TUGAS 3 PEROLEHAN INFORMASI

Batas Akhir Pengumpulan : 17 November 2013

Kesalahan ejaan (*spelling error*) mungkin terjadi pada saat memberikan kueri ke *search engine*. Dengan mengetahui kata-kata lain yang mirip, *search engine* bisa mendeteksi kesalahan ejaan dan memberikan *suggestion*. Kemiripan ini bisa diketahui dengan menghitung ***Levenshtein Distance*** antara dua kata, operasi yang valid pada konsep ini adalah *insert, delete* dan *substitute*. Selain itu, kemiripan juga bisa diketahui dengan menggunakan ***Soundex***, yang mampu mendeteksi kata-kata yang pengucapannya hampir sama.

1. Buatlah program perl yang mengimplementasikan :

   - ***Levenshtein Distance* :**
     - Input : Sebuah file dengan nama "**kueri_LD.txt**", tiap baris berisi sebuah kueri, jumlah kueri >= 1. Format kueri berupa "**inputkata_n"**, dimana **inputkata** adalah kata dengan kesalahan ejaan dan **n** adalah jarak (distance) maksimal yang diinginkan.
     - Proses **:** Program mencari kandidat kata yang mirip dalam korpus untuk tiap kueri.
     - Output : Sebuah file dengan nama "**hasil_LD.txt**", tiap baris berisi kueri "**inputkata_n"** dan hasil dari tiap kueri (urutan baris hasil harus sama dengan urutan kueri). Format output adalah "**outputkata (p)" ,** dimana **outputkata** adalah kata yang mirip dengan **inputkata** (dengan jarak maksimal **n**) dan **p** adalah jarak antara **inputkata** dan **outputkata**.

   - ***Soundex* :**
     - Input **:** Sebuah file dengan nama "**kueri_Soundex.txt**", tiap baris berisi sebuah kueri, jumlah kueri >= 1. Format kueri berupa "**inputkata"**.
     - Proses **:** Program mencari kandidat kata yang mirip dalam korpus untuk tiap kueri.
     - Output **:** Sebuah file dengan nama "**hasil_Soundex.txt**", tiap baris berisi "**inputkata** (kode ***soundex***-nya)" dan hasil dari tiap kueri (urutan baris hasil harus sama dengan urutan kueri). Format output adalah "**outputkata" ,** dimana **outputkata** memiliki kode soundex yang sama dengan **inputkata**.

   - Gunakan korpus yang diberikan, yang merupakan gabungan dari artikel yang anda submit pada tugas 2.
   - Hasil dari tiap kueri berupa daftar kata sesuai format output (maks. 10 kata), tidak boleh *null* atau *empty string*. Contoh format file input dan output terlampir.

2. Buatlah laporan berisi *flow chart* yang mengilustrasikan alur kerja program yang anda buat dan berikan penjelasan yang ringkas tapi lengkap (maks. 2 halaman).

3. Buatlah program se-*simple* mungkin. Tidak perlu mengerjakan program dari nol, boleh memanfaatkan yang sudah ada, pahami dan sesuaikan dengan kebutuhan. Tentukan strategi saat melakukan pencarian sehingga waktu eksekusi program tetap *reasonable*, ini adalah salah satu kriteria penilaian.

4. File program dinamai T3_NPM-Nama.pl dan file laporan dinamai T3_NPM-Nama.pdf, zip dalam satu file T3_NPM-Nama.zip. Program harap disertai dengan dokumentasi yang jelas di dalamnya.

5. Tugas ini diunggah di Scele paling lambat Minggu, 17 November 2013 jam 23.55. Bagi yang terlambat mengumpulkan tugas, silahkan kirim via email ke mirna@cs.ui.ac.id (cc bilih.priyogi31@ui.ac.id), tapi nilainya akan dikurangi 10 poin/hari.

6. Kriteria penilaian berdasarkan implementasi yang dibuat (50 poin), kelengkapan dokumentasi dalam program (10 poin), penjelasan *process flow* dari program (20 poin). Program juga akan dites dengan kata-kata tertentu (20 poin), karena itu pastikan program dapat berjalan dengan baik & benar.

Selamat Mengerjakan ☺

Review

Levenshtein Distance
merupakan salah satu algoritma edit distance yang banyak digunakan. Algoritma ini bertujuan mencari jumlah operasi minimum yang dibutuhkan untuk merubah dari suatu kata menjadi kata lain. Operasi yang dapat dilakukan adalah menambahkan huruf (*insert*), menghapus huruf (*delete*) dan menukar huruf (*substitute*). Pada tugas ini Anda harus mengimplementasikan algoritma ini :

kata yang dicari : "bunga"
kata yang dibandingkan : "nabung" : "nabung" → hapus huruf "n"
"abung" → hapus huruf "a"
"bung"
"bunga" → tambahkan huruf "a"

"tiga" : "tiga" → tukar huruf "t" dengan "u"
"uiga" → tukar huruf "i" dengan "n"
"unga"
"bunga" → tambahkan huruf "b"

Jadi, distance kata "nabung" = "tiga" = 3.

Soundex
merupakan algoritma untuk mengetahui kemiripan kata berdasarkan kemiripan ucapan. Varian dari algoritma ini cukup banyak untuk mengatasi perbedaan kaidah dasar pada bahasa-bahasa yang berbeda. Algoritma ini sudah ada di slide IR, dimana yang dibahas adalah jenis American Soundex. Pada tugas ini Anda harus mengimplementasikan algoritma American Soundex :

1. Pertahankan huruf pertama dari kata, dan buang huruf {a, e, i, o, u, y, h, w}
2. Ubah huruf menjadi angka, dengan ketentuan :
   - {b, f, p, v} => 1
   - {c, g, j, k, q, s, x, z} => 2
   - {d, t} => 3
   - {l} => 4
   - {m, n} => 5
   - {r} => 6
3. Jika dua atau lebih huruf dengan konversi angka yang sama bersebelahan pada kata, hanya ambil huruf pertama. Dua huruf dengan konversi angka sama yang dipisahkan oleh huruf 'h' atau 'w' digabung menjadi satu, sedangkan jika dipisahkan oleh huruf vocal {a, i, u, e, o} tidak digabung.
4. Hasil akhir adalah berupa kode yang terdiri dari 1 huruf dan 3 angka. Jika kata terlalu pendek dan menghasilkan kurang dari 3 angka, tambahkan angka '0' sampai memenuhi 3 angka. Jika kata terlalu panjang dan menghasilkan lebih dari 3 angka, ambil hanya 3 angka pertama yang muncul.

"bank" dan "bang" : "bank" → tandai huruf yang akan dibuang diakhir proses
"ba5k" → ganti huruf 'n' dengan angka '5'
"ba52" → ganti huruf 'k' dengan angka '2'
"b52" → buang huruf yang ditandai sebelumnya
"b520" → tambah angka '0' untuk memenuhi kuota 3 angka

"bang" → tandai huruf yang akan dibuang diakhir proses
"ba5g" → ganti huruf 'n' dengan angka '5'
"ba52" → ganti huruf 'g' dengan angka '2'
"b52" → buang huruf yang ditandai sebelumnya
"b520" → tambah angka '0' untuk memenuhi kuota 3 angka

"herman" dan "hermann" : "herman" → tandai huruf yang akan dibuang diakhir proses
"he6man" → ubah huruf 'r' dengan angka '6'
"he65an" → ubah huruf 'm' dengan angka '5'
"he65a5" → ubah huruf 'n' dengan angka '5'
"h655" → buang huruf yang ditandai sebelumnya, walau 'm' dan 'n' angkanya sama tapi karena dipisahkan oleh huruf vokal, maka konversinya tidak digabung

"hermann" → tandai huruf yang akan dibuang diakhir proses
"he6mann" → ubah huruf 'r' dengan angka '6'
"he65ann" → ubah huruf 'm' dengan angka '5'
"he65a55" → ubah huruf 'n' pertama dan kedua dengan angka '5'
"h655" → buang huruf yang ditandai sebelumnya, walau '**m**' dan '**n**' pertama angkanya sama tapi karena dipisahkan oleh huruf vokal, maka konversinya **tidak digabung**. Sesuai aturan konversi '**n**' pertama dan '**n**' kedua yang bersebelahan**, hanya ambil huruf pertama**.