ADEPTIA

Integration of Levenshtein Algorithm with Adeptia

Release Date October 16, 2012

Adeptia Inc.
443 North Clark Ave,
Suite 350
Chicago, IL
60654, USA

Introduction

Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is equal to the number of single-character edits required to change one word into the other. The term edit distance is often used to refer specifically to Levenshtein distance.

The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character.

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

1. **k**itten → **s**itten (substitution of 's' for 'k')
2. sitt**e**n → sitt**i**n (substitution of 'i' for 'e')
3. sittin → sittin**g** (insertion of 'g' at the end).

Implementation of Levenshtein Algorithm in Adeptia

Here is snipped code for the implementation of Levenshtein Algorithm:

```
// Get the value from context variable
String firstString = (String) context.get("firstString");
String secondString = (String) context.get("secondString");

//String firstString = "kitten";
//String secondString = "sitten";
int cost= 0;

firstString = firstString.toLowerCase();
secondString = secondString.toLowerCase();
int[] costs = new int[secondString.length() + 1];
for (int i = 0; i <= firstString.length(); i++) {
      int lastValue = i;
      for (int j = 0; j <= secondString.length(); j++) {
            if (i == 0)
                  costs[j] = j;
            else {
                  if (j > 0) {
                        int newValue = costs[j - 1];
                        if (firstString.charAt(i - 1) !=
secondString.charAt(j - 1))
                              newValue =
Math.min(Math.min(newValue, lastValue),
                                          costs[j]) + 1;
                        costs[j - 1] = lastValue;
                        lastValue = newValue;
                  }
            }
      }
      if (i > 0)
```

```
            cost= costs[secondString.length()] = lastValue;
        }
        //System.out.println(cost);
        // Set the value in process context
        context.put("Levenshtein_distance",cost);
```

Adeptia Integration

We can use this plug-in in Adeptia process to find the **Levenshtein distance** between two strings