one strong and gets to be tie breaker)
    - Take into account risk to security of DNS
    - Ability to override computed score indicating probable confusability
        - Where actual similarity seems too tenuous to support confusability
        - Where actual confusability is deemed risky for DNS despite score
- Outcomes
    - Rejected if confusable with existing or pre-existing labels
    - Added to contention set, if confusable with another applied for label or variant
    - Otherwise: Accepted for further processing
- Documentation
    - Identify label and outcome
    - Document which labels were found to be confusable with the target label
    - Document the type (delegated, reserved, etc.) of confusable labels found
    - Account for considerations used in reaching decision
    - Record the decisions and number of experts publishing

12.2 Criteria for Post-Evaluation of Labels for Confusability

The criteria to be used in this evaluation are found throughout this document, including the annex relevant to the script in question. When reviewing a candidate label against a review target of a different script, considerations for both scripts need to be applied in turn. Once delegated, confusability goes both directions.

13 Next Steps

- Public Review of Study Report
- Collect Initial Data Set
- Trial-run of pre-screening on sample labels
- Refine Data Sets and Metrics
- Community input on test result and finalized methodology

14 Contributors

Editors

Asmus Freytag and Michel Suignard

ICANN Staff

Francisco Arias

Valerie Heng

Sarmad Hussain

Michael Karakash

Pitinan Kooarmonpatana

Yin May Oo

PART III SUPPLEMENTARY

15 Definitions

Allocatable Variant: *(see Variant Disposition).*

Blocked Variant: *(see Variant Disposition).*

Candidate Label: the label being reviewed for string similarity, candidate for short.

Confusability: the degree to which two labels can be substituted for each other without that fact being noticed. Users are confused about which label they are selecting.

Confusable: said of two labels for which the degree of confusability exceeds defined threshold. Here also used as a noun to indicate the relation between a pair of labels or code points where one is confusable with the other. The second meaning is used in analogy and contrast to *variant*.

Confusables Cluster: based on the idea that confusables for a certain label cluster around it in *perception space*. For code points, a confusables cluster represents all code points or sequences that are directly or by transitivity confusable with a given code point or sequence.

Confusable Similarity: the degree of similarity required to make two similar labels “confusable”.

Cross-repertoire: two code points in the same script that is shared by one or more languages and either one or both are not part of the shared subrepertoire.

Cross-script: two labels, entirely from different scripts. In selected cases, such labels may look identical, or confusingly similar.

Delegated Label: a label already in the registry. For the purpose of String Similarity Review, reserved words are also treated as if they are delegated.

In-repertoire: said of two code points that are part of the repertoire for the same language or script. A label made up entirely of such code points.

Intended label: the label the user wants to reach, and which the user is trying to match with the actual label presented. (See *recognition task*).

Original label: Here: the label which the variant label is a variant of. Formally: the label to which variant mappings are applied to reach the variant label. (See also *Primary Label).*

Primary label: term used in IDN EPDP policy document for the applied-for label against which the variant labels and their dispositions are calculated. (See *Original Label*).

Perception Space: a notional space in which each label has a location and the distance between two labels corresponds to the degree of similarity.

Recognition Context: the context in which the *Target User* is asked to distinguish between the intended label and some alternative. Here: a Target User recognizing a label presented in isolation as the intended label, without direct visual reference to the “correct” representation of the latter.

Recognition Task: the attempt to decide from visual inspection of an actual label, whether it is a valid representation of the intended label, an unrelated label, or an impostor.

Related Script: a script that shares common ancestry with some script. This often, but not always, leads to a large number of shared forms. If the overlap is large, the scripts are "strongly correlated".

Reserved words: labels that are by policy not available for delegation as part of the application process. (See [Reserved]).

Review Target: a label that is shortlisted for further review against the *candidate label*. When evaluating the contention between two or more applied for labels, each one is in turn considered the candidate label while the others are treated temporarily as a review target.

Script Family: a set of related scripts.

String Similarity: similarity in appearance, of a kind that contributes to the confusability between two labels. Here: visual similarity that may depend on actual rendering of the label (including font choice) and on the experience and background of the user doing the recognition task.

Strongly Correlated Script: a script that has a substantial overlap in forms with another script, resulting in the potential for many pairs of confusable labels.

Target User: typical user in terms of which *confusability* is defined. Here: a reasonably attentive user, natively familiar with script or language, but also with ASCII subset.

Target Context: typical recognition context in which a user is presented with a label and needs to decide whether it is the intended one. Here: the context includes a user being presented with label in foreign script or language when looking for a native label but excludes a user deciding between two labels in a foreign script or language.

Test Label: see *Candidate Label*.

Unique Substring: part of a label with a distinct appearance, not likely to be confused with any other.

Variant Disposition: whether a variant is allocatable (may be delegated to the same entity), or blocked (may not be delegated).

Variant Label: a label that a user will accept as a substitution, and consider it "same as" the original label. Variants based on visual similarity have identical or near identical appearance, but variants are not limited to visual similarity. Particularly for CJK scripts, semantic variants are common (characters of the same meaning), but there are also phonetic variants, such as Ethiopic characters of same pronunciation used interchangeably.

Variant Relation: whether two labels are variants of each other. The Variant Relation is an equivalence relation, and therefore transitive, unlike general confusability and similarity.

Weakly Correlated Script: a script that has minor, or accidental overlap in forms with another script.

Whole Label: used in contrast to processing a label code point-by-code point.

16 References

Feature contingencies when reading letter strings

Daniel R. Coates, Jean-Baptiste Bernard, Susana T.L. Chung, Feature contingencies when reading letter strings, Vision Research, Volume 156, 2019, Pages 84-95, ISSN 0042-6989, https://www.sciencedirect.com/science/article/pii/S0042698919300100

Extended Process Similarity Review Panel

[EPSRP] "Extended Process Similarity Review Panel" (EPSRP), ICANN, 2014

(https://www.icann.org/en/system/files/files/epsrp-european-union-30sep14-en.pdf).

Root Zone LGR Documentation

[RZ-LGR-Overview] Integration Panel, "Root Zone Label Generation Rules - LGR-5: Overview and Summary", 26 May 2022 (PDF), https://www.icann.org/sites/default/files/lgr/rz-lgr-5-overview-26may22-en.pdf

[Survey] Visual Variants Study for Japanese Root Zone LGR, Appendix B and C in https://www.icann.org/en/system/files/files/proposal-japanese-lgr-appendices-20dec21-en.zip

Examples of variants, confusables, IDNs and EAI mail addresses

[UTW] Sarmad Hussain, "Internationalizing the Internet's Domain Name System (DNS): Root Zone Label Generation Rules", Unicode Technical Workshop Presentation, 7 November 2023 https://www.unicode.org/events/utw/documents/slides/UTW_2023-Internationalizing_DNS_v2.pdf

16.1 Data

European

[EURID] EURID, ".EU" Registry manager, Rules for domain names, accessed 14 January 2024, https://eurid.eu/en/register-a-eu-domain/rules-for-eu-domains/

Reserved Names

[Reserved] ICANN, "Reserved Names for gTLDs", accessed 14 January 2024, https://www.icann.org/reserved-names-en

Root Zone Confusables Recommendations

[Ed. Note: this is a placeholder for either a link to a consolidated data collection or the citations of individual source documents.]

The following RZ-LGR proposals contain suggested confusables:

[Proposal-Cyrillic] Section 6 and Appendix C and C, in Cyrillic Generation Panel, "Proposal for Cyrillic Script Root Zone Label Generation Rules", 3 April 2018, https://www.icann.org/en/system/files/files/proposal-cyrillic-lgr-03apr18-en.pdf

[Proposal-Greek] Section 6 and Appendix A, in Greek Generation Panel, "Proposal for a Greek Script Root Zone Label Generation Ruleset (LGR)", 1 February 2022, https://www.icann.org/en/system/files/files/proposal-greek-lgr-01feb22-en.pdf

[Proposal-Latin] Section 6 in Latin Generation Panel, "Proposal for a Latin Script Root Zone Label Generation Rule-Set (LGR)", 27 January 2022, https://www.icann.org/en/system/files/files/proposal-latin-lgr-27jan22-en.pdf and Appendix D and E in Appendices to "Proposal for a Latin Script Root Zone Label Generation Rule-Set (LGR)", 27 January 2022 (ZIP), https://www.icann.org/en/system/files/files/proposal-latin-lgr-appendices-27jan22-en.zip

[Proposal-Gurmukhi] Group 1 and 2 in Section 6, "Candidate Variants", in Neo-Brahmi Generation Panel, "Proposal for a Gurmukhi Script Root Zone Label Generation Ruleset (LGR)", 22 April 2019, https://www.icann.org/en/system/files/files/proposal-gurmukhi-lgr-22apr19-en.pdf

Root Zone Database

[ROOTDB] Root Zone Database, represents delegation details of top-level domain, may contain not assigned TLDs, accessed 14 January 2024, https://www.iana.org/domains/root/db

[TLD-ALPHA] List of alphabetically ordered TLDs, 14 January 2024, unlike previous it does not include unassigned TLDs, https://data.iana.org/TLD/tlds-alpha-by-domain.txt

Unicode Confusable Detection

[UTS#39] Described in Unicode Technical Standard #39, https://www.unicode.org/reports/tr39/#Confusable_Detection.

[UTS#55] Described in Unicode Technical Standard #55, https://www.unicode.org/reports/tr55/#Confusability .

16.2 Additional References

[FastTrack] Final Implementation Plan for IDN ccTLD Fast Track Process, ICANN, 2019 March 28, https://www.icann.org/en/system/files/files/idn-cctld-implementation-plan-28mar19-en.pdf

[IDNA2008] The IDNA2008 specification is defined by a cluster of IETF RFCs:

- Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework, https://www.rfc-editor.org/info/rfc5890
- Internationalized Domain Names in Applications (IDNA) Protocol, https://www.rfc-editor.org/info/rfc5891
- The Unicode Code Points and Internationalized Domain Names for Applications (IDNA), https://www.rfc-editor.org/info/rfc5892

- Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA), https://www.rfc-editor.org/info/rfc5893

There is also an informative document:

- Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale, https://www.rfc-editor.org/info/rfc5894

[IDN EPEP] Phase 1 Initial Report on the Internationalized Domain Names EPDP, GNSO https://itp.cdn.icann.org/en/files/internationalized-domain-names-idn/public-comment-summary-report-phase-1-initial-report-on-the-internationalized-domain-names-epdp-10-07-2023-en.pdf

Final Report on the Internationalized Domain Names EPDP, GNSO https://mm.icann.org/pipermail/council/attachments/20231108/fcbce142/Phase1FinalReportontheInternationalizedDomainNamesExpeditedPolicyDevelopmentProcess-0001.pdf

[Levensthein] Wikipedia, "Levenshtein distance", https://en.wikipedia.org/wiki/Levenshtein_distance

[SubPro] Final Report on the new gTLD Subsequent Procedures Policy Development Process, GNSO, Feb 2nd 2021: https://gnso.icann.org/sites/default/files/file/field-file-attach/final-report-newgtld-subsequent-procedures-pdp-02feb21-en.pdf

Annex A

Considerations for Latin, Greek, Cyrillic and Armenian

A number of issues are common to this group of related scripts and therefore presented with additional detail here. However, it should be noted that some considerations also apply to different script families. For example, the *nukta* found in several of the neo-Brahmi scripts is graphically a diacritic, even though its linguistic function is perhaps a bit different from a typical accent mark. Nevertheless, issues for diacritics that are related for confusability may be applicable to nuktas. Where possible, this will be called out, but the discussion will not be repeated.

A.1 Related Scripts: Latin, Greek, Cyrillic, and Armenian

Latin, Greek, Cyrillic and Armenian are historically close scripts; three of them being derived from Greek. In modern usage, there is a tendency for harmonized forms, which tends to make string more similar across script pairs. Many labels in any of these scripts have cross-script homographs (labels that look identical for all practical purposes or have the same outlines in certain fonts even though all letters are in another script).

A common feature of Latin, Greek and Cyrillic is the use of diacritical marks, whether to indicate stress or some other shift in pronunciation; or to create entirely new letters. Diacritical marks are a source of confusability (See A.2 “Diacritics and Confusability”).

The three scripts are bicameral, with uppercase and lowercase forms having direct equivalences across scripts; however, the equivalence pairs are not always the same for upper and lower case. (See Section 6 “Effect of Casing on Confusability”).

Other typographical features are also similar. Both Latin and Cyrillic have an “æ” digraph for example, which may be confusable with an “ae” sequence. (See A.3 “Digraphs and Ligatures as Atomic Letters”)

Both scripts have italic forms, where some letters assume somewhat different shapes (in addition to an overall slanted appearance). For example, Cyrillic “*и*” appears in italic form as “*u*”, indistinguishable from Latin “*u*”. Similar confusion may exist within a script. An example that is treated as a variant in the Root Zone is Latin “f” *vs.* “ſ”. Here, users who don’t know of the existence of an *f with hook* character may be misled by its similarity with the italic form “*f*”.

The Latin and Cyrillic scripts share the feature that they are among the scripts used for writing a wide array of languages, each using their own, partially overlapping subrepertoire. Almost every one of these languages contributes one or more extensions to the basic repertoire of the script; these extensions can be new letter forms, modified forms of existing letters or base letters modified with diacritics. Generally, users of one language are not familiar with extensions for another language and will not always recognize them as such. Instead, they may accept them as substitutes if they resemble the expected letter enough.

The Root Zone LGR handles the most egregious and clear-cut cases of confusability related to this family of related scripts via in-script and cross-script variants, but there are some letter pairs with more approximate similarity that need to be checked for confusability as part of String Similarity Review. This applies in particular to any confusability between letters or sequences of the ASCII subset (see A.1.1 “ASCII subset of Latin”). The proposal documents for all the scripts discussed in this section contain addenda that identify a starter set of additional potentially confusable pairs not handled by the variant

mechanism. Similar information can be found in the policy documents for certain second level registries such as [EURID].

A.1.1 ASCII subset of Latin

The Latin script contains the ASCII characters as a subset, sometimes called Basic Latin. ASCII characters form LDH labels, which are still the most common form of domain name labels. The Root Zone LGR can be used to validate both ASCII and IDN labels for the Root Zone and to check them for collisions, including collisions between ASCII labels and whole script lookalikes in, for example Cyrillic, but also unrelated scripts.

However, historically, no variants were defined among LDH labels, so the Root Zone LGR does not define variants between letters or sequences of the Basic Latin subset. Any confusability that may exist between them must therefore entirely be resolved by String Similarity Review. Any existing methodologies developed for LDH labels would be applicable for this subset.

However, other Latin labels may also be confusable with ASCII labels, as can other IDN labels. In some cases, the number of confusable code points in the other script is limited, and only some very unusual labels (in the context of the other script) may give rise to confusability issues. In contrast, the overlap with related scripts, particularly Cyrillic is so extensive that a large number of labels would risk perfect spoofing if confusability was not mitigated.

A.2 Diacritics and Confusability[41]

Diacritical marks are small annotations that alter the interpretation of a base letter. Some of the most generic ones have shapes like a comma, bar or period and arranged above or below a letter. Whether the use of a particular diacritic is seen as creating a new letter or merely modifying the base letter in some ways depends on the writing system or the orthography for a particular language; the status may be different for different marks, or different for different combinations of mark and base letter for the same language.

Many scripts have marks that visually combine with, or arrange around a base character, like dependent vowels in the neo-Brahmi scripts. While they use a similar encoding mechanism in Unicode (Combining Mark), they are considered separately from diacritical marks for this discussion.

Users of different languages differ in whether diacritical marks are never used, are seen as optional, are allowed to be dropped in selected circumstances or are required without exception. English generally uses no diacritical marks, but alternate spellings such as:

EXAMPLE: naïve *vs.* naive

commonly exist. Greek users think of accents as indicators of stress and are tolerant when such marks are absent. Swedish users consider e-acute (é) an optional method that helps in disambiguation between certain words that are otherwise spelled the same, but diaeresis on “ä“ or “ö“ is seen as forming new, distinct letters with their own sorting position in the alphabet. German users may think of “ae” as a fallback for “ä”, while English users may see it as fallback for “æ.” In turn, English commonly use forms like “ä” as if they were merely decorated forms of “a” in contexts such as brand names.

As seen from these examples the status of diacritical marks may be different for different marks, or different for different combinations of mark and base letter for the same language. From the standpoint

[41] This is both a generic issue and one most strongly applicable to LGC scripts.

of string similarity this introduces a certain degree of confusability between the unadorned base letter and the correct combination of base letter and diacritic.

Users may be somewhat familiar with a few of the diacritics used in languages that are not their own, but commonly, the full repertoire for a script may contain diacritics that are unexpected or that may occur in unexpected combinations. From the standpoint of string similarity this introduces a certain degree of confusability between different diacritics applied to the same base letter, or base letters presented with diacritics that do not belong. An extra diacritic may be experienced as a "decoration" and an incorrect, but similar-looking diacritic may be attributed to a quirk in font design.

For example, users not familiar with the double acute accent may accept it instead of the double dot above (diaeresis), as in the example of "ű" *vs.* "ü". This is especially true for a user expecting a "ü" for some commonly used word.

Where context is established that limits ambiguity, fonts may even substitute diacritic shapes without affecting readability. For example, fonts might not distinguish between comma below and cedilla below and instead use some generic, wedge-like appendage, relying on locality and language context to firmly establish the intent and thus remove the inherent ambiguity. In the context of strings such as domain name labels lacking any inherent context, confusability concerns arise.

Diacritics tend to be small, which means that an unexpected "extra" one may be overlooked, as in this example where a *nukta* (dot below) is unexpectedly placed under a letter other than one a particular user is familiar with. In Devanagari, the nukta is most commonly placed below a consonant to create a new letter. But in one language it may also appear under a vowel, such as "ऄ". Many users can be expected to miss the dot in this combination "ऄ़". This example also illustrates that some diacritics, whether by shape or position (below) may be harder to recognize or identify than others.

This particular case is addressed in the Root Zone LGR via the variant mechanism, but generally it is neither possible nor desirable to capture the full extent of confusability of diacritics in a set of variant definitions.

A.3 Digraphs and Ligatures as Atomic Letters[42]

For the alphabetic scripts, most digraphs or ligature forms that have an assigned code point in Unicode are not supported in the Root Zone. They are generally hard to distinguish from an ordinary letter sequence, and for the Latin script the tendency would be great to use the sequence as a fallback, or to use a fallback that is a pure ASCII letter sequence. There are some exceptions for letters that, while being digraphs in origin, are treated like first class letters for some languages.[43]

For example, in Danish or Norwegian "æ" is considered a letter that sorts at the 28th place in the alphabet and is therefore distinct from an accidental juxtaposition of "a" and "e". Something similar applies to the French "œ" *vs.* "oe". Both of these are distinct from, but still similar to their corresponding letter sequences. The Dutch "ij" on the other hand is commonly written not as a single digraph letter but

[42] This is mostly a concern for alphabetic scripts, abjads and abugidas have more generalized issues.

[43] The letter "w" is an obvious example of a single letter that had a digraph as its origin. The original sequence "vv" would not make a perfect substitute, but digraphs such as ʼn are often indistinguishable from their sequence (ʼn).

as the two-letter sequence “ij”. This digraph character and other like it, are therefore considered inappropriate for use in Root Zone labels.

Note: in some languages, such as English, the use of these ligatures (“æ” or “oe”) is either optional, as in “aesthetics” or even replaced by a different single letter like “esthetics” for a more traditional or academic spelling of “æsthetics”.

A.4 Considerations from Main Document That Apply to These Scripts

This section denotes general considerations discussed in the main document and that apply to labels in the Latin/Greek/Cyrillic and Armenian scripts or to labels that are confusable with them.

- Uppercase
- Digraphs
- Minimalist Shapes

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel’s working copy of the list above.

A.5 Non-goal: Plurals and Similar Spelling Variations

In certain European languages, plural terms can be formed by very simple spelling changes, such as adding the letter ‘s’. Depending on the language, this can be the dominant pattern, or it can be one of several patterns. This makes the two labels containing singular and plural spelling of the same word, potentially both visually and semantically confusable. The confusion might be less acute whenever the spellings for singular and plural deviate more. However, it is not rare for two unrelated words having the same spelling, and for either spelling to match a different form of a different word altogether (for example, many English verbs have a form that also adds an ‘s’ and many nouns and verbs share spellings).

Finally, the connection between singular and plural spelling is limited to a given language; there are cases where a plural might match the spelling of an unrelated word in a different language, or where languages share a common word, but have different plurals.

EXAMPLE: 'canals' (English) *vs.* 'canaux' (French)
EXAMPLE: 'bureaus' (English) *vs.* 'bureaux' (French)

Such issues make the problem at once not-so-tractable as well as not that different from cases like “color” *vs.* “colour” which are related to regional spelling differences of the same word.

These and similar considerations have been ruled out of scope of String Similarity Review for the time being. They are mentioned here only to note them explicitly as a non-goal. Because of the nature of the European writing systems, including the fact that two of them (Latin and Cyrillic) are shared by very large numbers of distinct languages and regional writing systems, this particular set of issues would apply most notably to the scripts covered here and is therefore discussed in this annex.

Annex B

Considerations for the Arabic Script

This is a placeholder. A full guideline covering aspects specific to Arabic would need participation by experts native to the script; it would even make sense for it to be owned by AF-TIDN.

B.1 Issues that could cause user confusion

1. extra dots

   Some Arabic characters only exist in the number of dots, such as dal with dot below and ddahal.

   ڊ ڍ

   068A 068D

   The letters in this example are both used in the same language, Sindhi, and it can therefore be expected that native users will be alert to the distinction, much like users of many languages know to look for the difference between i and ï.

   There may be other cases that a guideline would need to consider.

2. horizontal *vs.* vertical placement of dots

   Some Arabic characters differ in the horizontal *vs.* vertical placement of dots. One example consists of the two characters *teh* and *tteheh*

   ت ٹ

   062A 067A

   This example has been handled with a variant definition for the Root Zone.

   There may be other cases that a guideline would need to consider.

3. letters that have confusingly similar positional forms

   Arabic letters have up to four positional forms: *initial, medial, final* and *isolate*. Characters only need to be distinct in one of these forms to have been considered for encoding. The Root Zone LGR already captures a large number of examples using variants, but there may be cases where positional forms aren't identical, or confusing only if next to certain other letters.

   A guideline may need to consider such cases.

4. sequences that may look similar

   In addition to letters looking similar, the same may apply to sequences as in this example where the first label has *shin + re* and the second one has three *noons + re*

   ننںر شر

The case with *noon* repeated three times would usually have three separate dots, but sometimes the middle dot may be slightly higher making things more similar. depending on font.

B.2 Considerations from Main Document That Apply to This Script

The following lists general considerations discussed in the main document and that apply to labels in the Arabic script or to labels that are confusable with them.

[Note to reviewers: this will be elaborated upon during the next stage]

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel's working copy of the list above.

Annex C

Considerations for CJK Scripts

C.1 General considerations

There are two distinct definitions of “script” that are relevant in the context of script usage for the East Asian languages. One definition distinguishes the overall writing systems, Chinese, Japanese and Korean, with some distinction between Traditional and Simplified Chinese. However, these writing systems are characterized by the fact that all of them share usage of the large set of Han characters which may be augmented by also using other scripts, such as the Latin Script (or at least the ASCII subset), or native Japanese scripts like Hiragana and Katakana or the Korean Hangul.

The writing systems, then, are combinations of scripts, in the sense the word script is used in the Unicode Standard, but in other classification systems the writing systems as a whole are given a “script” identifier, such as “Kore” for the Korean admixture of Hangul and Han, or “Jpan” for the Japanese admixture of Katakana, Hiragana and Han.

With very few exceptions, like the character for “one” (一) the typical character in any of the East Asian scripts has a visual appearance that ranges from moderately to extremely complex. Native users know how to analyze these shapes visually and will recognize components or clusters of components. They are therefore able to distinguish characters that non-native users would find confusing.

A large number of variants has been defined for the Han scripts, as many signs may have related meanings and may be mistakenly or intentionally be substituted for another character based on the same meaning. A typical case for that is the alternation between simplified and traditional versions of Han characters. Because it represents a simplification in appearance, a simplified character may look quite distinct from the traditional form of a character, although some features are related.

In contrast, there are fewer cases of outright visual similarity of the kind where one shape accidentally looks similar to another. The work on the Root Zone identified a number of Katakana, Hiragana and Hangul character that happen to be close enough in appearance to a Han character that they could be classed as visual variants. However, these variants do not exhaust the set of all visually similar, or confusingly similar pairs of characters. This leaves scope for some additional work as part of String Similarity Review.

A common type of similarity for CJK is characters that share components that are visually distinct at full size but become visually confusable when sized down as a component or even radical (example components like moon/meat 月, 肉 or soldier/earth 士,土).

C.2 Considerations from Main Document That Apply to CJK Scripts

The following lists general considerations discussed in the main document and that apply to labels in the CJK scripts or to labels that are confusable with them.

[Note to reviewers: this will be elaborated upon during the next stage]

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel’s working copy of the list above.

Annex D

Considerations for Neo-Brahmi Scripts

D.1 Conjuncts etc.

[Note to reviewers: this will be elaborated upon during the next stage]

D.2 Considerations from Main Document That Apply to Neo-Brahmi Scripts

The following lists general considerations discussed in the main document and that apply to labels in the Neo-Brahmi scripts or to labels that are confusable with them.

[Note to reviewers: this will be elaborated upon during the next stage]

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel's working copy of the list above.

Annex E

Considerations for SEA Scripts

E.1 General Considerations.

[Note to reviewers: this will be elaborated upon during the next stage, by adding relevant considerations here, as long as unique to SEA scripts. These scripts are Khmer, Lao and Thai.]

E.2 Considerations from Main Document That Apply to SEA Scripts

The following lists general considerations discussed in the main document and that apply to labels in the SEA scripts or to labels that are confusable with them.

[Note to reviewers: this will be elaborated upon during the next stage]

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel's working copy of the list above.

Annex F

Considerations for All Other scripts

[Note to reviewers: this will be elaborated upon during the next stage]

F.1 General Considerations.

[Note to reviewers: this will be elaborated upon during the next stage] All scripts not covered explicitly elsewhere like Ethiopic, Georgian, Hebrew. Some share accidental confusables with some other scripts, but nothing systematic; for a start look for any variants defined and then consider whether some additional code point pairs also fall high-ish on the similarity scale.

F.2 Considerations from Main Document That Apply to Other Scripts

Any suitable.

[Note to reviewers: this will be elaborated upon during the next stage] List

The String Similarity Review Panel will be able to consult with script experts who may decide to identify other considerations described in this document or elsewhere and add them to the panel's working copy of the list above.