СРАВНЕНИЕ ИЕРАРХИЧЕСКИХ МОДЕЛЕЙ ДАННЫХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Лычковский А. В.

Волорова Н. А. – к-т. техн. наук, доцент

В результате деятельности человека накапливается информации, которая, как правило, хранится и обрабатывается в электронном виде. По своему происхождению и предназначению эта информация может выполнять разные функции: описывать некоторые объекты, хранить их состояние, содержать произвольные данные. Часто возникает необходимость сравнения имеющейся версии информации с тем, что было некоторое время назад. Также частым является случай сравнения информации полученной из разных источников.

Существует много областей, где сравнение данных является частью процесса работы. Рассмотрим некоторые примеры информации, которая может нуждаться в сравнении по тем или иным причинам.

Часто возникает задача конфигурирования программного обеспечения схожим образом, но с сохранением некоторых персональных параметров конфигурации. Также может возникнуть задача нахождения различия в конфигурации программного обеспечения для устранения неправильной работы или в других целях.

```
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = true
  hideDotFiles = dotGitOnly
[branch "master"]
  remote = origin
  merge = refs/heads/master
[branch "dal-222-main"]
  remote = origin
  merge = refs/heads/dal-222-main
[push]
  default = upstream
```

```
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = false
  hideDotFiles = dotGitOnly
[branch "master"]
  remote = origin
  merge = refs/heads/master
[branch "dal-222-dev"]
  remote = origin
  merge = refs/heads/dal-222-dev
[branch "dal-222-main"]
  remote = origin
  merge = refs/heads/dal-222-main-new
```

Пример 1 - Разные версии файлов конфигурации, описывающие состояния систем разных пользователей.

Информация, которая требует сравнения, может быть представлена и в графическом виде, который формируется на основании некоторой модели. Примером таких случаев будут являться алгоритмы визуальных языков программирования, структурные диаграммы.

Часто информация может быть разбита на уровни, где уровень будет представлен набором элементов, каждый из которых также может быть разбит на уровни. Информация, разбитая на уровни указанным способом, может быть представлена иерархической моделью данных.

Использование модели данных для представления информации дает некоторые преимущества:

- Абстрагирование от источника информации.
- Абстрагирование от способа представления информации.
- Использование информации разной по структуре.

Разработка эффективного метода сравнения таких моделей данных позволит решать прикладные задачи быстрее и эффективнее.

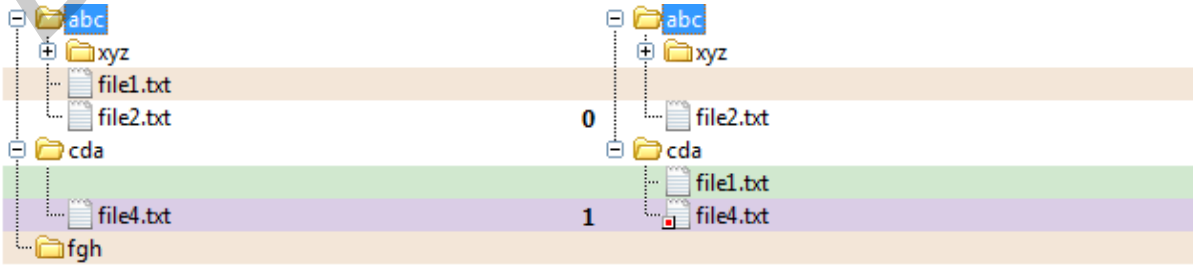

Рисунок 1 - Пример результата сравнения файловых директорий одной из найденных программ.

Существуют готовые решения, для некоторых специфических задач: программы позволяющие сравнивать файлы с данными, файловые директории, онлайн-сервисы, которые сравнивают продукты, продаваемые онлайн-магазинами. Файлы данных и продукты онлайн-магазинов могут быть представлены простой иерархической моделью данный, которая имеет один уровень. Для представления структуры файловой директории понадобятся дополнительные уровни.

Сравнение иерархических моделей данных можно свести к сравнению двух последовательностей элементов. Сравнение должно начинаться с верхнего уровня, а при сравнении элементов этого уровня необходимо учитывать, что каждый из них может также иметь иерархическую структуру. В этом случае каждый из таких элементов сравнивается как самостоятельный элемент. Т.е. мы получаем рекурсивный алгоритм сравнения, который будет продолжаться, пока сравниваемый элемент будет иметь дочерние элементы в своей структуре.

Для оптимальной работы алгоритма описанного выше необходимо решить задачу сравнения последовательностей элементов. В теории алгоритмов эта задача известна как задача о нахождении наибольшей общей подпоследовательности (англ. longest common subsequence, LCS).

Полный перебор. Существуют разные подходы при решении данной задачи при полном переборе — можно перебирать варианты подпоследовательности, варианты вычеркивания из данных последовательностей и т. д. Однако в любом случае, время работы программы будет экспонентой от длины строки.

Метод динамического программирования. Данная задача может быть решена методом динамического программирования. По полученным данным наибольшая общая подпоследовательность может быть восстановлена. Время работы алгоритма будет $O(n_1 \cdot n_2)$.

Расстояние Левенштейна. Расстояние Левенштейна (также редакционное расстояние или дистанция редактирования) между двумя строками в теории информации и компьютерной лингвистике — это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Расстояние Левенштейна может быть обобщено на последовательности элементов. Также можно использовать разные цены для операций вставки, удаления, замены символа. Задача вычисления расстояния Левенштейна может быть решена методом динамического программирования, при этом время работы алгоритма будет $O(M \cdot N)$.

Расстояние Дамерау — Левенштейна. Если к списку разрешённых операций добавить транспозицию (два соседних символа меняются местами), получается расстояние Дамерау — Левенштейна. Для неё также существует алгоритм, требующий $O(M \cdot N)$ операций. Дамерау показал, что 80 % ошибок при наборе текста человеком являются транспозициями. Кроме того, расстояние Дамерау — Левенштейна используется и в биоинформатике.

An O(ND) Difference Algorithm. Этот алгоритм позволяет находить наибольшую общую подпоследовательность строк и требует $O(N \cdot D)$ операций, где N – сумма длин сравниваемых строк, а D – размер минимального редакционного предписания. Алгоритм эффективен, когда сравниваемые сроки очень схожи. В своей работе алгоритм использует графы для решения задачи. Этот алгоритм также можно обобщить для сравнения последовательностей элементов.

В результате проделанной работы были исследованы примеры из реальной жизни, которые делают задачу сравнения иерархических моделей данных актуальной. Были проанализированы существующие решения поставленной задачи. Также были найдены алгоритмы и структуры данных позволяющие решать поставленную задачу. На основании этих алгоритмов может быть разработано программное обеспечение, которое решает задачу сравнения с учетом индивидуальных особенностей иерархических моделей. Это программное обеспечение может быть использовано для решения специфических задач при конфигурировании промышленного оборудования, анализе полученной информации и т.д.

Список использованных источников:

1. Longest common subsequence problem [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Longest_common_subsequence_problem
2. Levenshtein distance [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Levenshtein_distance
3. Damerau–Levenshtein distance [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance
4. An O(ND) Difference Algorithm and Its Variations/ Eugene W. Myers [Электронный ресурс]. – Электронные данные. – Режим доступа: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.6927