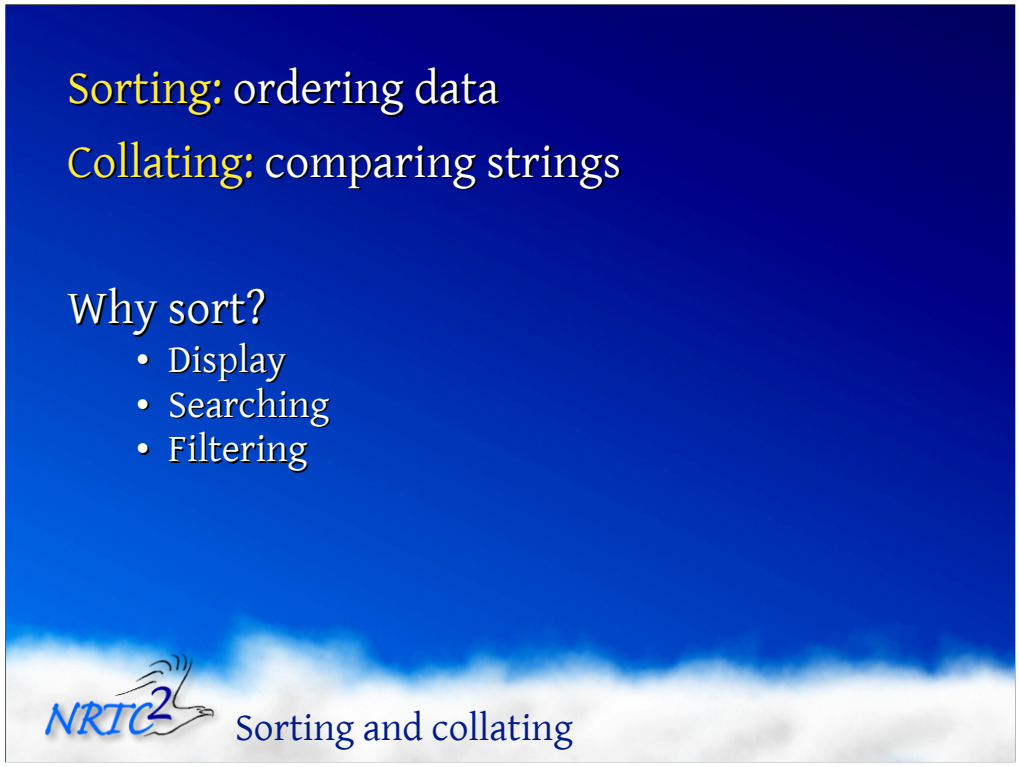
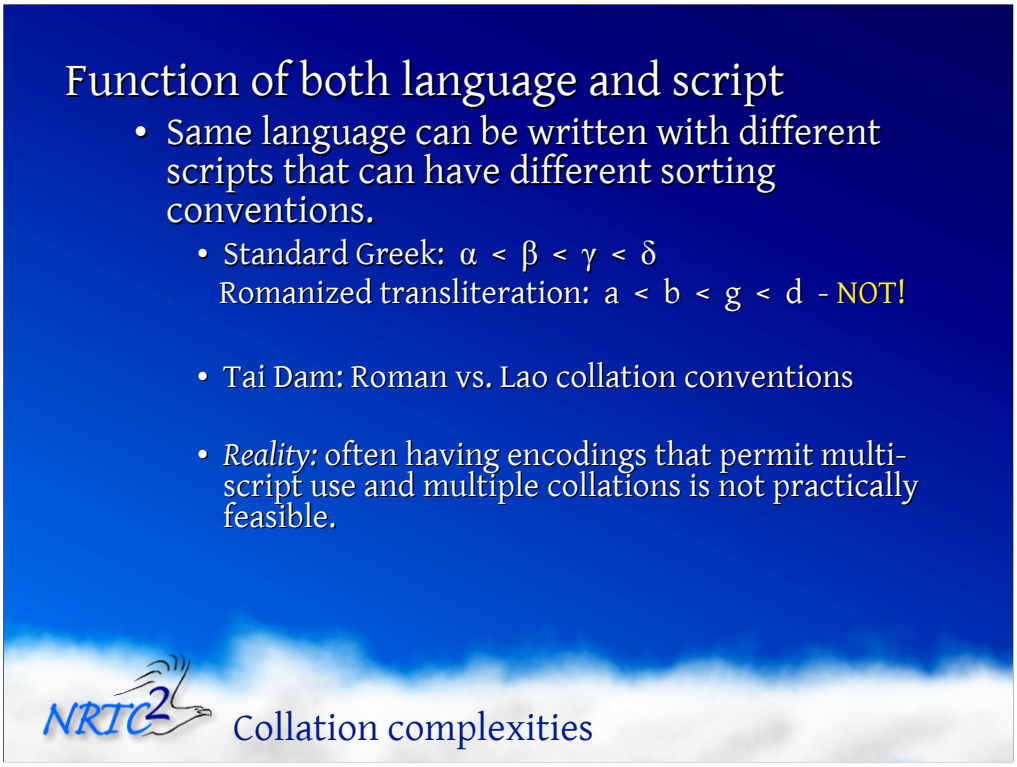
This presentation describes the complexites of sorting textual data in a way that is natural and appropriate for human use. Specifically it describes a standard collation algorithm and how it can be used to implement sorting of multilingual data.



The words "sorting" and "collating" are often used interchangeably, but they have slightly different meanings. Sorting refers to the process of putting a set of data in a specified order, while collating refers to the process of comparing two strings to determine which is "greater", or if they are equal. Collating can be thought of as a fundamental part of the process of sorting.

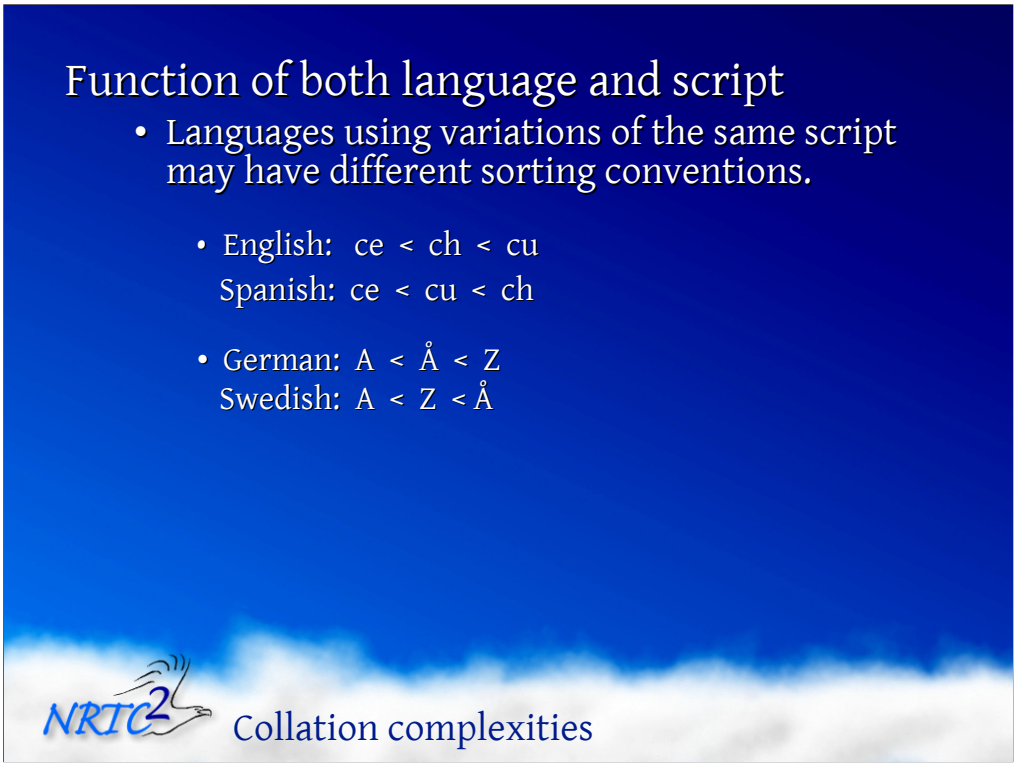
Sorting, in turn, is needed to allow other kinds of processing to occur. Often data needs to be displayed in sorted order. Searching sorted data is much more efficient than searching randomly-ordered data. Sorting and collating are also used to support the process of filtering to display only data the meets certain specifications.



Sorting is a function of both language and script. A single language can be written with different scripts that have distinct sorting conventions. For instance, in standard Greek, the letter gamma collates between beta and delta. But when Greek data is transliterated into the Roman script, the result is incorrectly ordered data. This is because gamma is transliterated to "g", which does not come between "b" (beta) and "d" (delta).

Similarly, the Tai Dam language can be written with both the Roman and Lao scripts, which have different collation conventions. So if a set of data stored in Roman script and sorted according to the Roman conventions, then transliterated to the Lao script, the result will be sorted incorrectly according to the Lao conventions.

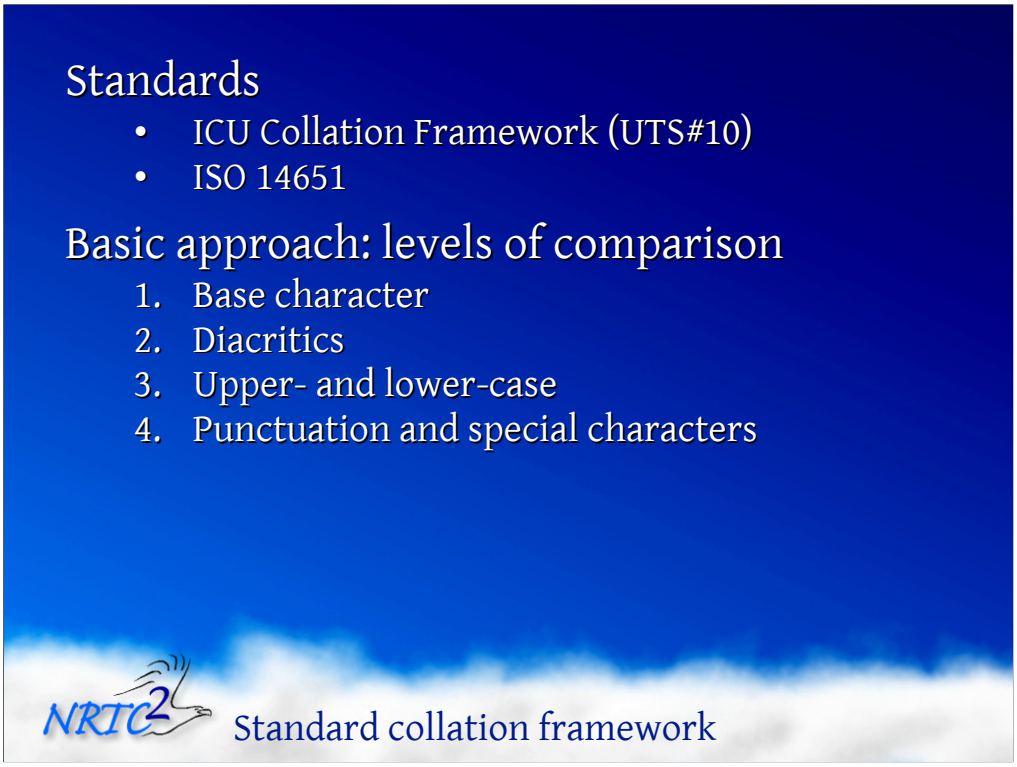
It is often considered a desirable goal on the part of script implementers to define a single encoding for languages that can be written with more than one script, making transliteration between the two scripts unncessary. However, this problem of inconsistent collation is one of the reasons why this goal is rarely feasible.



Not only can the same language be written with different scripts having different sorting conventions, but even a single script can have different conventions when used for different languages. There are several examples of this in Roman script. In the traditional Spanish sorting convention, “ch” is considered a separate grapheme and is sorted as its own letter. So a Spanish word beginning with “ch” would follow those beginning with “cu”, which does not happen in English and other Roman-script languages.

Similarly, the A-ring character is used in both German and Swedish. In German, however, this character is considered a special form of the letter A and is ordered accordingly, while in Swedish it is considered a completely independent character and is placed after the letter Z.

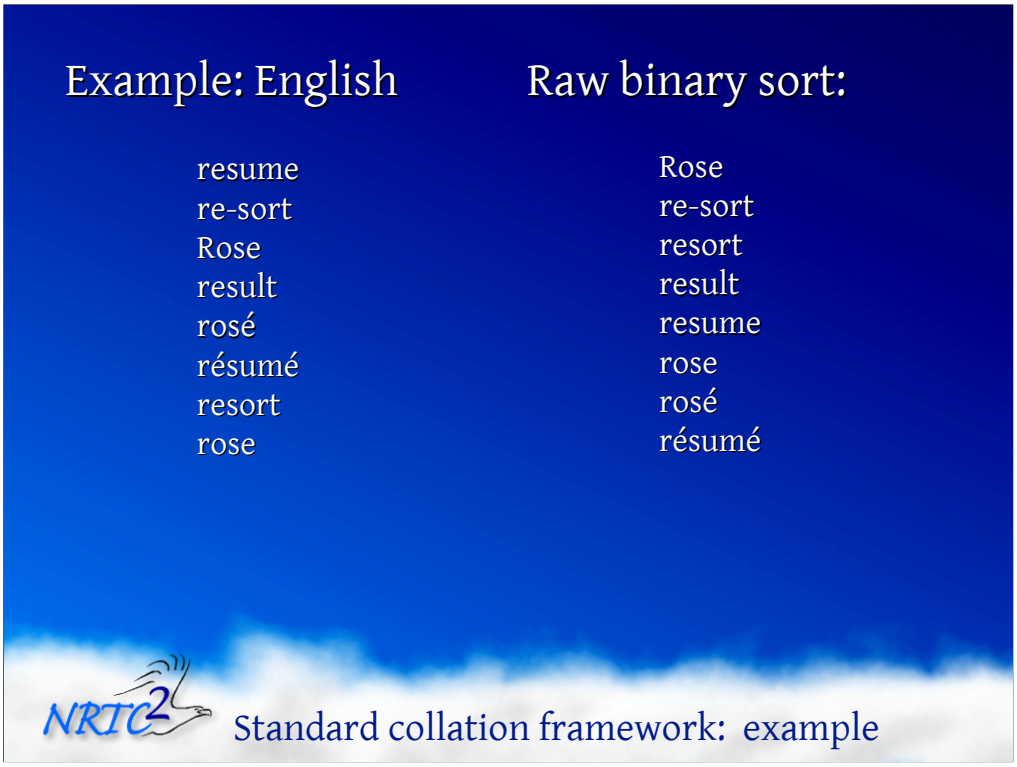
It is even possible for different collations to exist depending on the type of information being sorted. Often dictionaries use a different sorting convention from a telephone book, which may in turn be different from a library card catalog system.



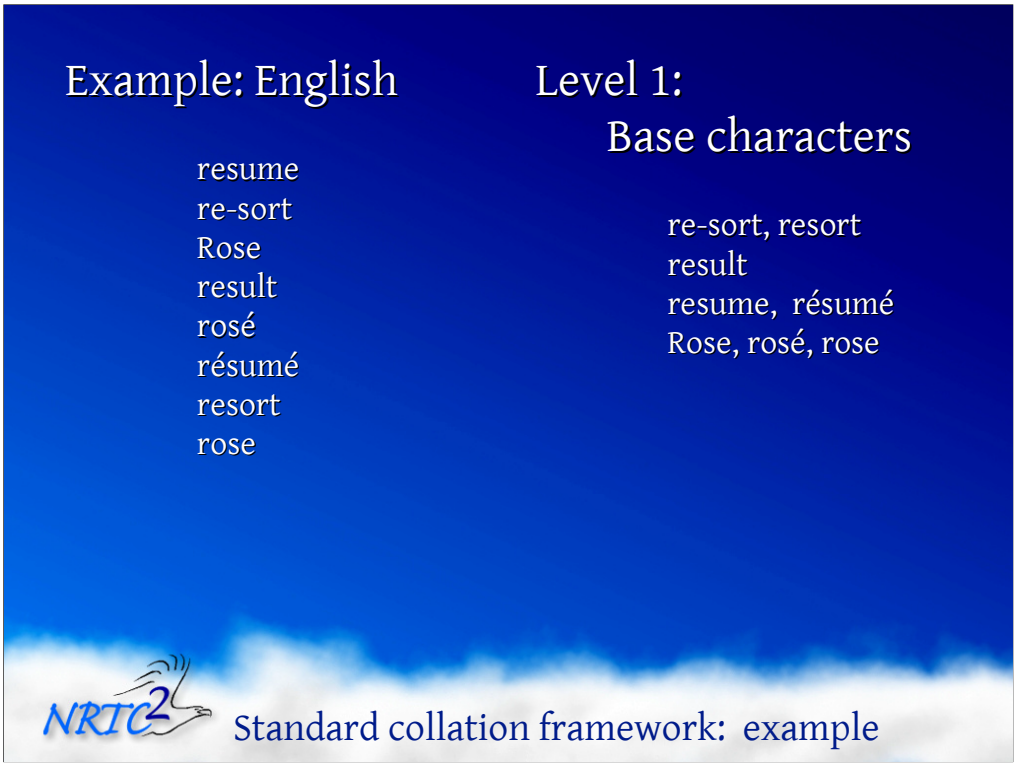
There is an approach to implementing collations that been adopted by several standards bodies such as Unicode and ISO. It involves defining multiple levels of distinction at which characters are compared.

A typical set of levels is shown above: the most fundamental level involves comparing base characters, while ignoring case and diacritics. If two strings are are equal at this level, the diacritics are compared. If necessary, case is considered at the third level, followed by punctuation and special characters.

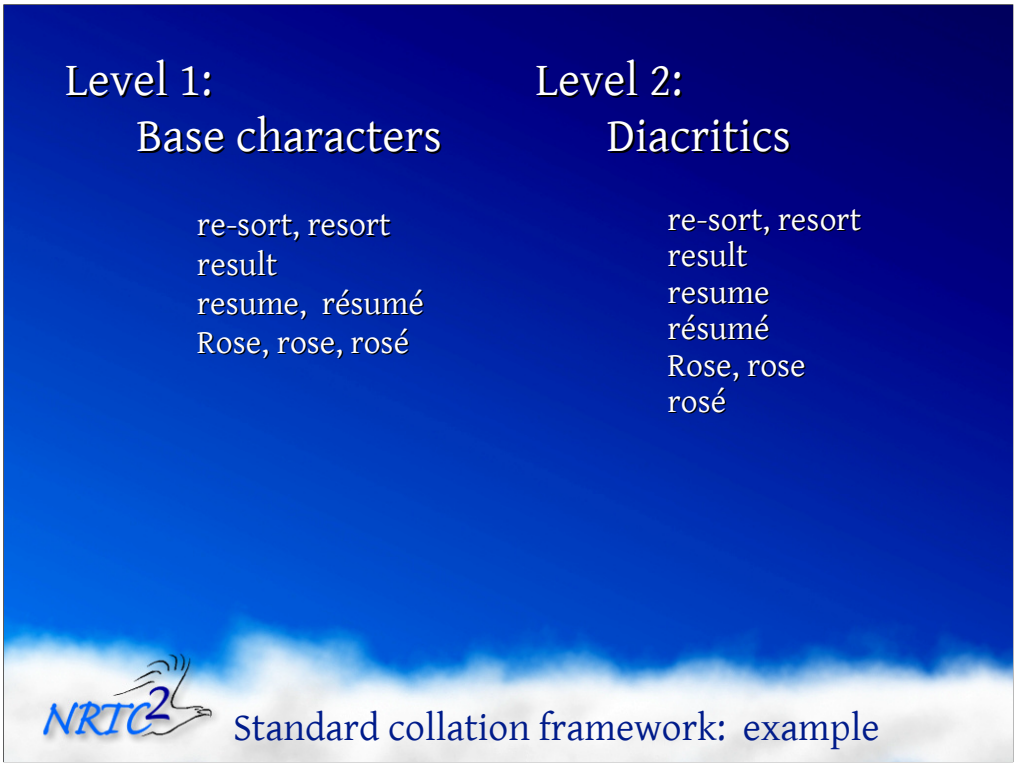
Note that these levels are what is defined and work well for a wide variety of languages. However, it is possible to to use the standard framework to define the levels differently. For phonetic data, for example, one might want to compare consonants at the first level and vowels at the second. In some situations case might be considered more signficant than diacritics, and so correspond to a higher level. The same framework can be used to ignore certain characters altogether.



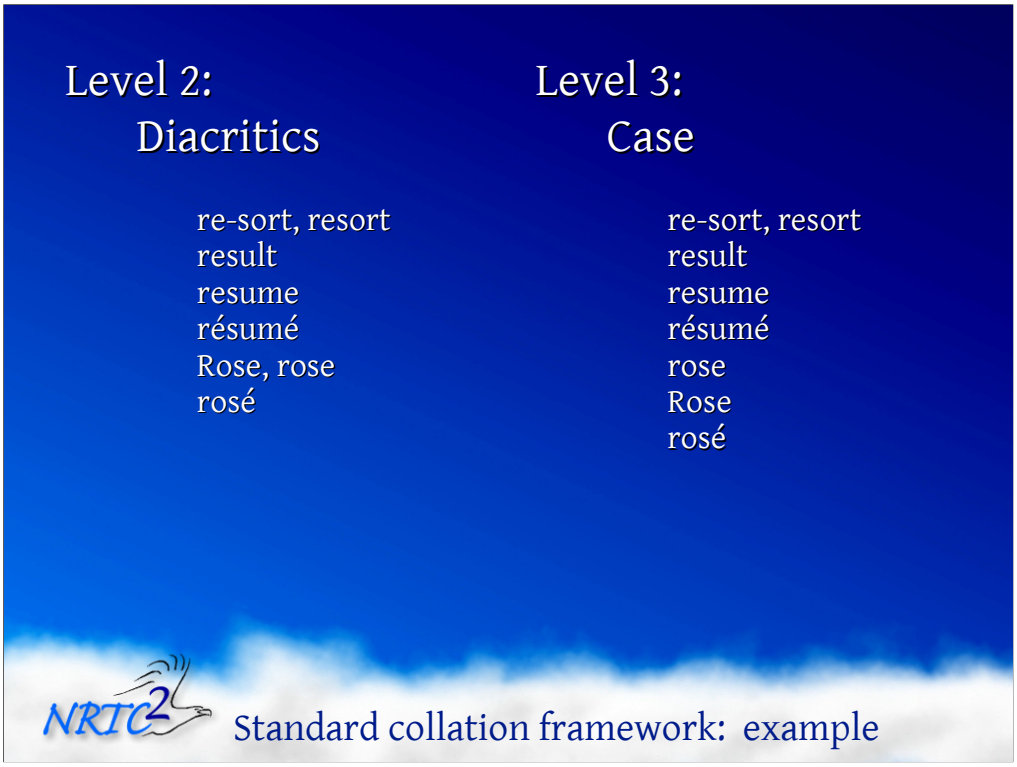
We can see the need for a collation framework by comparing it with a raw binary sort of English data, whose writing system is not even very complex. Above we can see the results of sorting a set of English words using the raw Unicode character values. Some of the more blatant problems are the fact that the capitalized word is placed at the beginning of the list, and the e-with-acute is placed at the end.



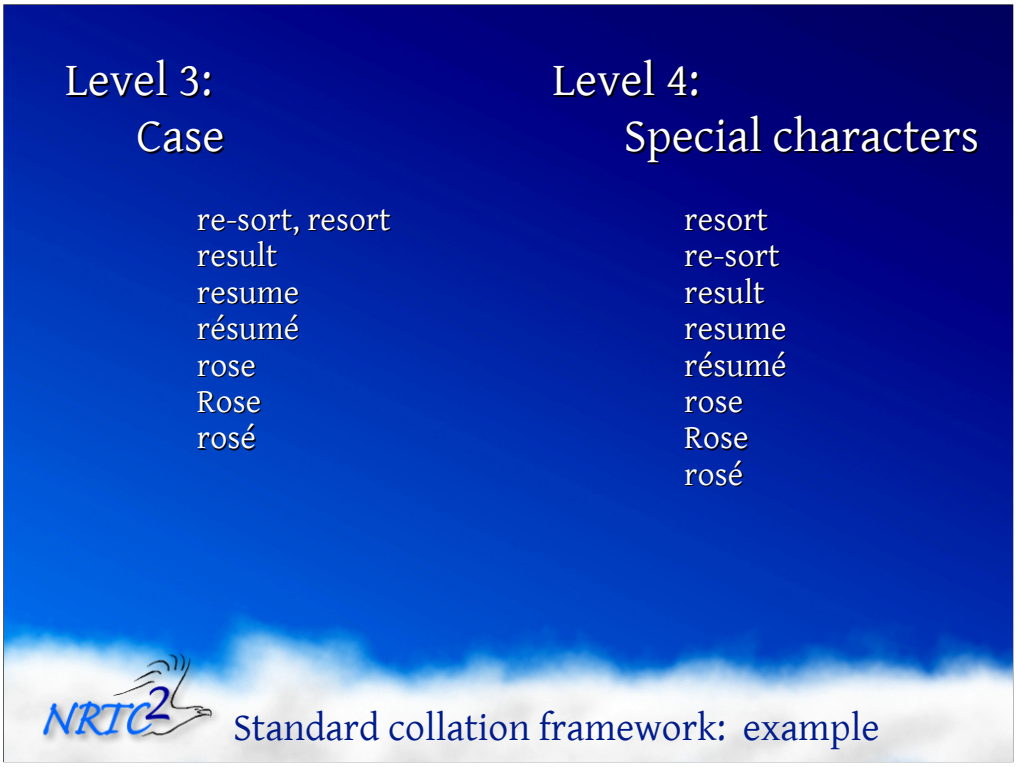
At the first level of comparison, we compare all the base characters, ignoring case, diacritics, and special characters. The result is shown above on the right, where words on the same line are considered equivalent.



Next we apply the second level of comparison, which places characters without diacritics before those with them.

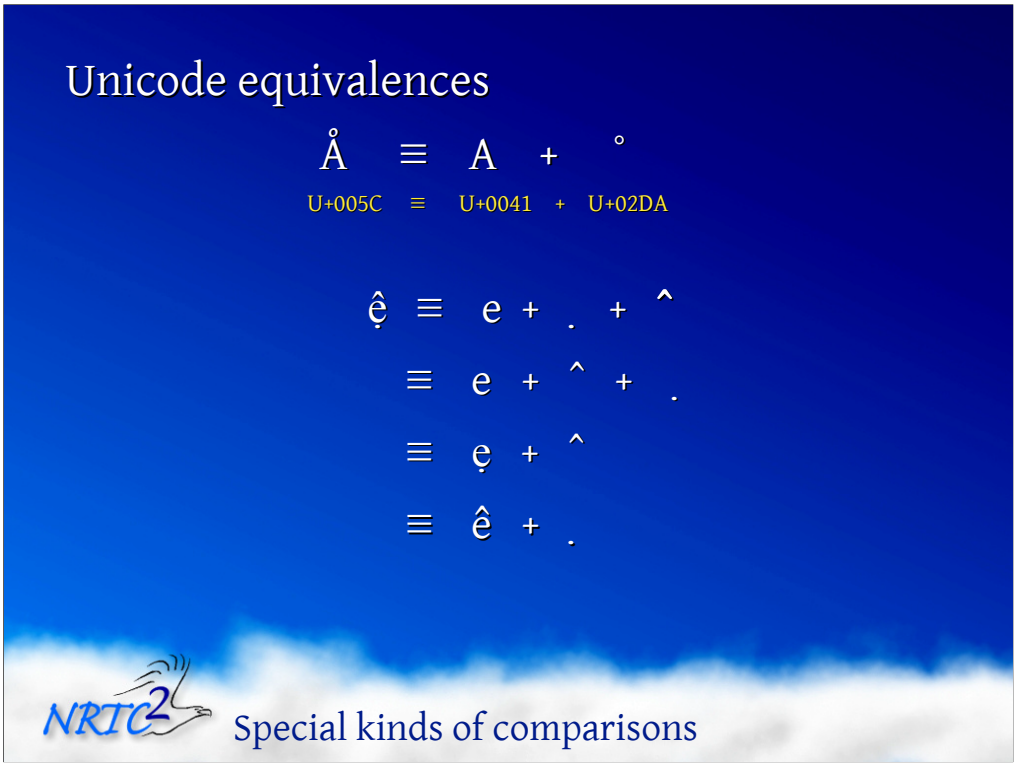


At the third level, upper-case characters are compared against lower-case. In our example, this causes “Rose” to be placed after “rose” in the sorted list.

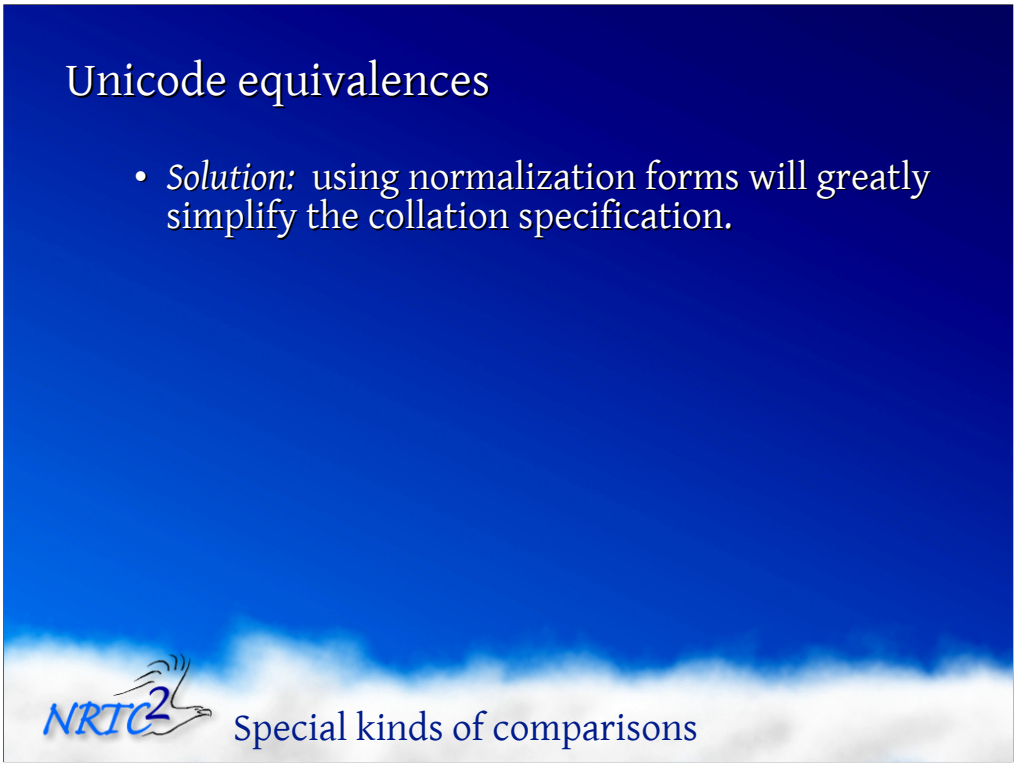


Finally, specially characters are taken into account, such as the hyphen in "re-sort."

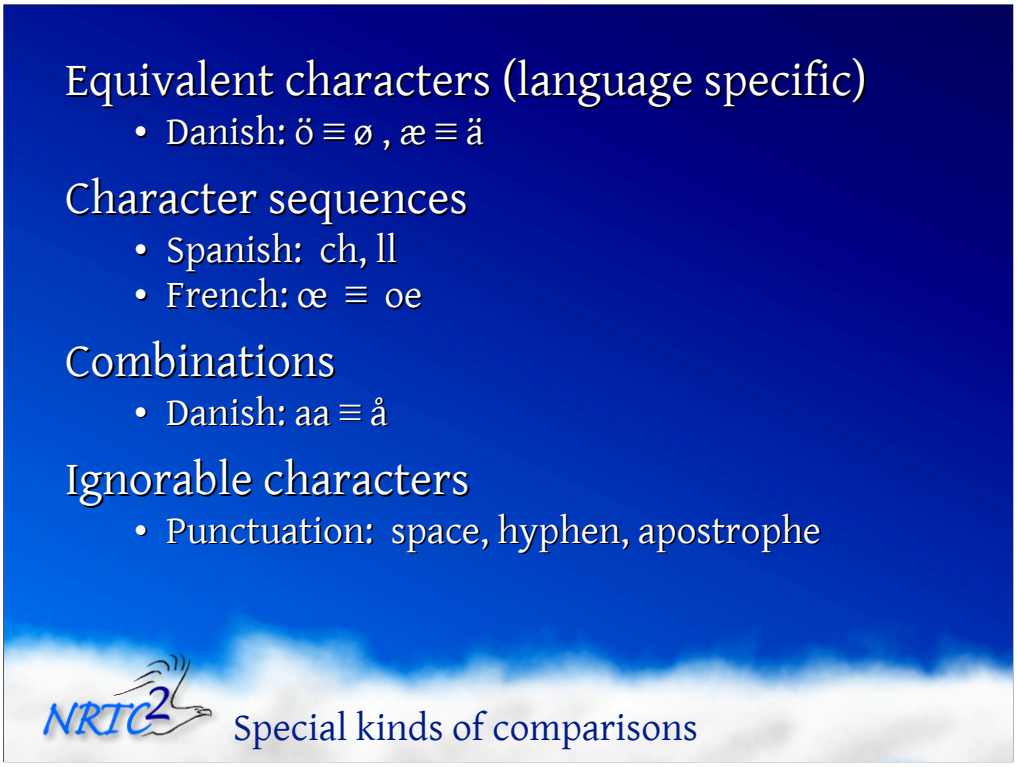
Note that even after applying the fourth level, it is possible for there to be differences between the strings that are not taken into consideration. These words are considered to be exactly equal for the purposes of the colleciton.



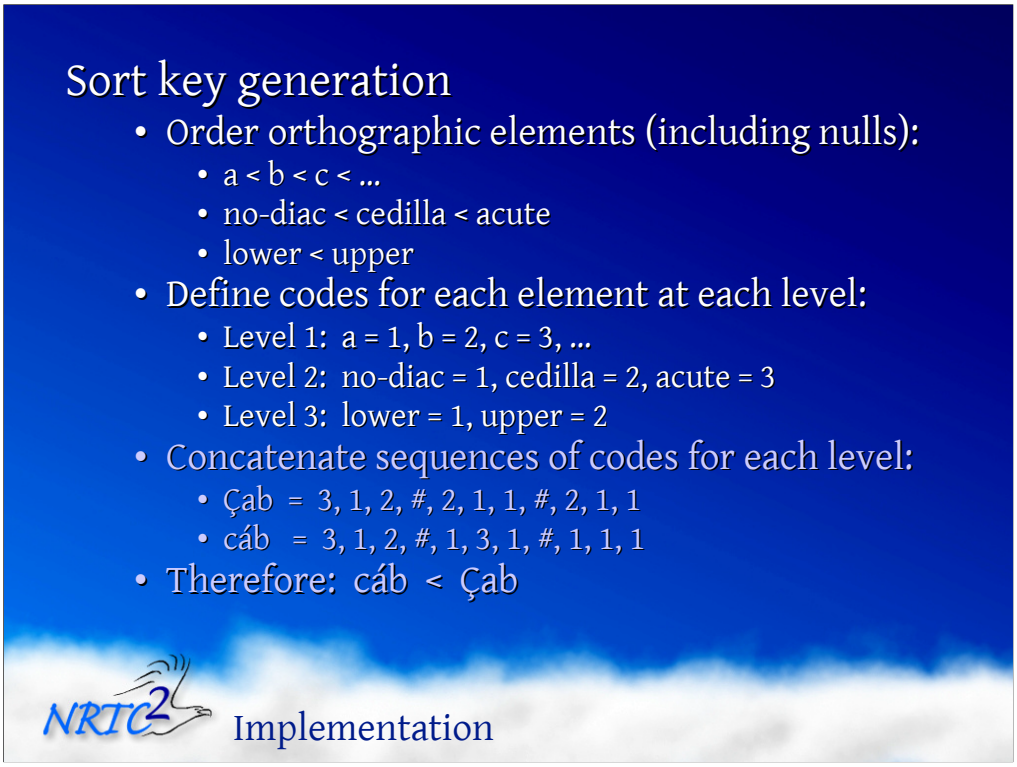
When implementing a collation algorithm, there are special kinds of issues that need to be taken into account. First of these is the fact that Unicode defines certain character sequences to be canonically equivalent, which means all processes must treat them identically. In particular, canonically equivalent character sequences must be considered exactly equal by the collation algorithm. Above are shown some examples of sequences that are canonically equivalent.



When implementing a collation algorithm, putting your data in normalized form, either NFC or NFD, will greatly help to produce a Unicode-compliant process.



In addition to Unicode canonical equivalences, there are language-specific equivalences that must be taken into account. Some of these are one-to-one correspondences, and others involved multiple characters. In addition, there are special characters that may be considered ignorable in some languages and sorting situations.

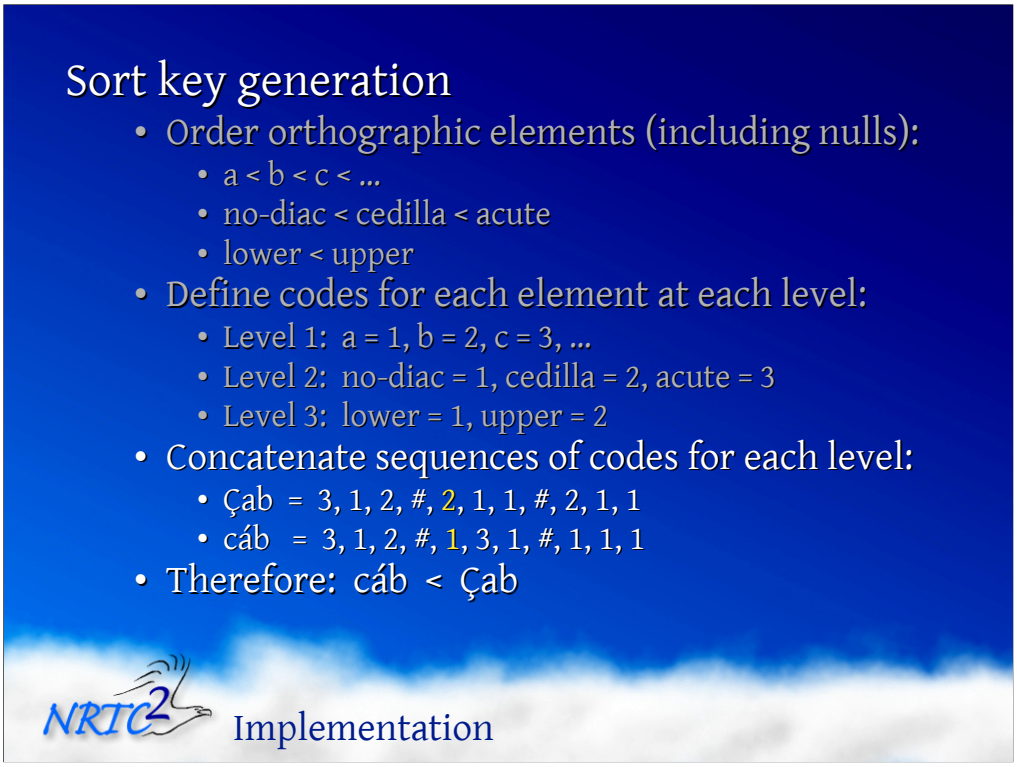


Above is shown the outline for an algorithm based on the standard collation framework. This could be used as the basis for a collation module in an application program.

The first step is to define an order for all the orthographic elements in the writing system. This must include null elements, such as the lack of a diacritic, as well as any signficant character "features" such as case.

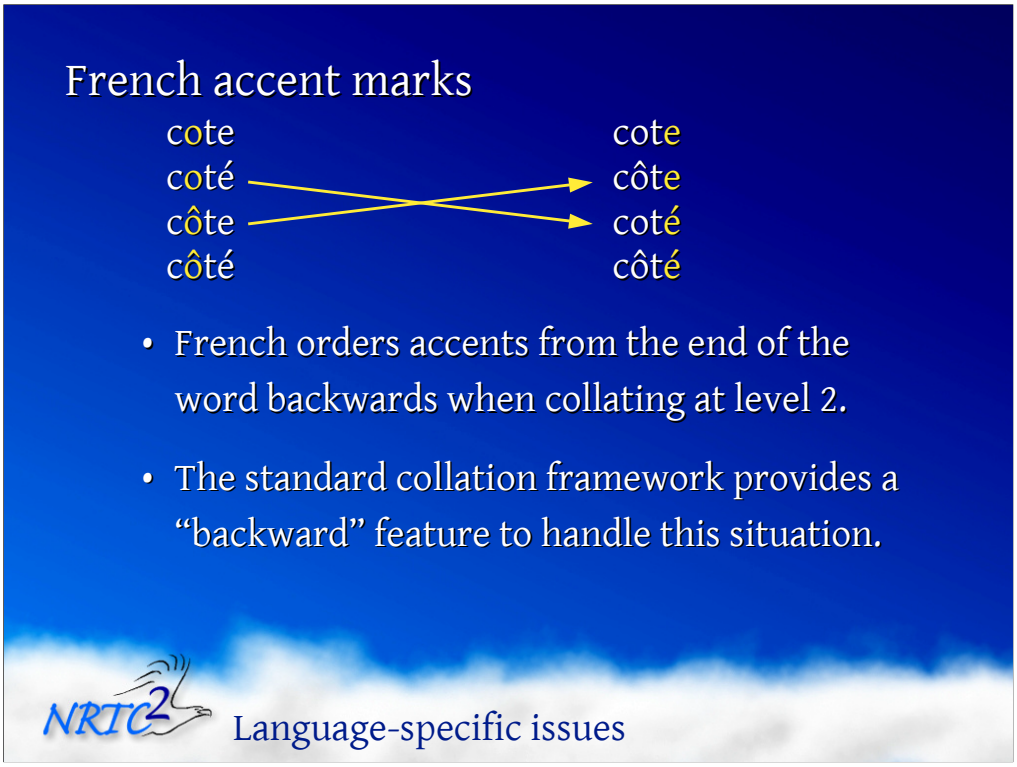
Then a numeric code is assigned to each orthographic element. The codes are organized by level, with the elements that are signficant at each level receiving successive values. In the standard Roman collation shown above, base character values receive corresponding integer values at level 1. The "null" diacritic is assigned the lowest value at level 2, and the diacritics of interest receive successive values. At level 3, the lower-case feature is considered "lower" than the upper-case feature, so the former is given the value 1 and the latter 2.

Any characters or features that are not assigned a code are considered insignificant therefore ignorable.



Finally, a code sequence is generated for each string to be compared, with a separator value inserted between levels. Typically the separator is low in value (i.e., zero), causing short strings to be considered "less than" long strings. If the separator is given a high value, long strings will come first.

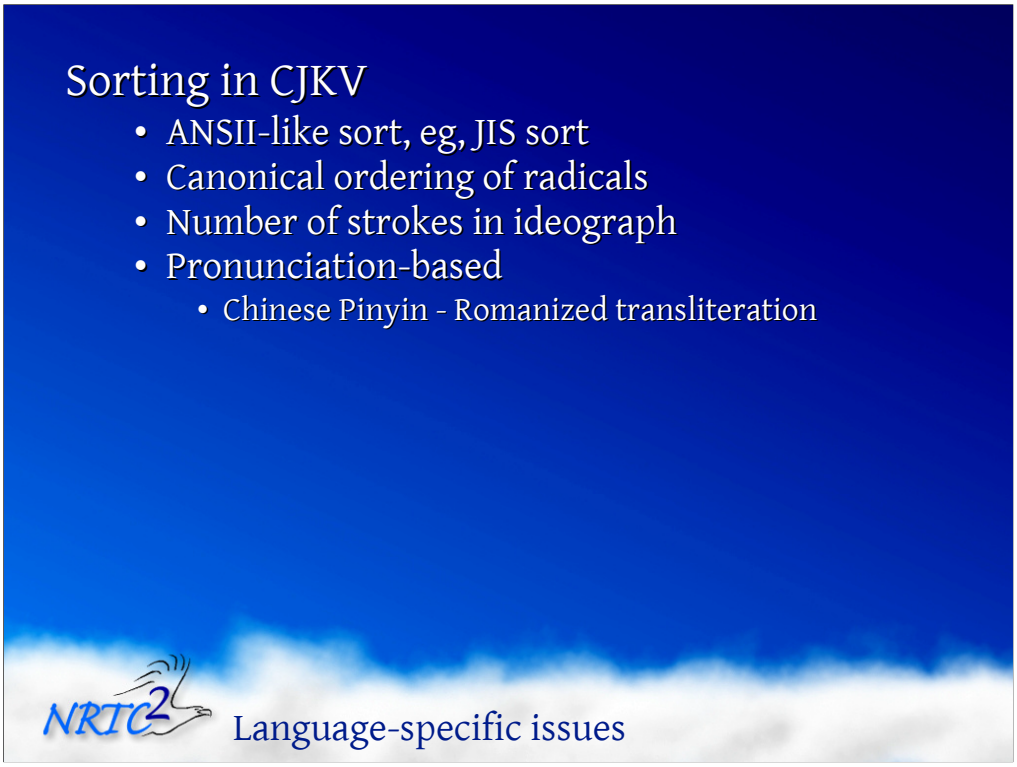
Once the code sequences are generated for each string, it is a simple matter to compare the sequences until a difference is found. In the example above, there are three differences. Two of these are at level 2, representing the differences in the diacritics, and there is one difference at level three, representing the difference in the case of the first letter. The first difference found is on the first letter at level 2, representing the presence/absence of the cedilla. Since this is the first difference found, and according to our ordering no diacritic (1) is considered to be less than a cedilla (2), this produces the answer shown above.



Some extensions must be made to the standard framework to handle a few languages. In French collation, diacritics at level 2 are compared from the end of the word backwards. For this reason the standard collation framework has built a special flag on the second level, indicating that the code sequence must be reversed before doing the comparison.

The standard collation framework does not completely handle sorting in Thai and Lao. The characters in these languages are rendered with certain vowels displayed to the left of (before) the preceding consonant. Although the encoding for these languages matches the rendering order, they are sorted using pronunciation order, that is, as if the vowels followed the consonants.

In order to sort Thai and Lao correctly, a some process must be run over the data to reorder the vowels before calling the standard collation algorithm.



Sorting in CJKV scripts can be very complicated due to the very large number of characters. Above are shown some of the sorting conventions that are in use in these languages.

For More Information Contact:

- Non-Roman Script Initiative
  SIL International
  7500 West Camp Wisdom Rd.
  Dallas, TX 75236
  (972) 708-7440
  nrsi@sil.org

Contact Information