File Comparison

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK

File comparison

In general, the problem of comparing different files (or texts) is difficult.

File comparison

In general, the problem of comparing different files (or texts) is difficult.

It, first, needs to be determined what it means to be different.

File comparison

In general, the problem of comparing different files (or texts) is difficult.

It, first, needs to be determined what it means to be different.

For example, if you save the same text under Unix and under DOS and then compare the files using the Unix utility "diff" it will report that all lines from both files are different. (Why?)

Unit of comparison

What is the primary unit of comparison?

- lines?
- words?
- arbitrary strings?

Unix: diff

Searches for the "longest common subsequence" (Hunt & McIlroy, 1975)

For example:

a b c d e f g
w a b x y z e

Unix: diff

Searches for the "longest common subsequence" (Hunt & McIlroy, 1975)

For example:

a b c d e f g
w a b x y z e

common subsequences: "a b" and "e"

A first algorithm

Save all lines from the first file in an array.
Save all lines from the second file in a second array.

A first algorithm

Save all lines from the first file in an array.
Save all lines from the second file in a second array.

Use nested foreach loops to compare every element of the first array with every element of the second array.

A first algorithm

Save all lines from the first file in an array.
Save all lines from the second file in a second array.

Use nested foreach loops to compare every element of the first array with every element of the second array.

This algorithm can be inefficient. For example, if both files have 30 lines, it will require ...

A first algorithm

Save all lines from the first file in an array.
Save all lines from the second file in a second array.

Use nested foreach loops to compare every element of the first array with every element of the second array.

This algorithm can be inefficient. For example, if both files have 30 lines, it will require ...

900 comparisons in the worst case!
30 comparisons in the best case.

A slightly modified algorithm

Assume that the lines are in the same order.
For example:

a b c d e f g
e w a b x y z

A slightly modified algorithm

Assume that the lines are in the same order.
For example:

a b c d e f g
e w a b x y z

have the longest common subsequence: "a b" and nothing else in common.

Comparing files based on their shared words

- used in information retrieval
- requires pre-processing: extract all words from a text
- this is sometimes called an "index"

Questions

- Create an array of all words that occur in each file or in both files together?
- What if a duplicate word is encountered? Should this be added to the array or should there be a counter?
- Are there any Perl, PHP or Unix functions that eliminate duplicates from an array (maybe after sorting the array)?
- Are there any other data structures that would be better for this than arrays?
- How many times do you need to process each file to collect the information?
- Are there any Perl, PHP or Unix functions that could be used to search through a file and to count occurrences of words in files?